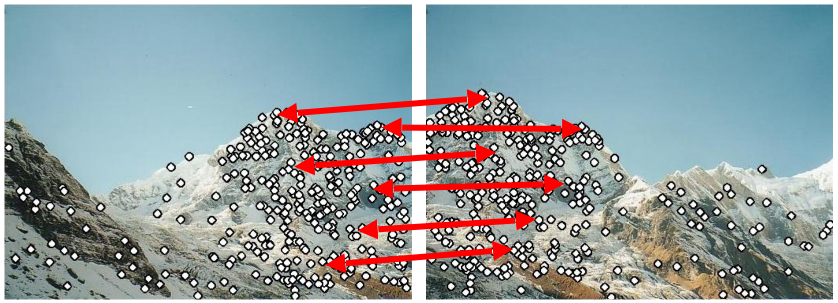
Lecture 5:
Fitting, Hough transforms, RANSAC

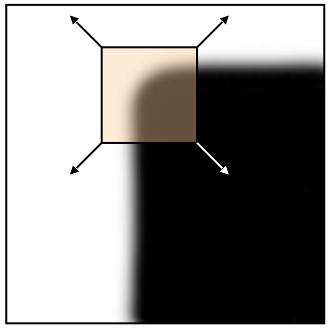
COS 429: Computer Vision

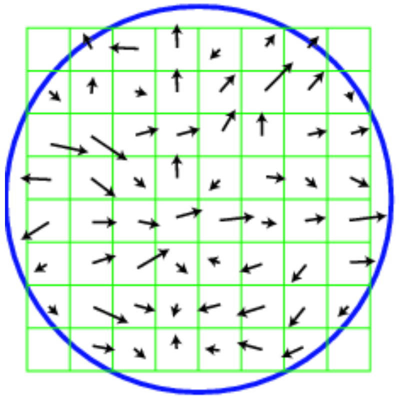
PRINCETON UNIVERSITY

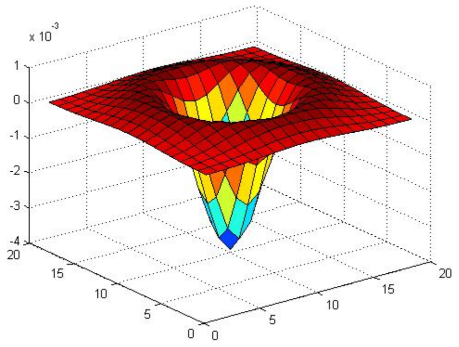
Slides adapted from: Szymon Rusinkiewicz, Jia Deng, Svetlana Lazebnik, David Fouhey

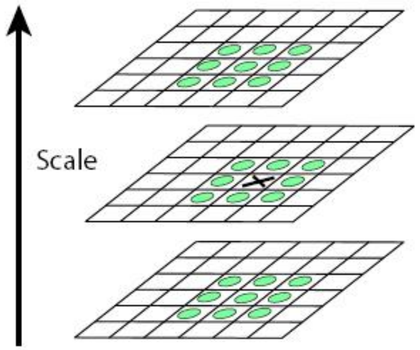
Last time: interest point detection and description

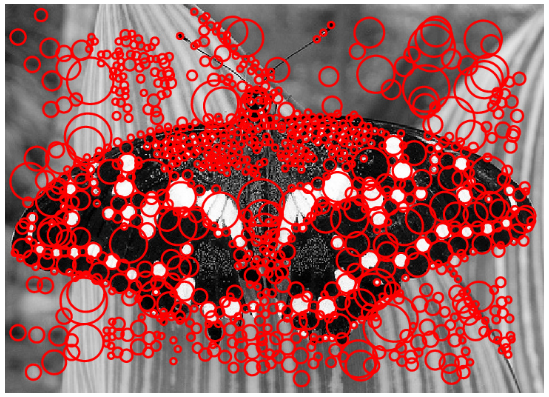




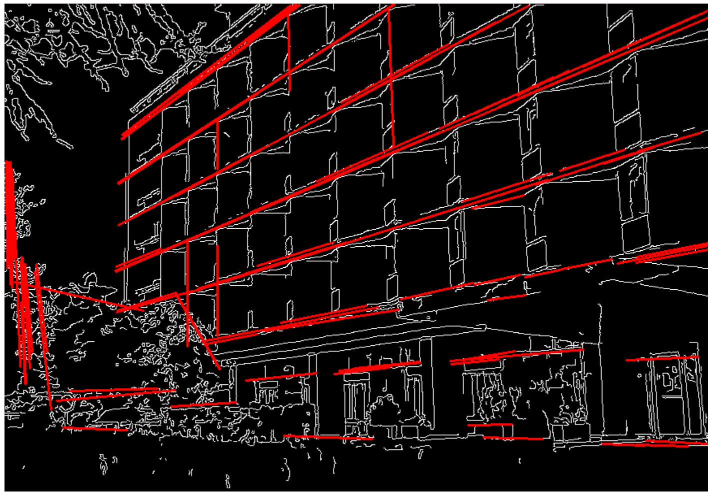






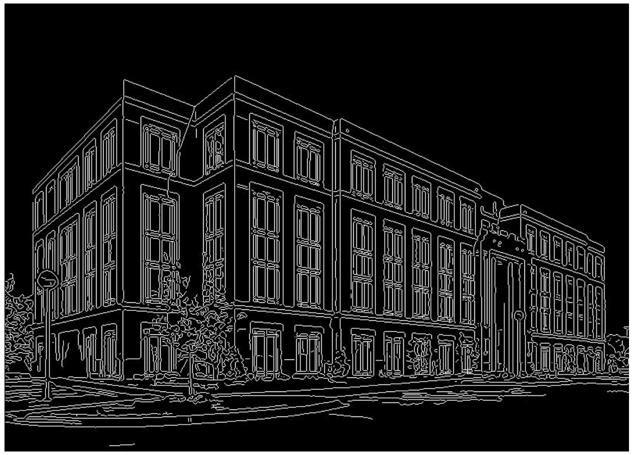


Fitting



Fitting

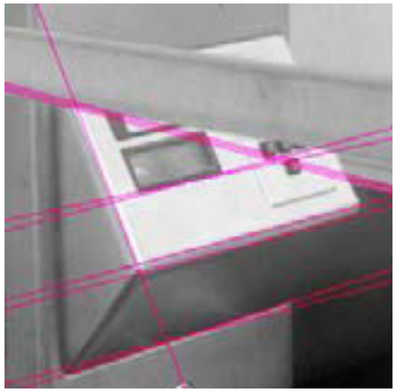
- We’ve learned how to detect edges, corners, blobs. Now what?
- We would like to form a higher-level, more compact representation of the features in the image by grouping multiple features according to a simple model



Slide: S. Lazebnik

Fitting

- Choose a parametric model to represent a set of features


simple model: lines

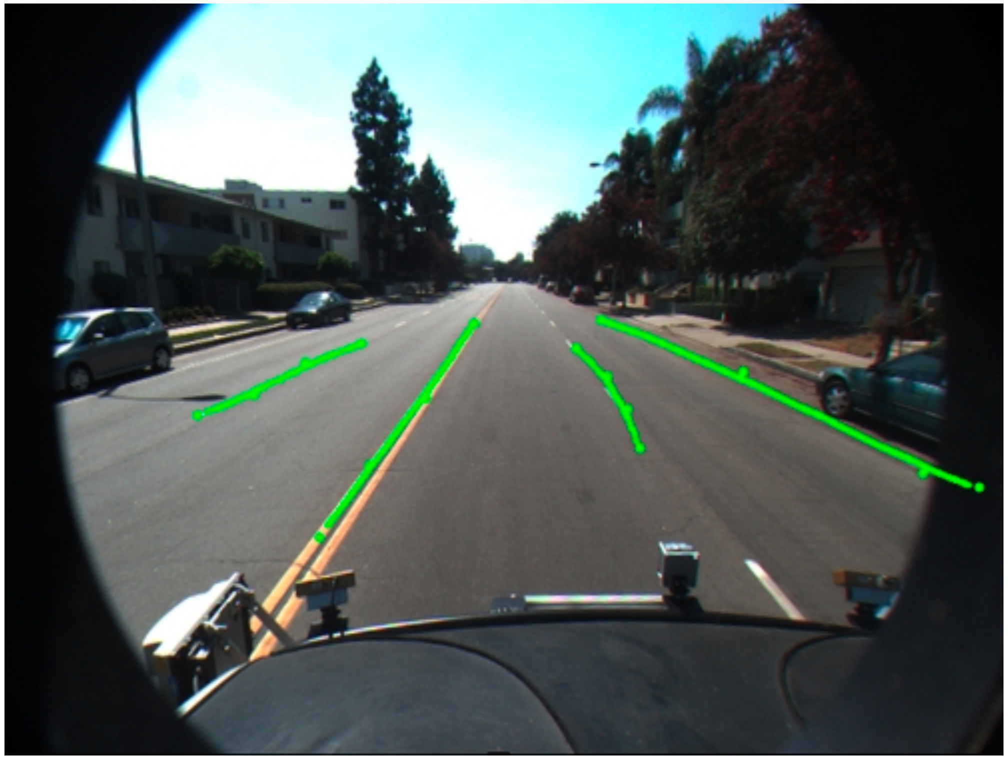
simple model: circles

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

Model fitting

- Need three ingredients:

Data: what data are we trying to explain with a model?

Model: what’s the compressed, parametric form of the data?

Objective function: given a prediction, how do we evaluate how correct it is?

Source: David Fouhey

Example: Least-Squares

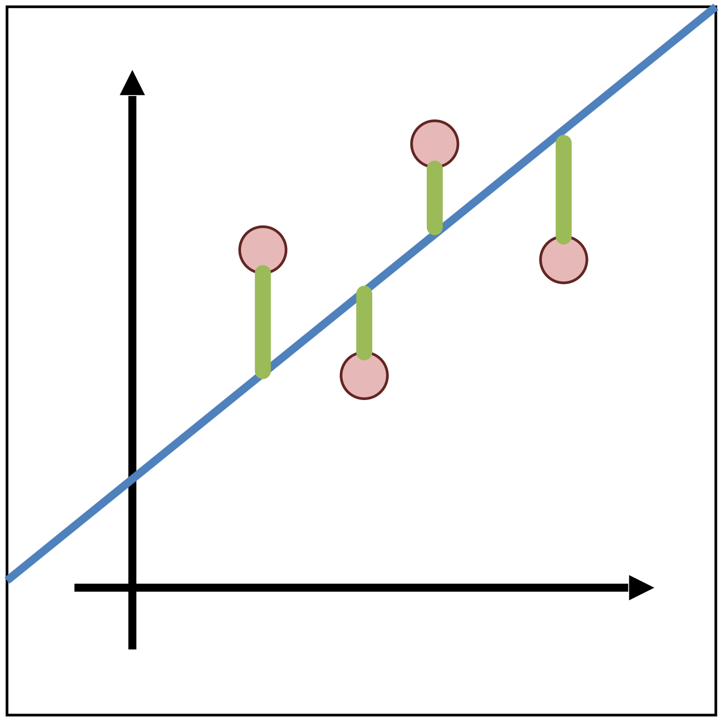
- Fitting a line to data

Data: (x_1,y_1), (x_2,y_2), …, (x_k,y_k)

Model: (m,b) $\quad y_i=mx_i+b$

Or $(\mathbf{w})$ $\quad y_i = \mathbf{w}^T\mathbf{x}_i$

Objective function:
$(y_i - \mathbf{w}^T\mathbf{x}_i)^2$



Source: David Fouhey

Least Squares Setup

$$\sum_{i=1}^{k} \left(y_i - \boldsymbol{w}^T \boldsymbol{x_i}\right)^2 \quad \rightarrow \quad \left\|\boldsymbol{Y} - \boldsymbol{XW}\right\|_2^2$$

$$\boldsymbol{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & 1 \\ x_k & 1 \end{bmatrix} \qquad \boldsymbol{W} = \begin{bmatrix} m \\ b \end{bmatrix}$$

Note: I'm writing the most general form here since we'll do it in general and you can make it specific if you'd like.

Source: David Fouhey

Solving Least Squares

$$\left\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W} \right\|_2^2$$

$$\frac{\partial}{\partial \boldsymbol{W}} \left\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W} \right\|_2^2 = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W} - 2\boldsymbol{X}^T\boldsymbol{Y}$$

$$\mathbf{0} = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W} - 2\boldsymbol{X}^T\boldsymbol{Y}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W} = \boldsymbol{X}^T\boldsymbol{Y}$$

$$\boldsymbol{W} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

Recall: derivative is 0 at a maximum / minimum. Same is true about gradients.

Aside: **0** is a vector of 0s

Source: David Fouhey

Derivation for the Curious

$$\|\boldsymbol{Y} - \boldsymbol{XW}\|_2^2 = (\boldsymbol{Y} - \boldsymbol{XW})^T(\boldsymbol{Y} - \boldsymbol{XW})$$

$$= \boldsymbol{Y}^T\boldsymbol{Y} - 2\boldsymbol{W}^T\boldsymbol{X}^T\boldsymbol{Y} + (\boldsymbol{XW})^T\boldsymbol{XW}$$

$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{XW})^T(\boldsymbol{XW}) = 2\left(\frac{\partial}{\partial \boldsymbol{W}}\boldsymbol{XW}^T\right)\mathbf{XW} = \mathbf{2X^TXW}$$

$$\frac{\partial}{\partial \boldsymbol{W}}\|\boldsymbol{Y} - \boldsymbol{XW}\|_2^2 = 0 - 2\boldsymbol{X}^T\boldsymbol{Y} + 2\boldsymbol{X}^T\boldsymbol{XW}$$

$$= 2\boldsymbol{X}^T\boldsymbol{XW} - 2\boldsymbol{X}^T\boldsymbol{Y}$$

Source: David Fouhey

Two solutions to getting W

In One Go

Implicit form
(normal equations)

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W} = \boldsymbol{X}^T\boldsymbol{Y}$$

Explicit form
(don't do this)

$$\boldsymbol{W} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

Iteratively

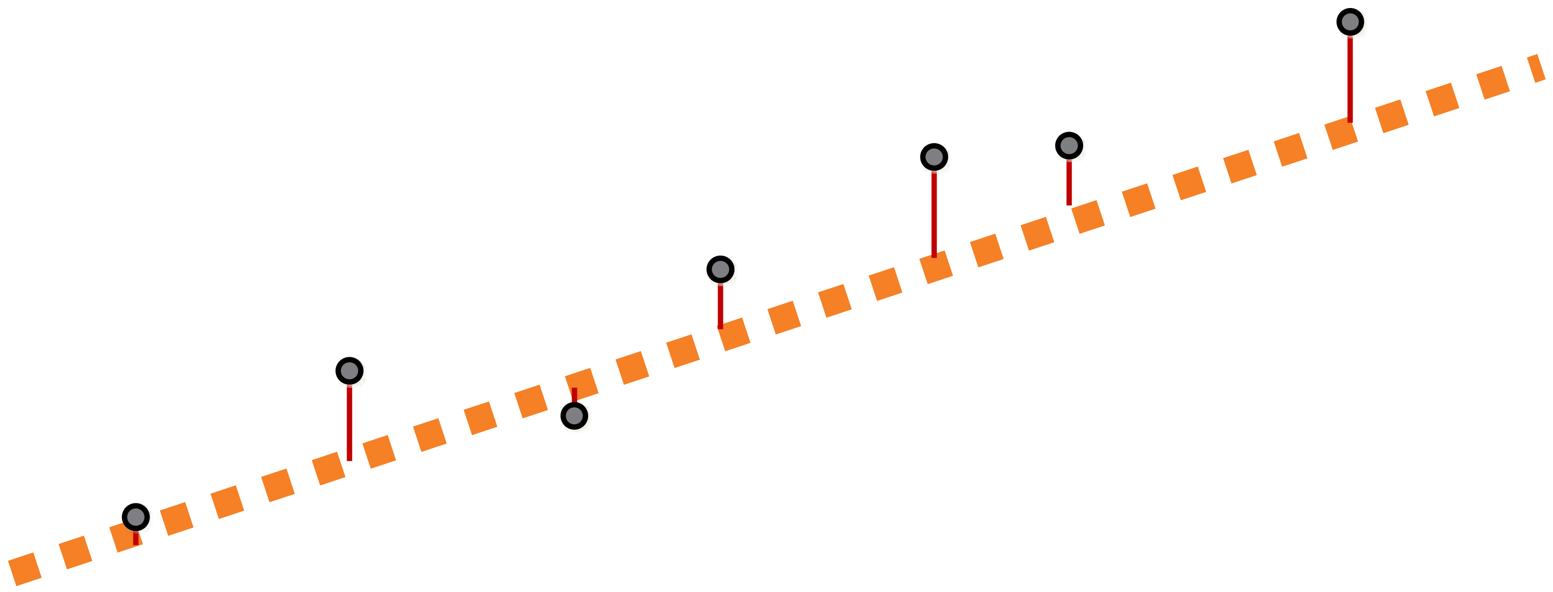
Recall: gradient is also direction that makes function go up the most. **What could we do?**

$$\boldsymbol{W}_0 = \mathbf{0}$$

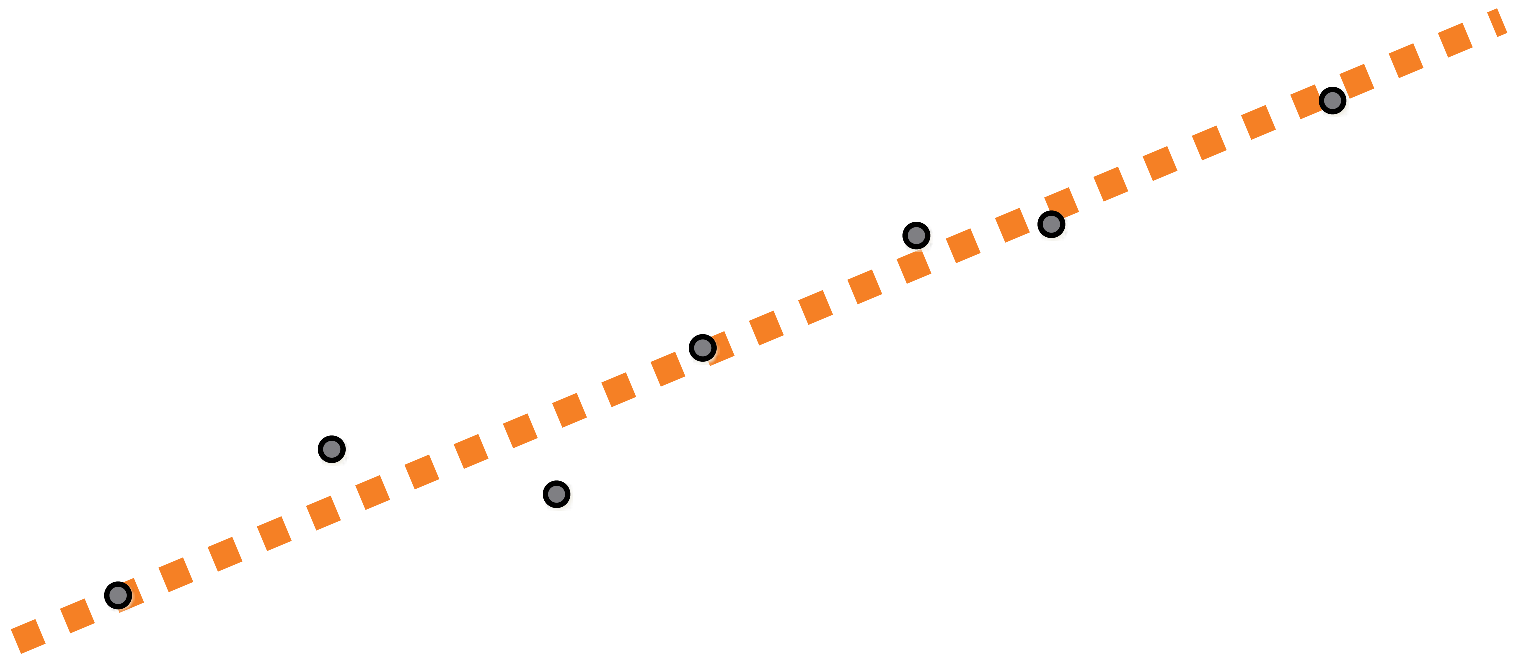
$$\boldsymbol{W}_{i+1} = \boldsymbol{W}_i - \gamma\left(\frac{\partial}{\partial \boldsymbol{W}}\left\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W}\right\|_2^2\right)$$

Source: David Fouhey

Least squares minimization

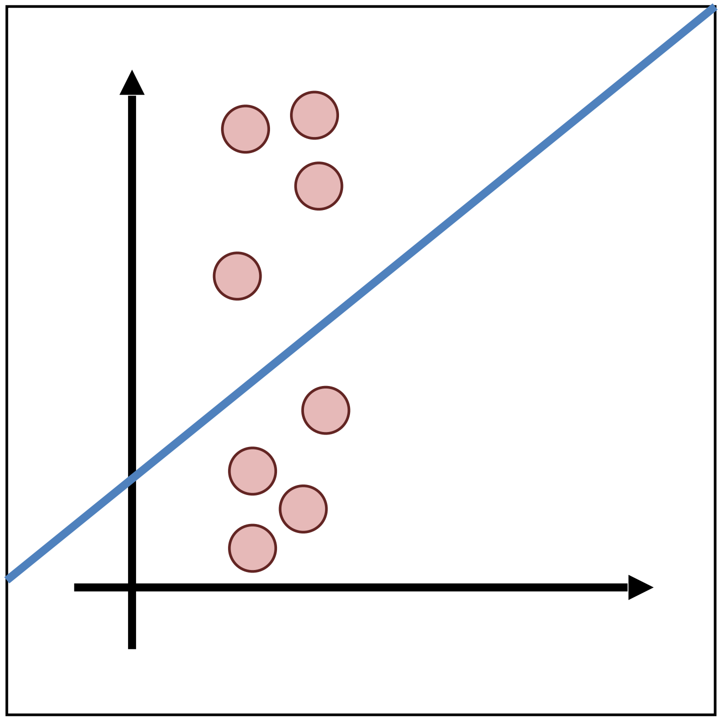


Least squares minimization



What's the problem?

- Vertical lines impossible!
- Not rotationally invariant: the line will change depending on the orientation of the points



Source: David Fouhey

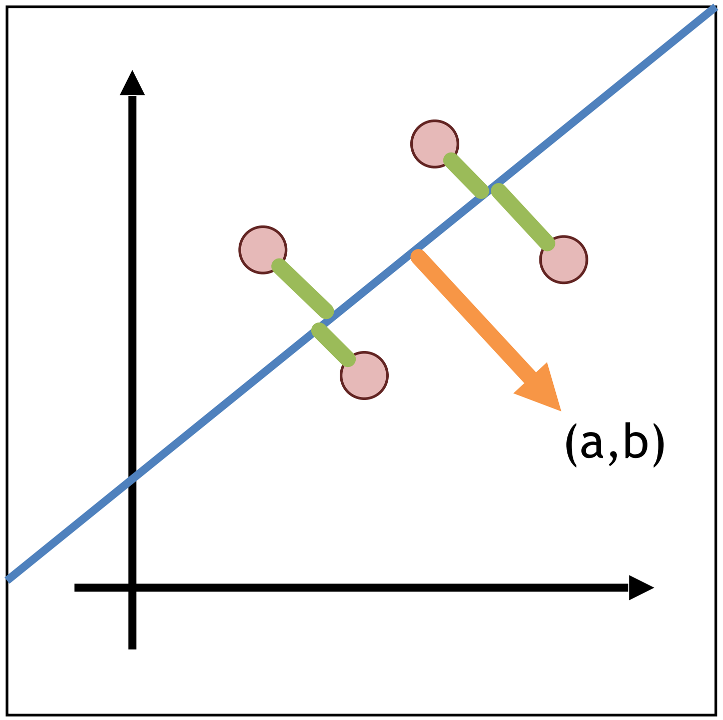
Alternative formulation: total least squares

- Fitting a line to data

Data: $(x_1,y_1), (x_2,y_2), \ldots, (x_k,y_k)$

Model: (a,b,d), $a^2+b^2=1$
$ax_i + by_i = d$

Objective function:
$(ax_i + by_i - d)^2$



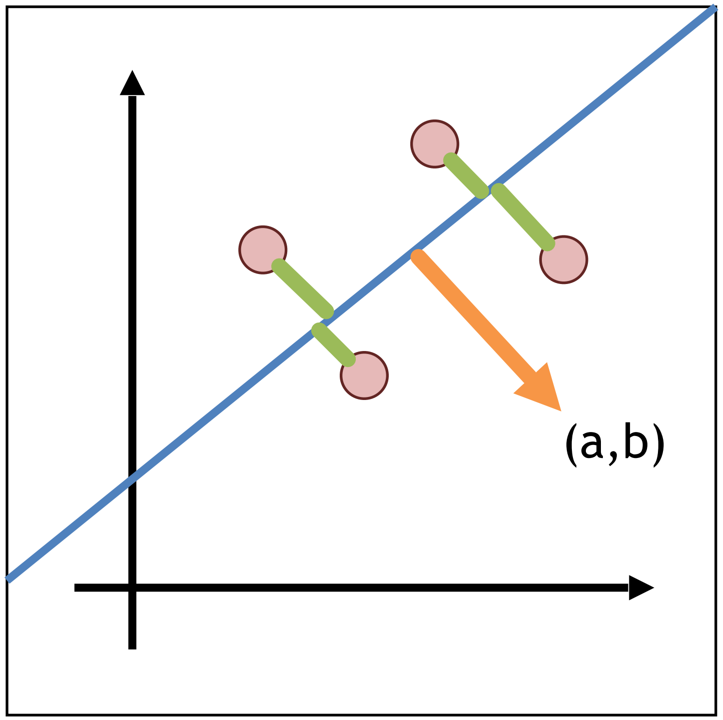
Source: David Fouhey

Alternative formulation: total least squares

- With a bit of algebra, can show this amounts 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



(derivation in the slides online)

Total least squares derivation, courtesy of Prof. David Fouhey

Total Least Squares Setup

Figure out objective first, then figure out ||n||=1

$$\sum_{i=1}^{k}\left(\boldsymbol{n}^T[x,y]-d\right)^2 \rightarrow \left\|\boldsymbol{X}\boldsymbol{n}-\mathbf{1}d\right\|_2^2$$

$$\boldsymbol{X}=\begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_k & y_k \end{bmatrix} \quad \mathbf{1}=\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad \boldsymbol{n}=\begin{bmatrix} a \\ b \end{bmatrix} \quad \boldsymbol{\mu}=\frac{1}{k}\mathbf{1}^T\boldsymbol{X}$$

$\uparrow$

The mean / center of mass of the points: we'll use it later

Source: David Fouhey

Solving Total Least-Squares

$$\left\|\boldsymbol{Xn} - \boldsymbol{1}d\right\|_2^2 = \left(\boldsymbol{Xn} - \boldsymbol{1}d\right)^T(\boldsymbol{Xn} - \boldsymbol{1}d)$$
$$= (\boldsymbol{Xn})^T(\boldsymbol{Xn}) - 2d\boldsymbol{1}^T\boldsymbol{Xn} + d^2\boldsymbol{1}^T\boldsymbol{1}$$

First solve for d at optimum (set to 0)

$$\frac{\partial}{\partial d}\left\|\boldsymbol{Xn} - \boldsymbol{1}d\right\|_2^2 = 0 - 2\boldsymbol{1}^T\boldsymbol{Xn} + 2dk$$

$$0 = -2\boldsymbol{1}^T\boldsymbol{Xn} + 2dk \Longrightarrow 0 = -\boldsymbol{1}^T\boldsymbol{Xn} + dk$$

$$\Longrightarrow d = \frac{1}{k}\boldsymbol{1}^T\boldsymbol{Xn} = \boldsymbol{\mu n}$$

Source: David Fouhey

Solving Total Least-Squares

$$\left\|\boldsymbol{X}\boldsymbol{n} - \mathbf{1}d\right\|_2^2 = \left\|\boldsymbol{X}\boldsymbol{n} - \mathbf{1}\boldsymbol{\mu}\boldsymbol{n}\right\|_2^2 \qquad d = \boldsymbol{\mu}\boldsymbol{n}$$

$$= \left\|\left(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu}\right)\boldsymbol{n}\right\|_2^2$$

Objective is then:

$$\arg\min_{||\boldsymbol{n}||=1} \left|\left|\left(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu}\right)\boldsymbol{n}\right|\right|_2^2$$

Source: David Fouhey

Homogeneous Least Squares

$$\underset{\|\boldsymbol{v}\|_2^2=1}{\arg\min} \|\boldsymbol{A}\boldsymbol{v}\|_2^2 \longrightarrow$$ Eigenvector corresponding to smallest eigenvalue of $A^T A$

Why do we need $||v||^2 = 1$ or some other constraint?

Applying it in our case:

$$\boldsymbol{n} = \text{smallest\_eigenvec}\left(\left(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu}\right)^T (\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu})\right)$$

Note: technically homogeneous only refers to $||Av||=0$ but it's common shorthand in computer vision to refer to the specific problem of $||v||=1$

Source: David Fouhey

Details For ML-People

Matrix we take the eigenvector of looks like:

$$\left(\boldsymbol{X}-\mathbf{1}\boldsymbol{\mu}\right)^{\boldsymbol{T}}(\boldsymbol{X}-\mathbf{1}\boldsymbol{\mu}) = \begin{bmatrix} \sum_i \left(x_i-\mu_x\right)^2 & \sum_i \left(x_i-\mu_x\right)\left(y_i-\mu_y\right) \\ \sum_i \left(x_i-\mu_x\right)\left(y_i-\mu_y\right) & \sum_i \left(y_i-\mu_y\right)^2 \end{bmatrix}$$

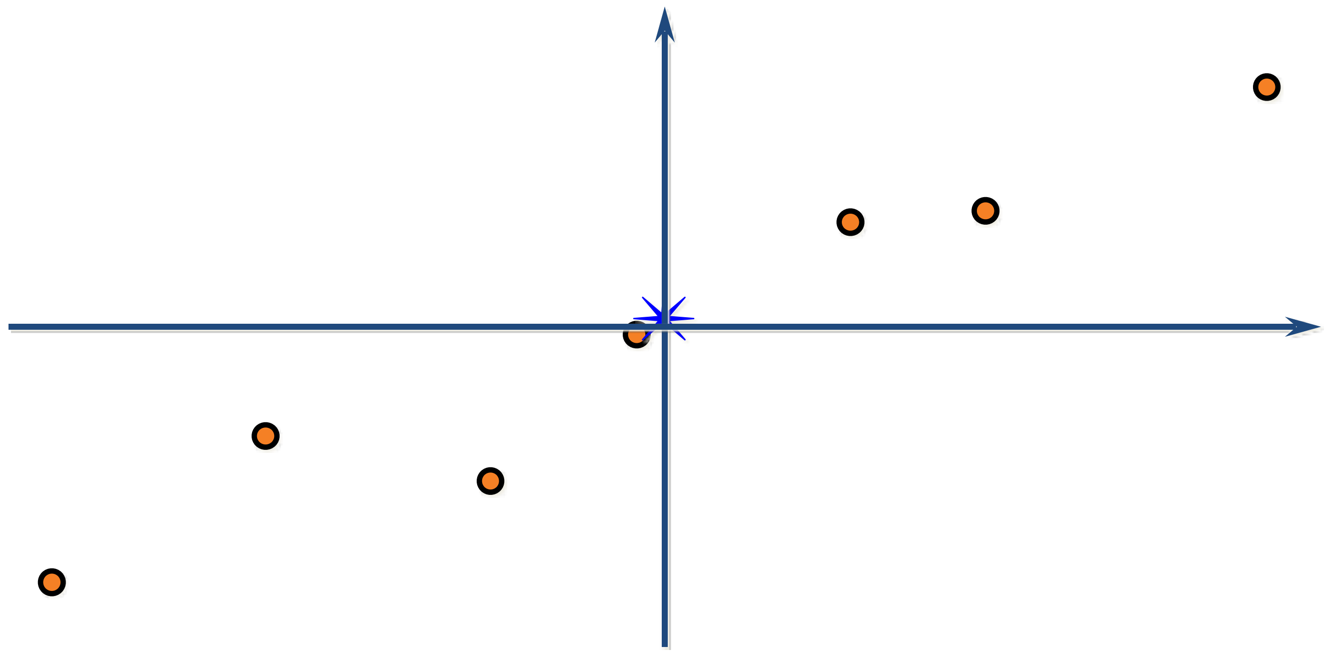
This is a scatter matrix or scalar multiple of the covariance matrix. We're doing PCA, but taking the least principal component to get the normal.

Note: If you don't know PCA, just ignore this slide; it's to help build connections to people with a background in data science/ML.

Source: David Fouhey

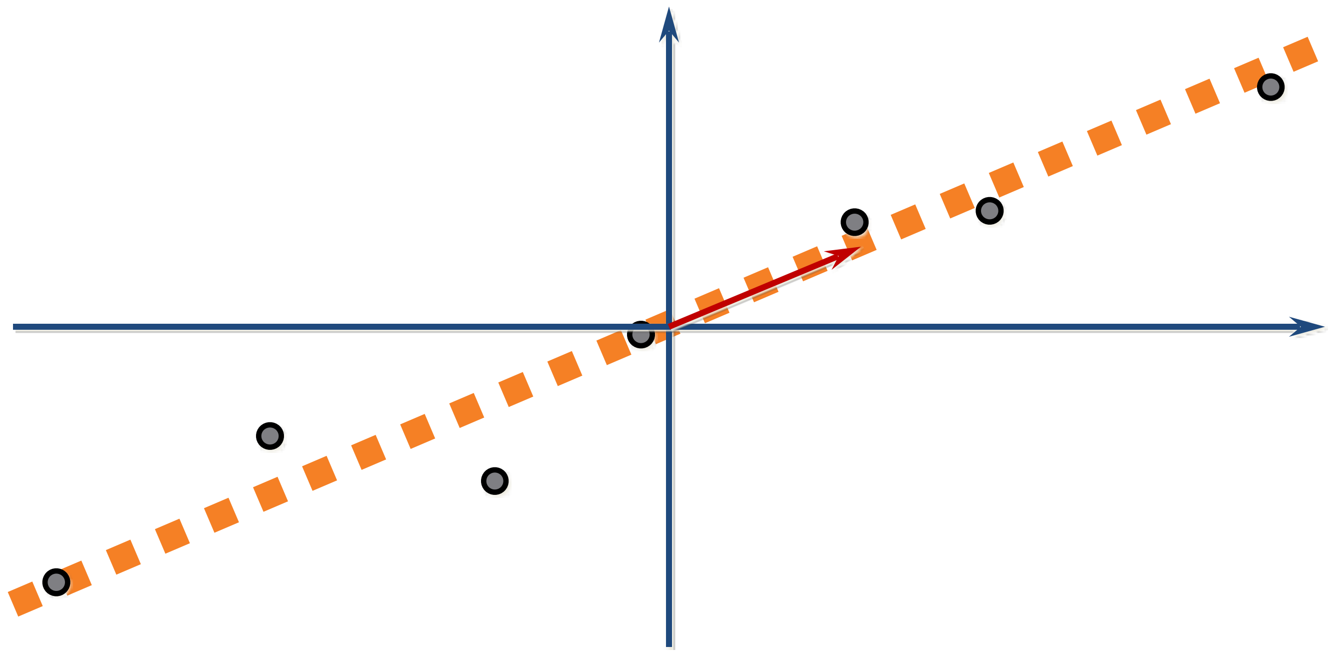
Total Least Squares

1. Translate center of mass to origin

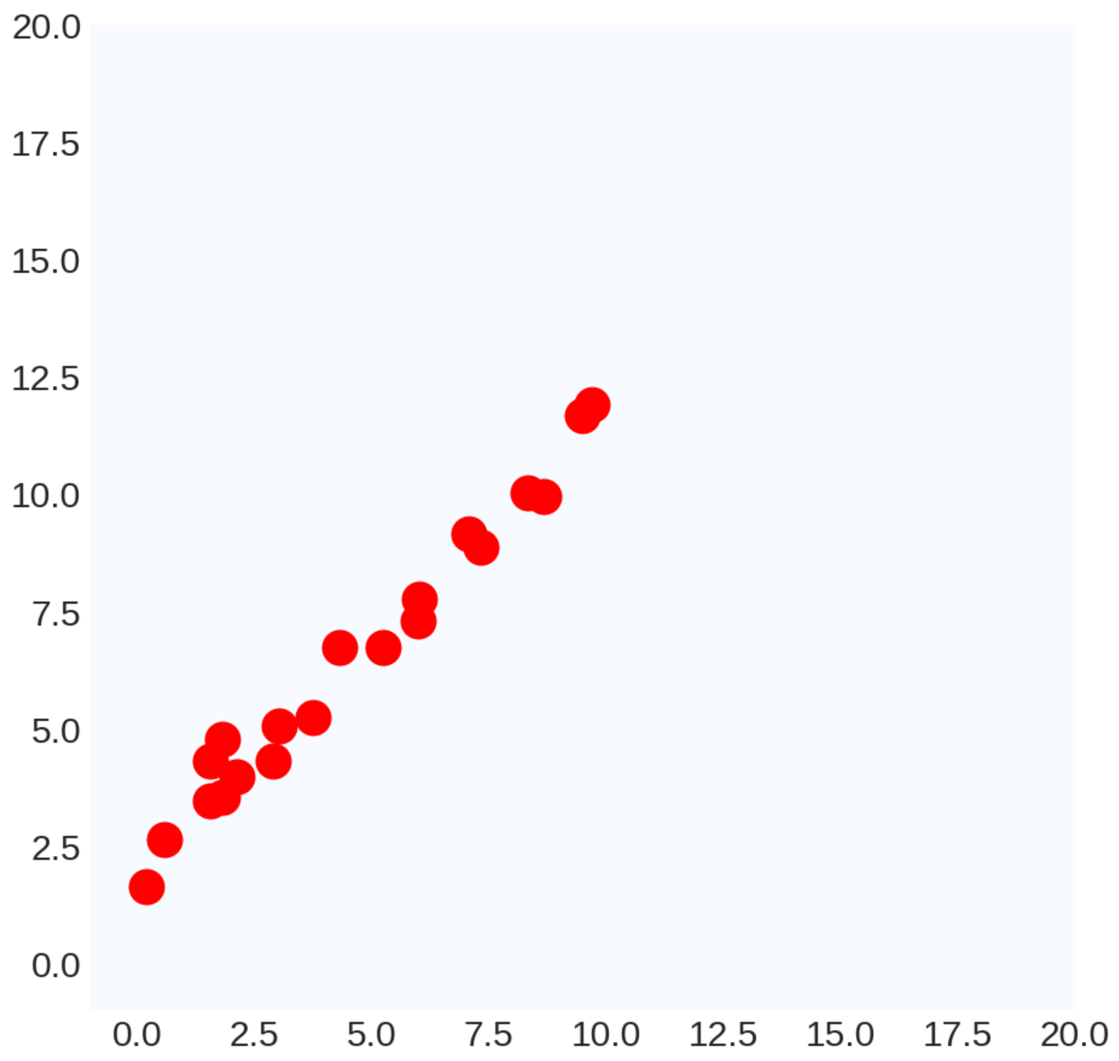


Total Least Squares

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

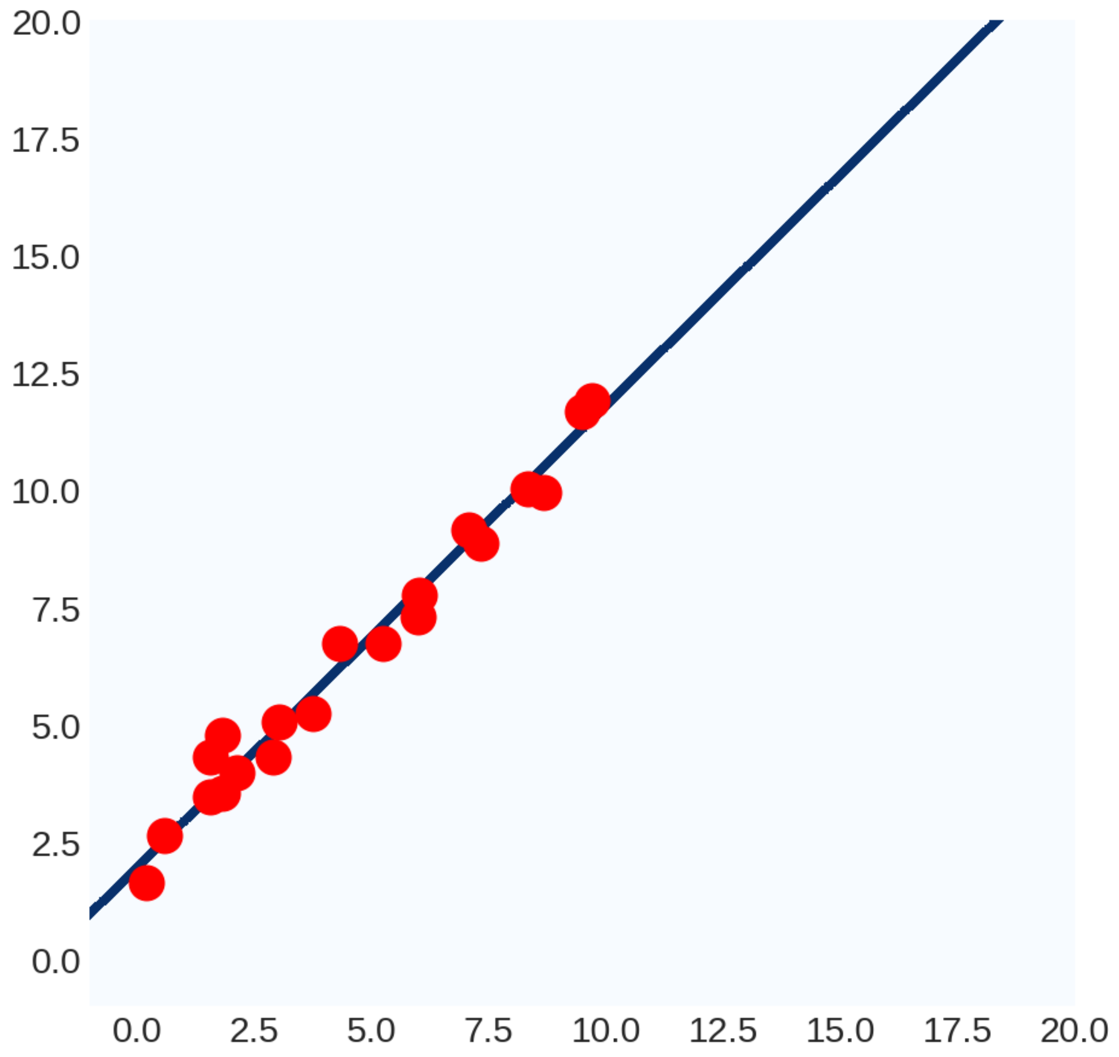


Running Least Squares



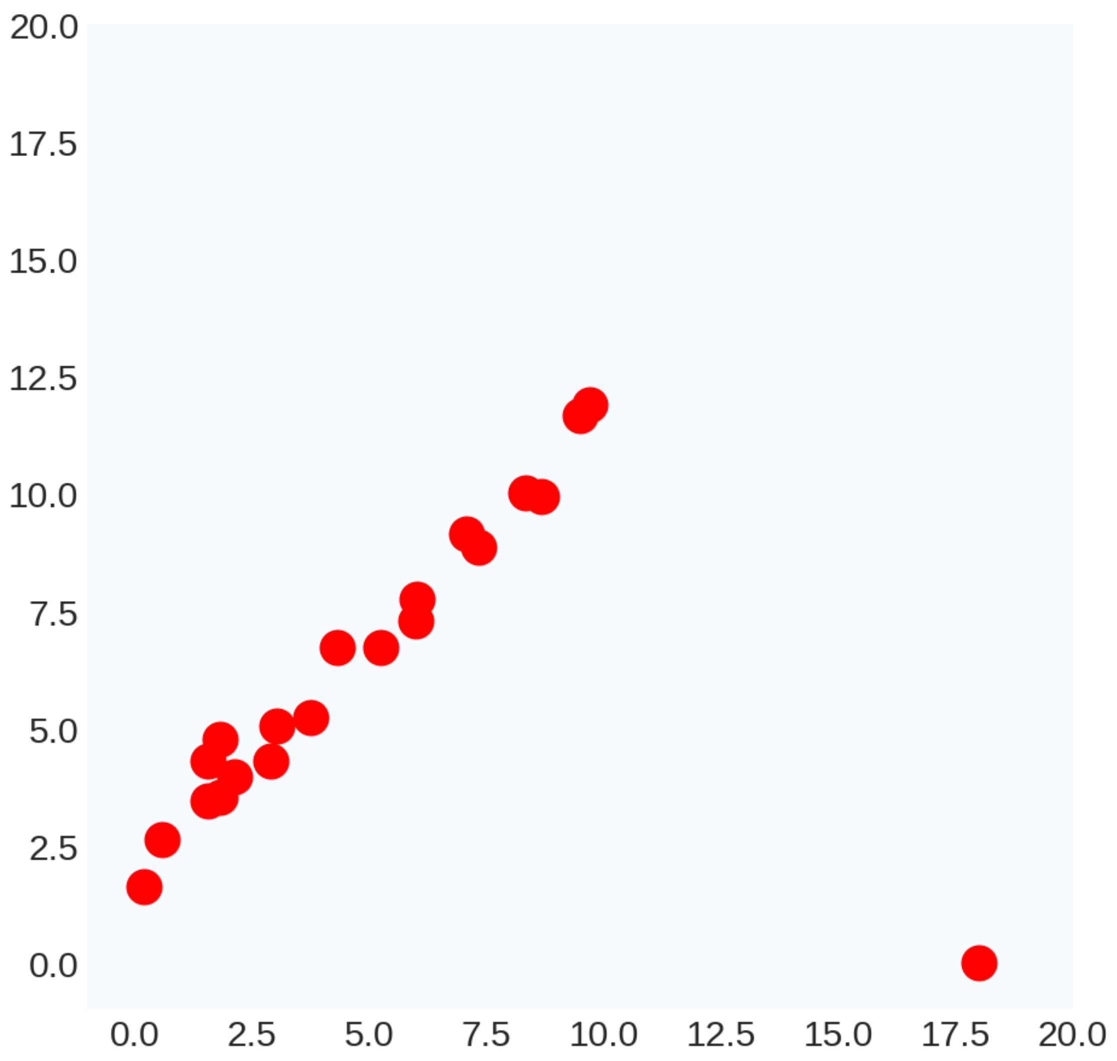
Source: D. Fouhey

Running Least Squares



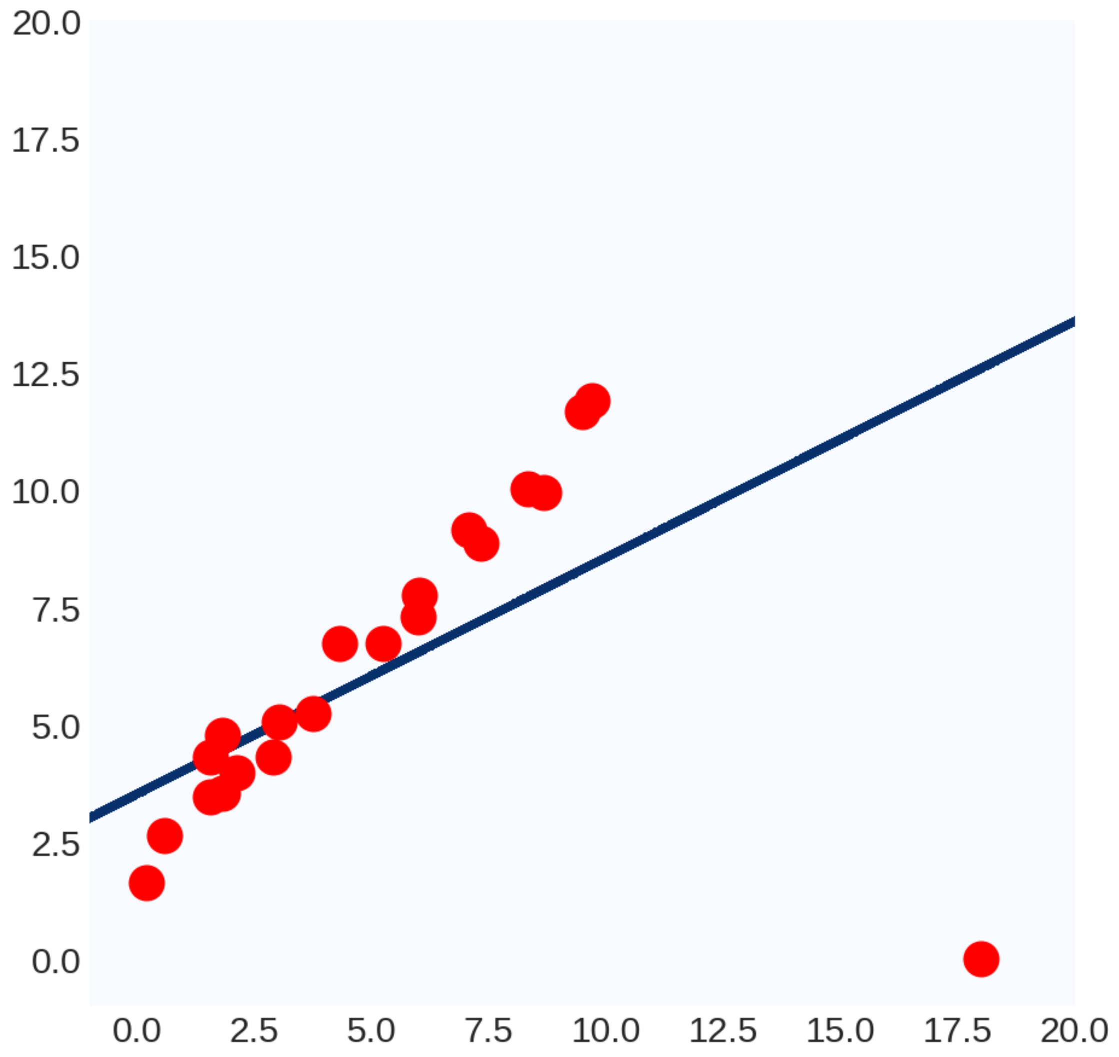
Source: D. Fouhey

Running Least Squares



Source: D. Fouhey

Running Least Squares



Source: D. Fouhey

Running Least Squares

- Way to think of this #1:
  - $||Y - XW||_2^2$
  - $100^2 >> 10^2$, least-squares model prefers having no large errors, even if the model is overall useless
- Way to think of this #2:
  - $W = (X^T X)^{-1} X^T Y$
  - Weights are a linear transformation of the output variable: can manipulate W by manipulating Y
- Way to think of this #3:
  - Least squares assumes Gaussian noise
  - Outliers: points with extremely low probability of occurrence (according to Gaussian statistics), thus 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 $\sum_i |y_i - f(x_i, a, b, \dots)|$ *(median)*

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

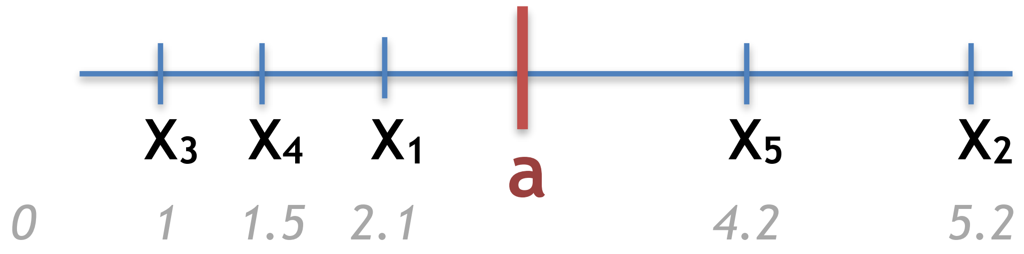
- Points far away from trend get comparatively less influence

Aside: why is minimizing sum of squares = mean?

- Goal: $\min_a \sum_i (a - x_i)^2$
- Algebraic solution:

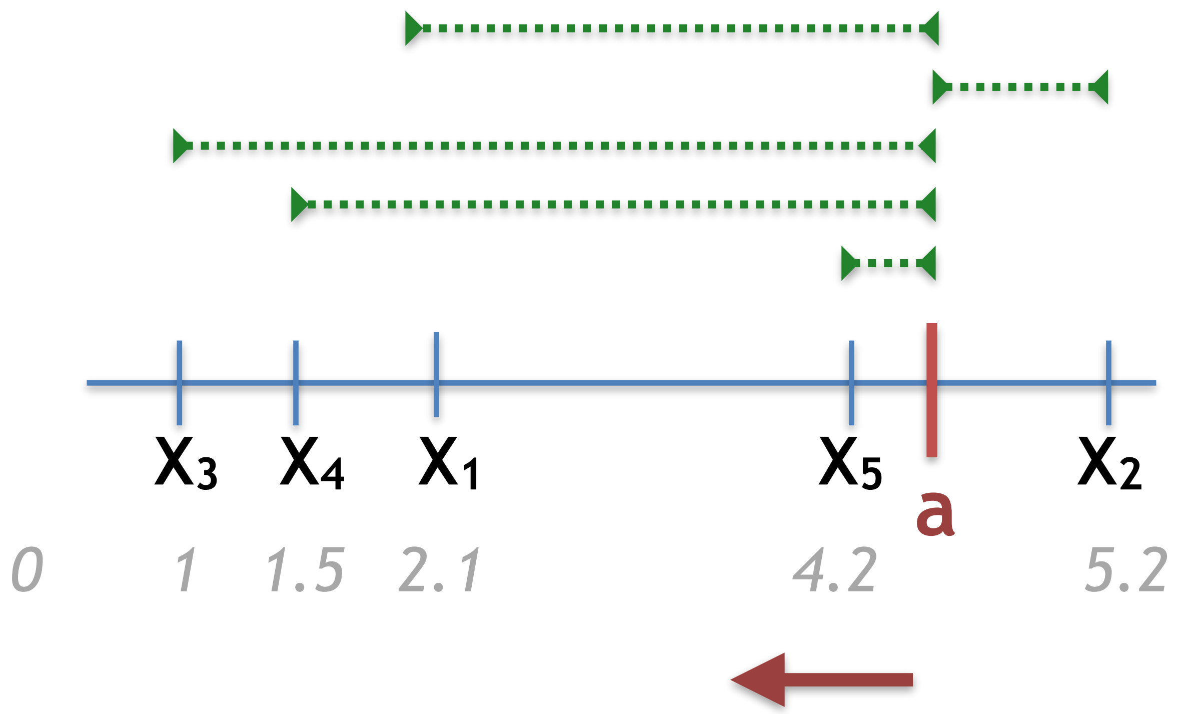
$$\frac{d}{da}\sum_{i=1}^{n}(a - x_i)^2 = 0 \quad \Longrightarrow \sum_{i=1}^{n} 2(a - x_i) = 0$$

$$\Longrightarrow \; 2an - 2\sum_{i=1}^{n} x_i = 0 \quad \Longrightarrow \; a = \frac{1}{n}\sum_{i=1}^{n} x_i$$



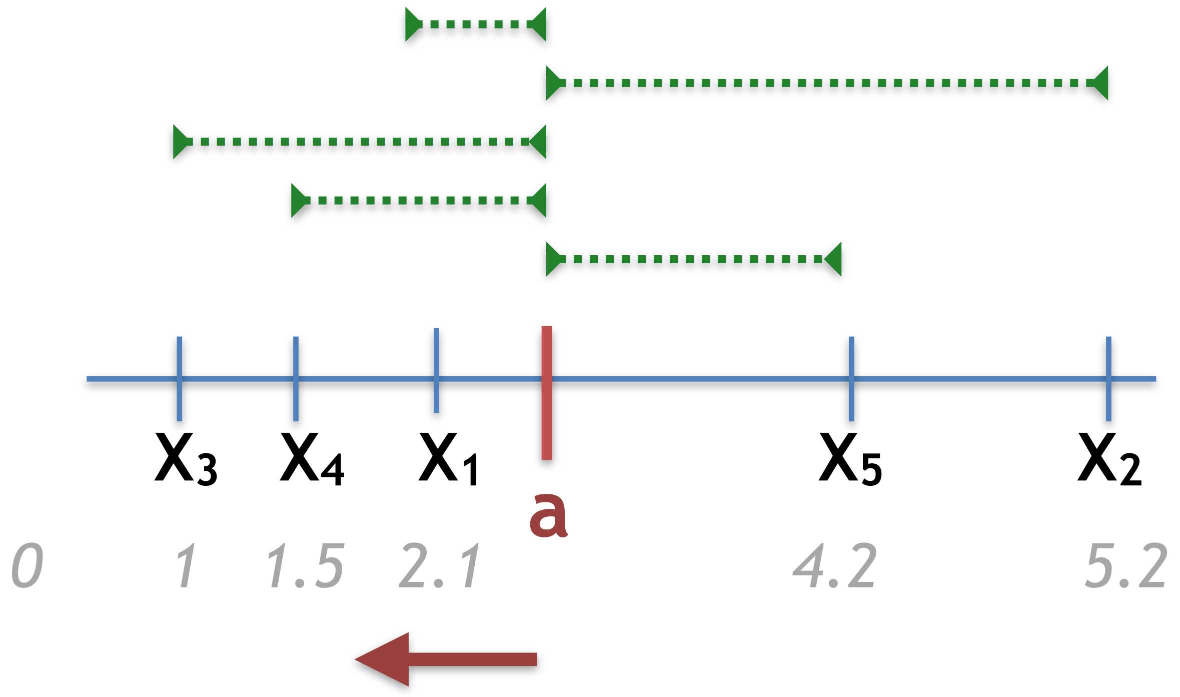
Aside: why is min. absolute distances = median?

- Goal: $\min_a \sum_i |a - x_i|$
- Geometric intuition:



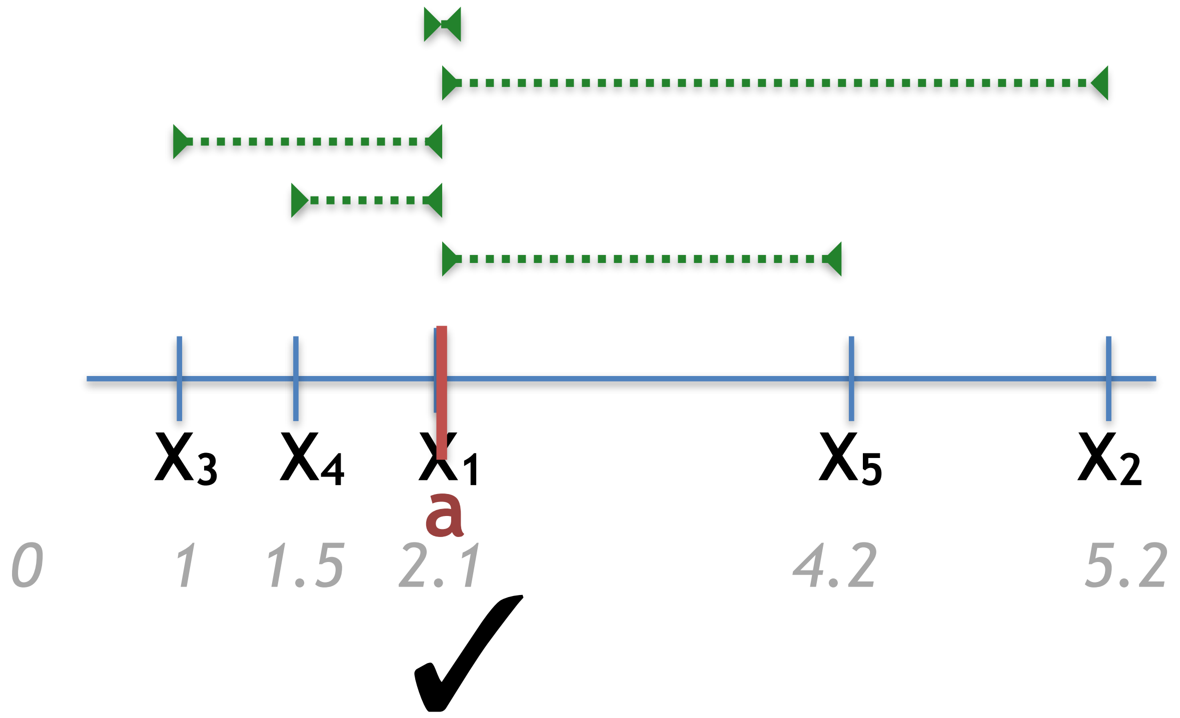
Aside: why is min. absolute distances = median?

- Goal: $\min_a \sum_i |a - x_i|$
- Geometric intuition:



Aside: why is min. absolute distances = median?

- Goal: $\min_{a} \sum_{i} |a - x_i|$
- Geometric intuition:



Outlier detection and rejection

- Lots of methods for fitting models in the presence of outliers
  - Median (L1) more robust than the mean (L2), but harder to optimize
  - Look up "iteratively reweighed least squares"
- Often not guaranteed to converge; require good starting point
  - (least squares estimator is often a good starting point)

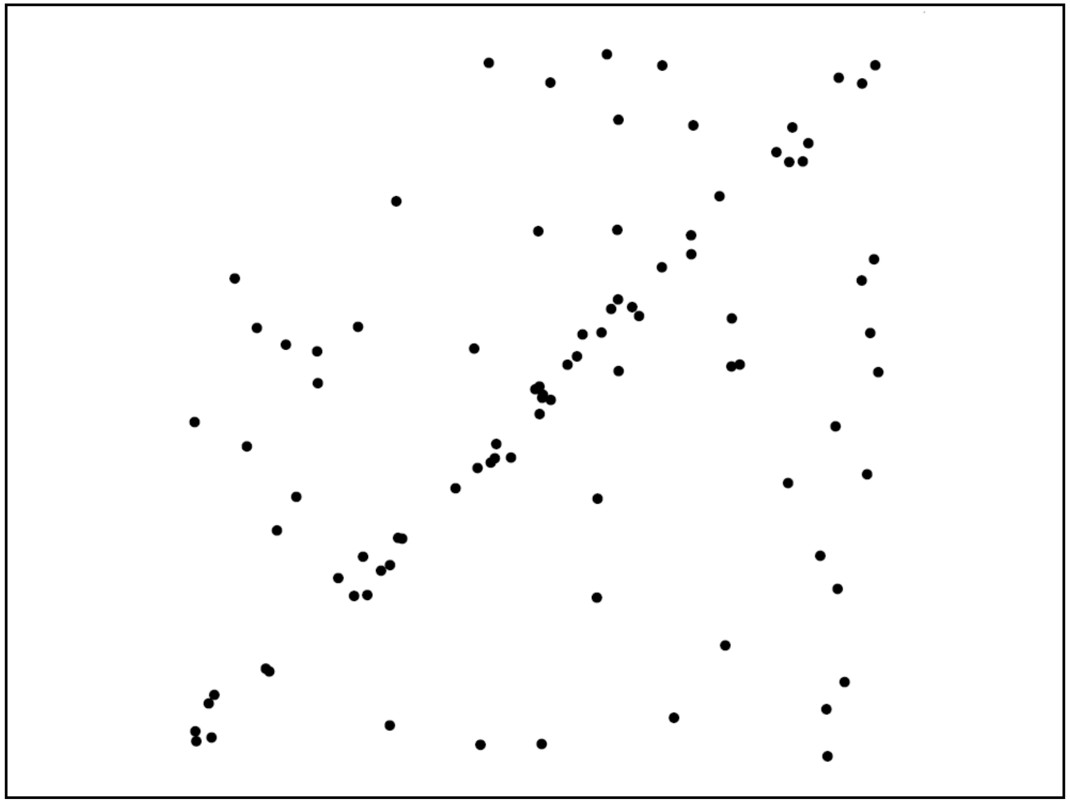
RANSAC

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

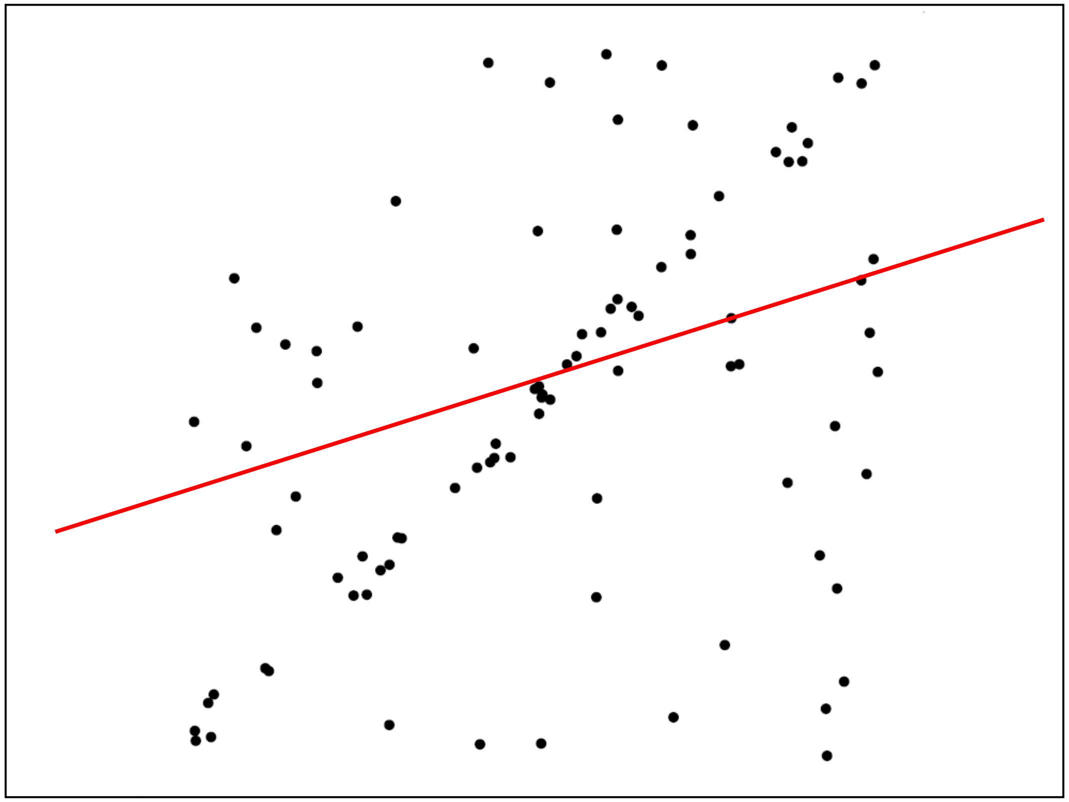
M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

RANSAC for line fitting example



Source: R. Raguram

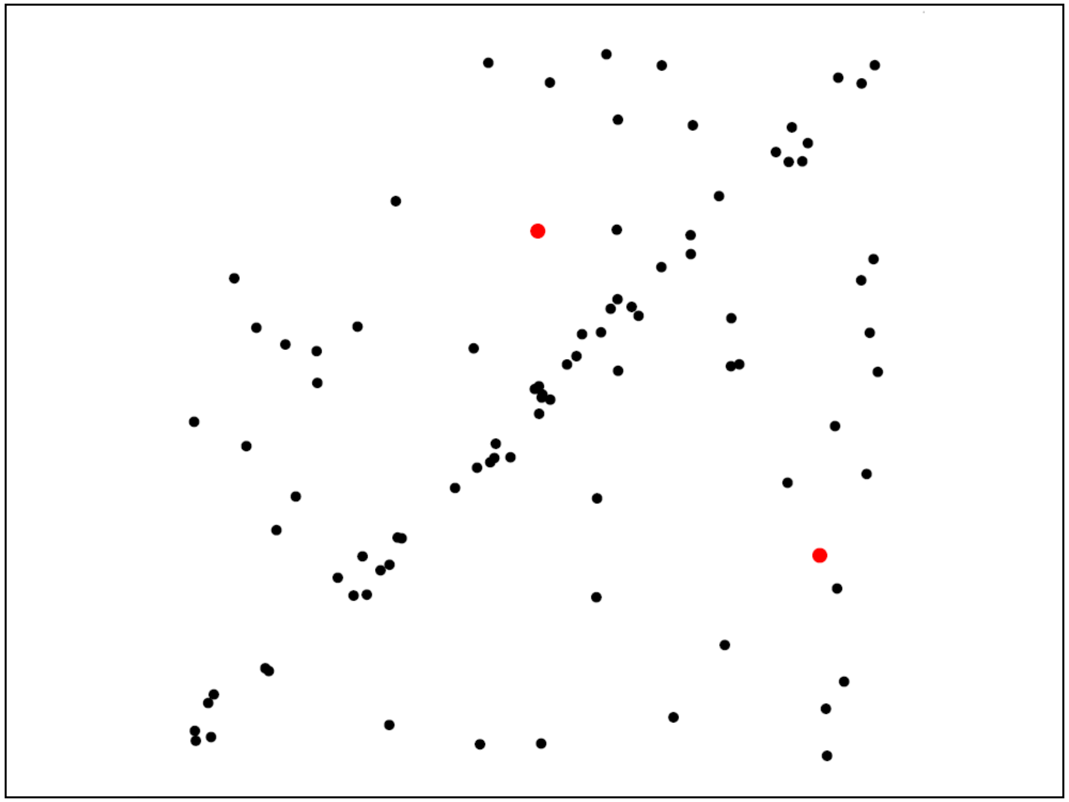
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

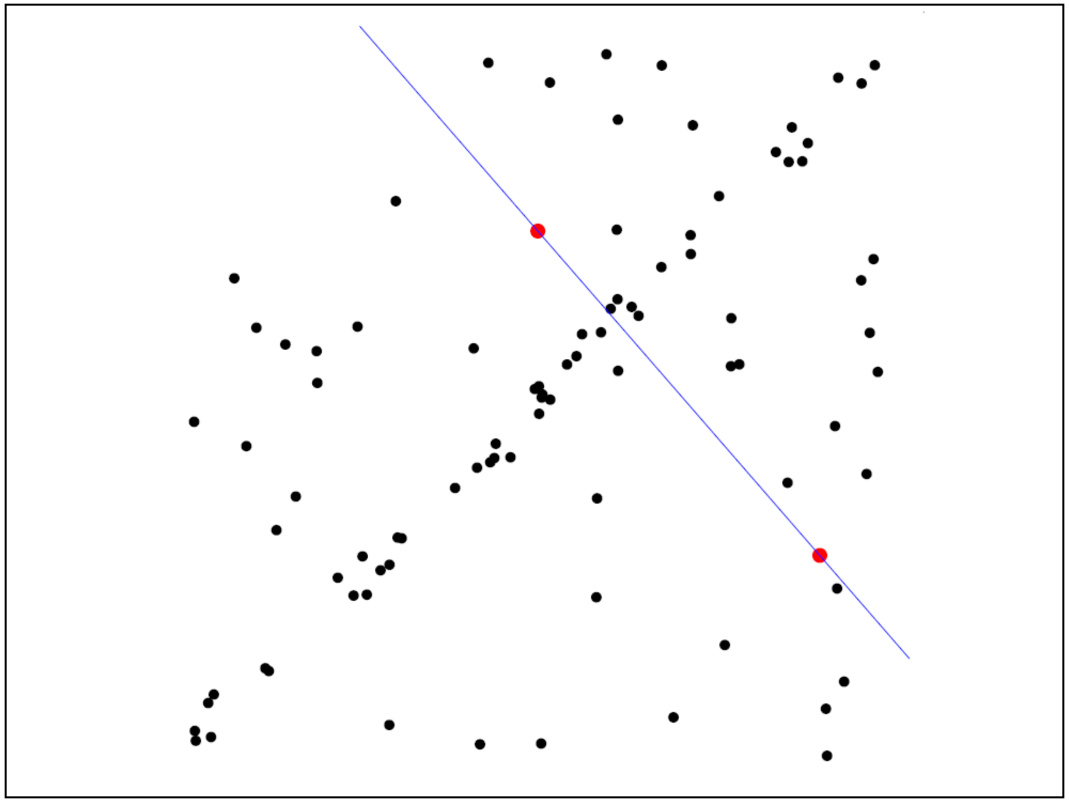
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

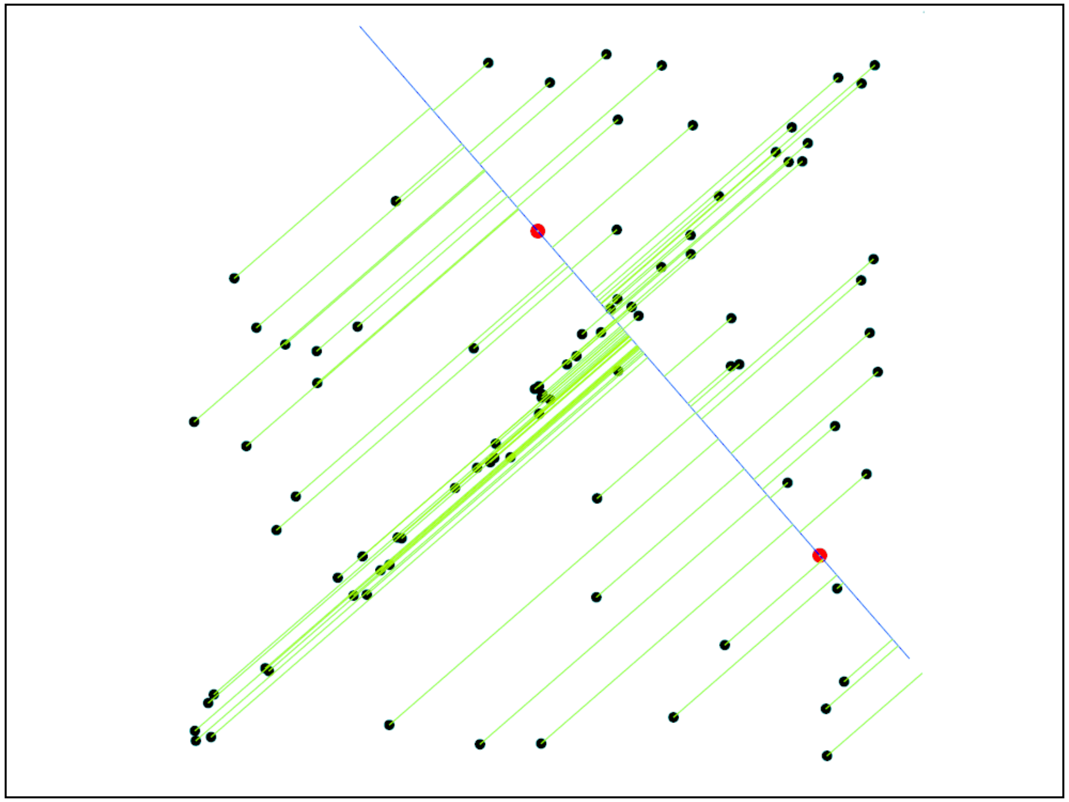
RANSAC for line fitting example



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

Source: R. Raguram

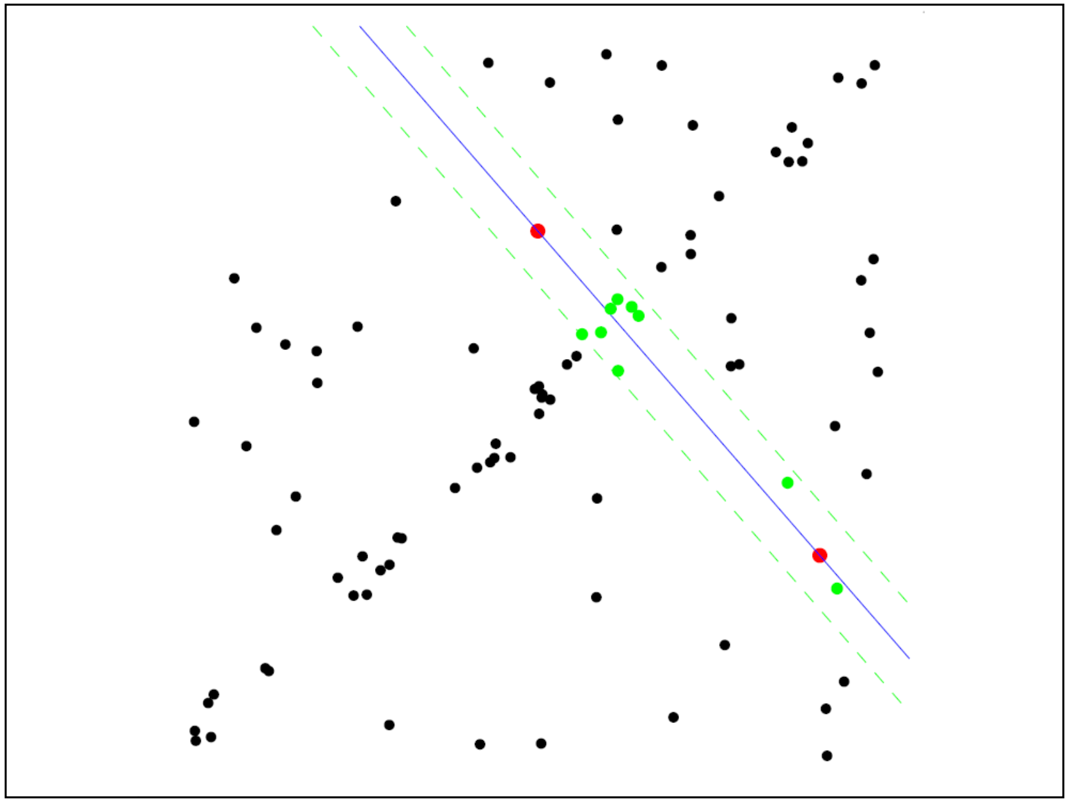
RANSAC for line fitting example



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

Source: R. Raguram

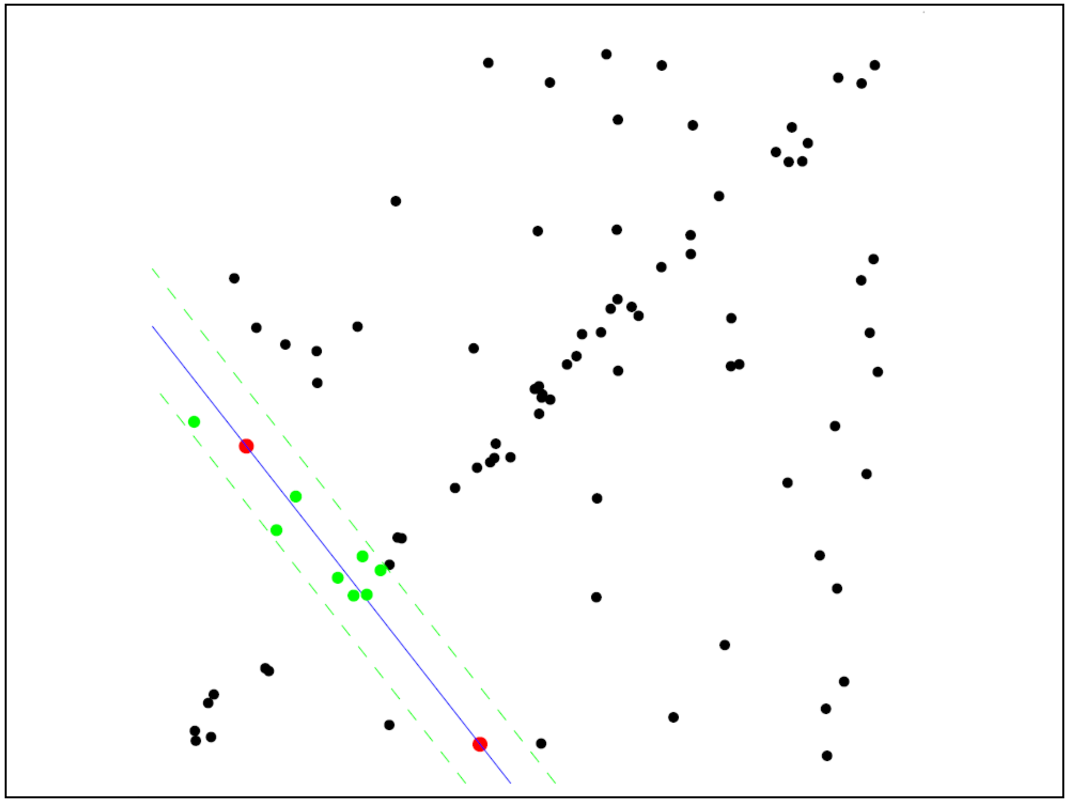
RANSAC for line fitting example



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

Source: R. Raguram

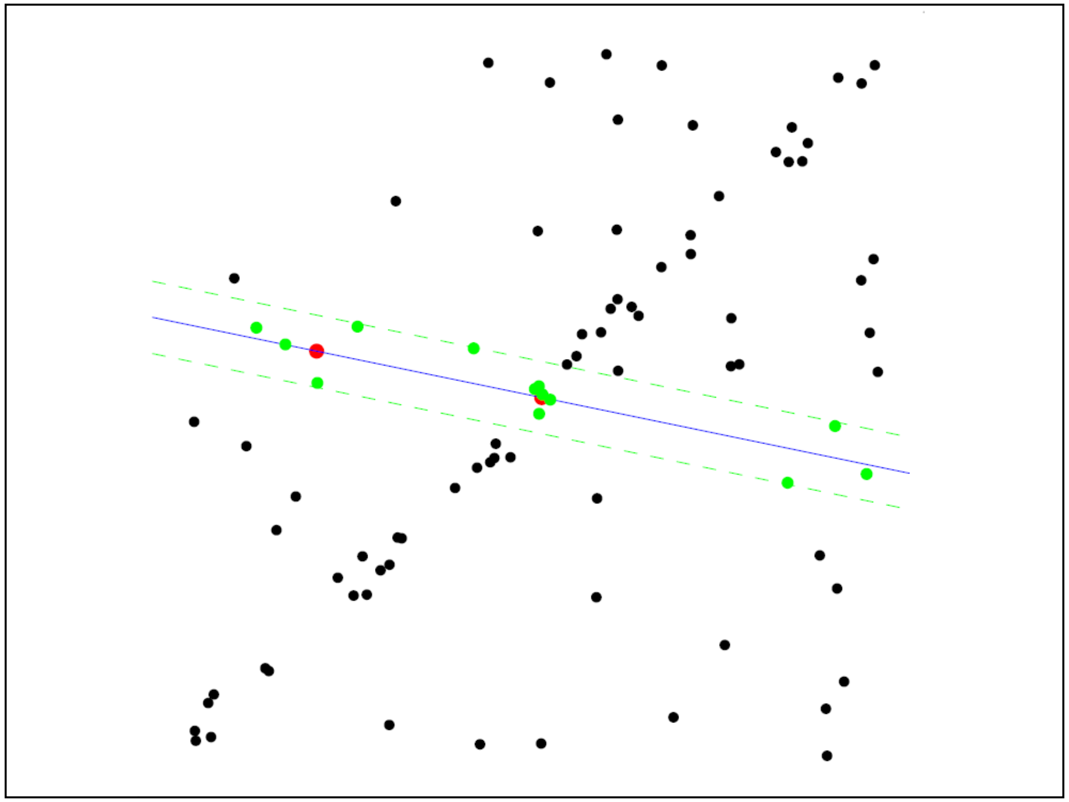
RANSAC for line fitting example



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

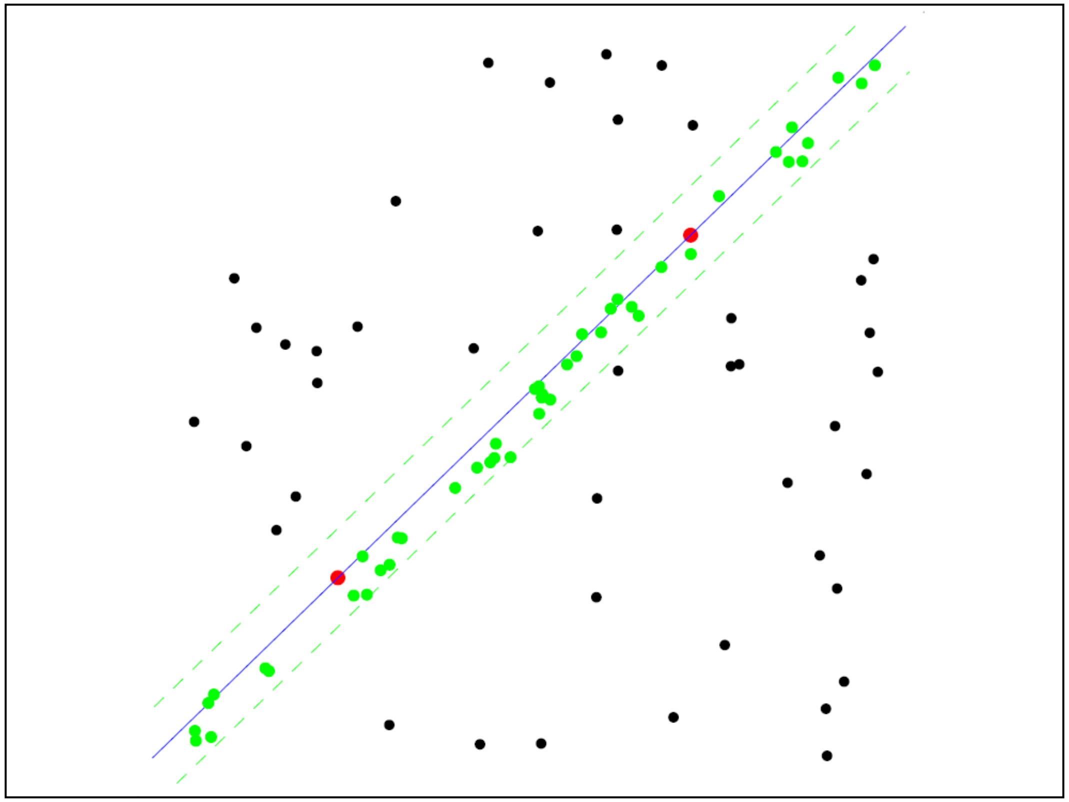


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

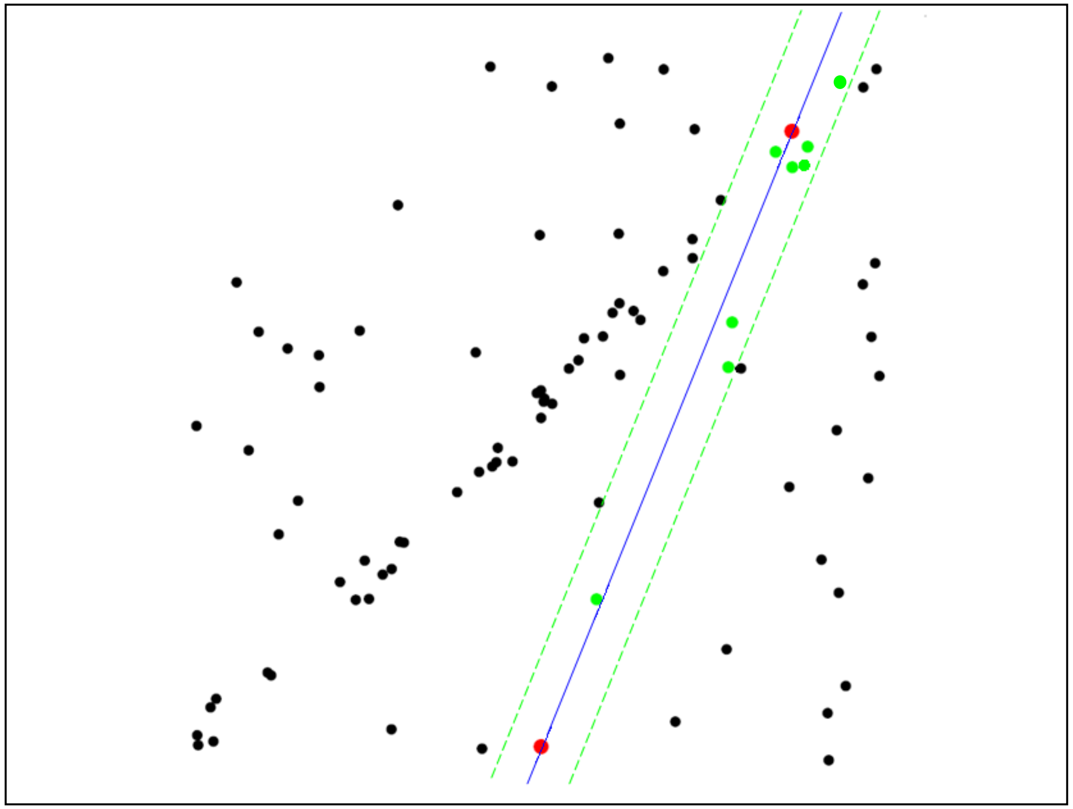
Uncontaminated sample



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

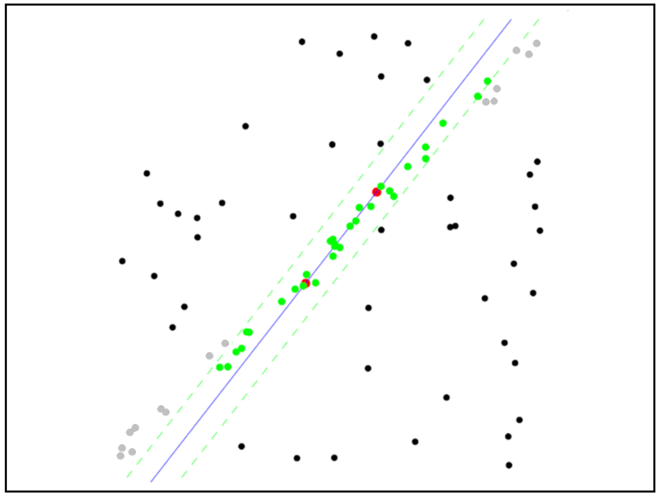
Choosing the parameters

- 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

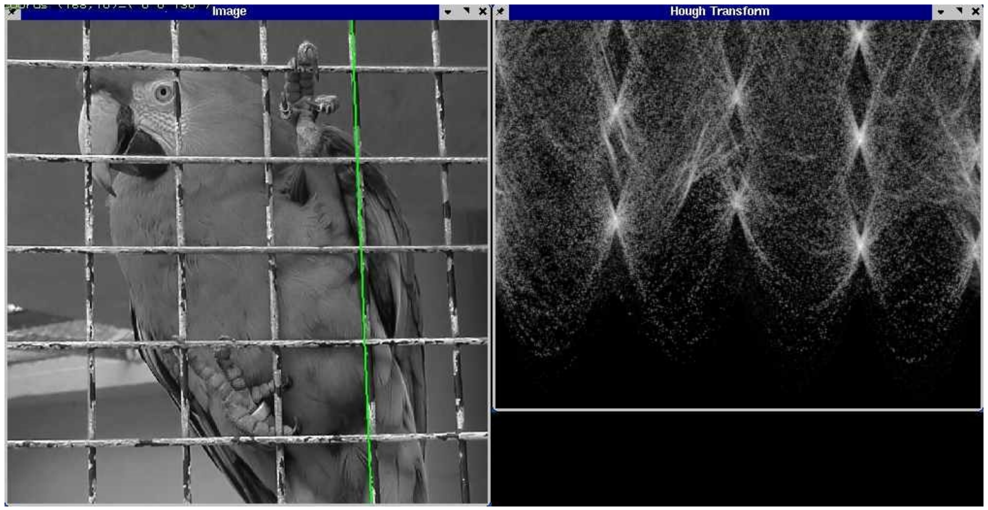
RANSAC pros and cons

- 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



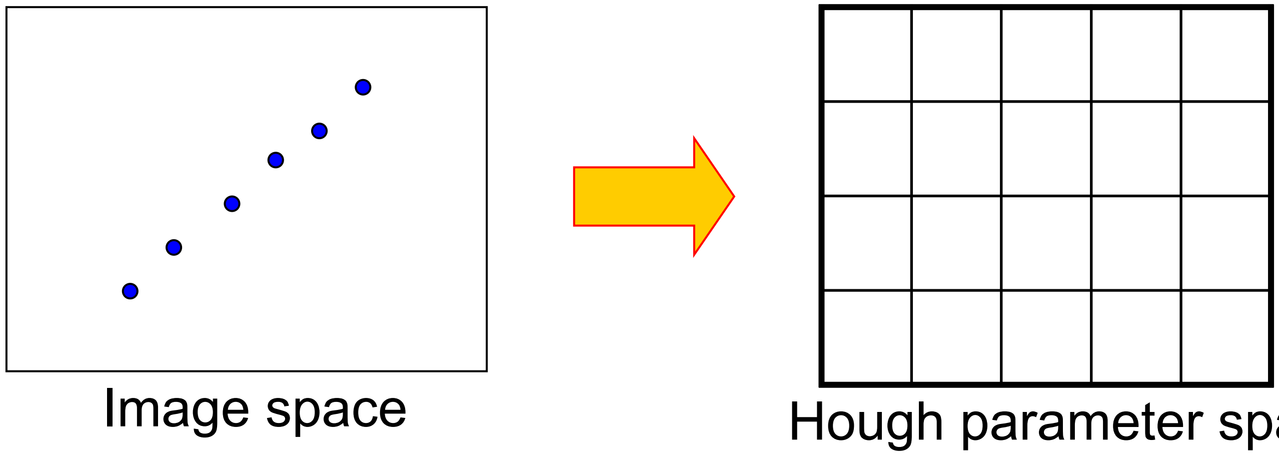
Voting schemes

- 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

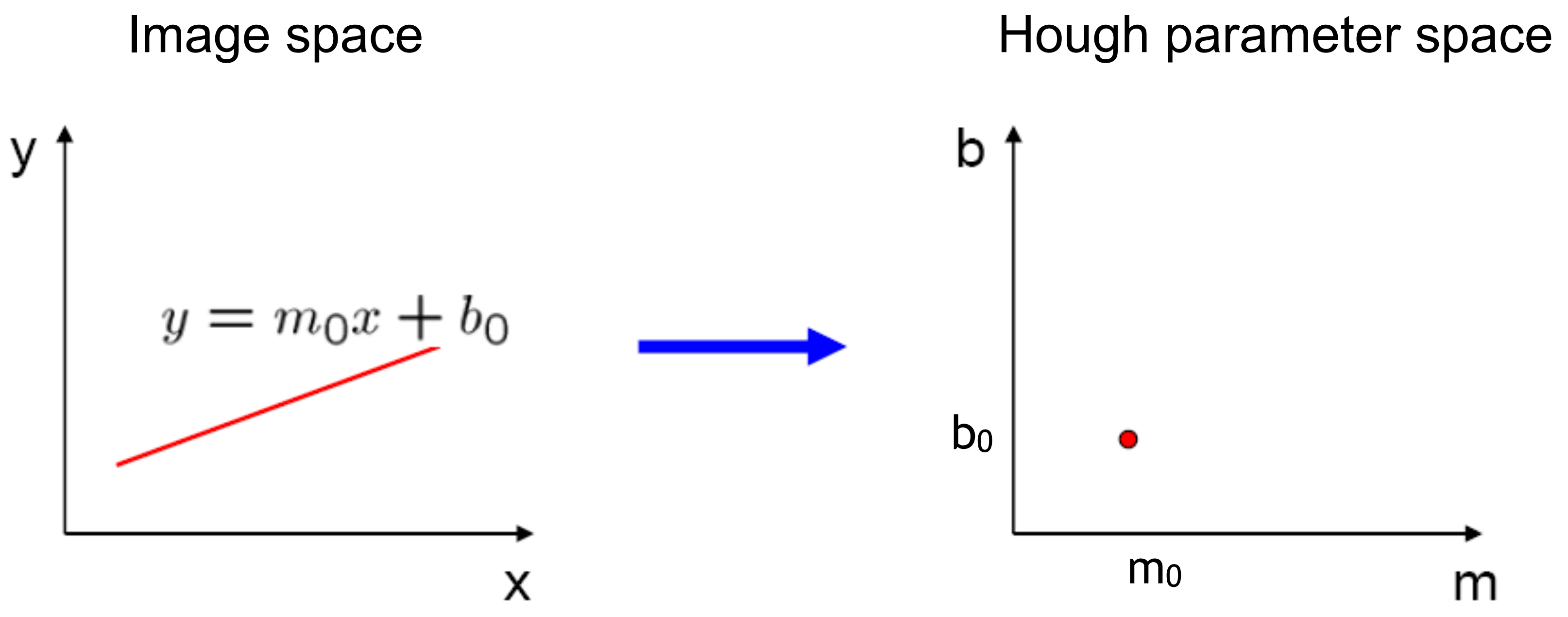


P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Source: S. Lazebnik

Parameter space representation

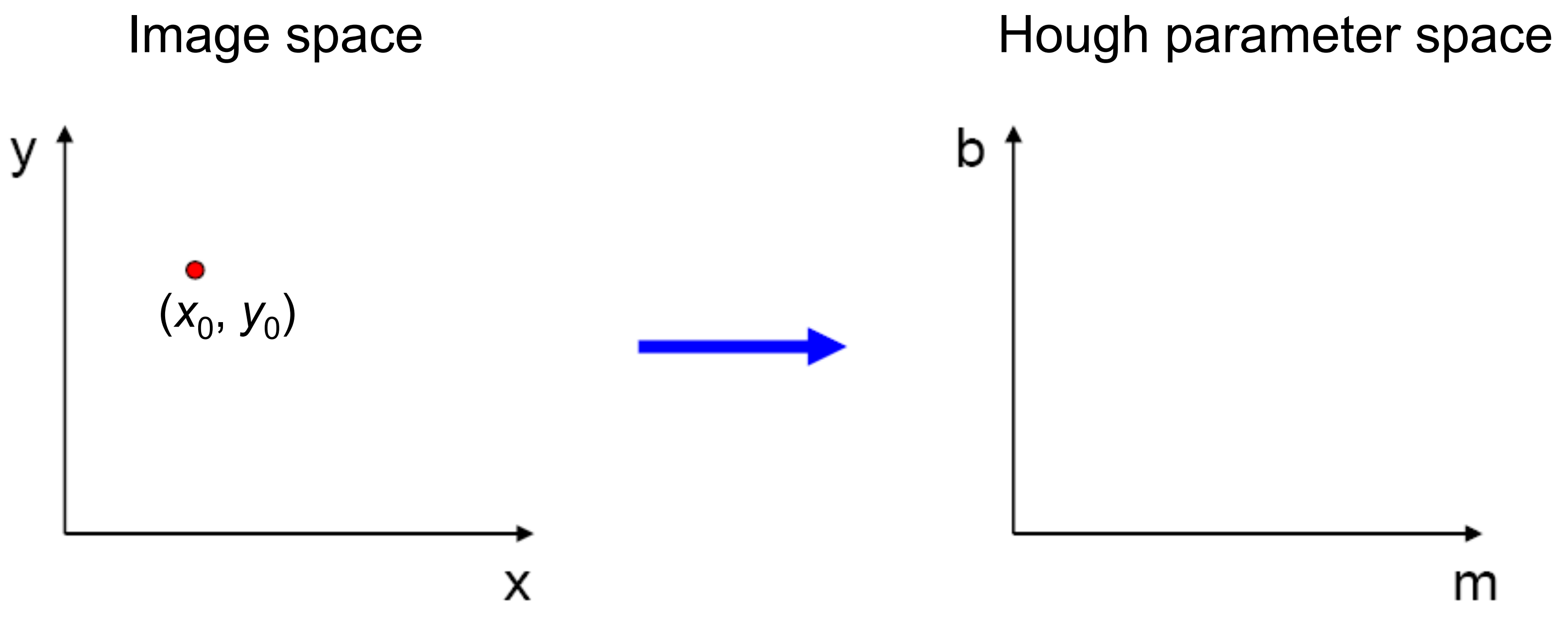
- A line in the image corresponds to a point in Hough space



Source: S. Seitz

Parameter space representation

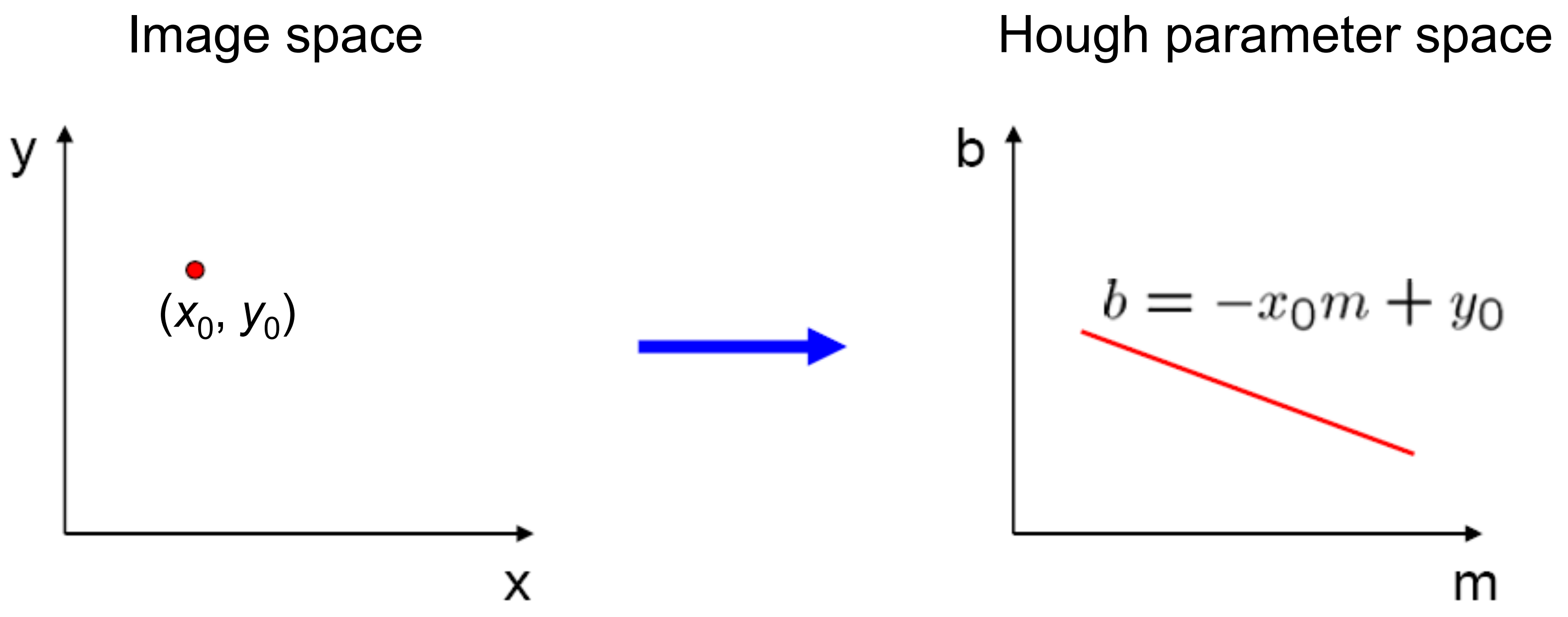
- What does a point (x_0, y_0) in the image space map to in the Hough space?



Source: S. Seitz

Parameter space representation

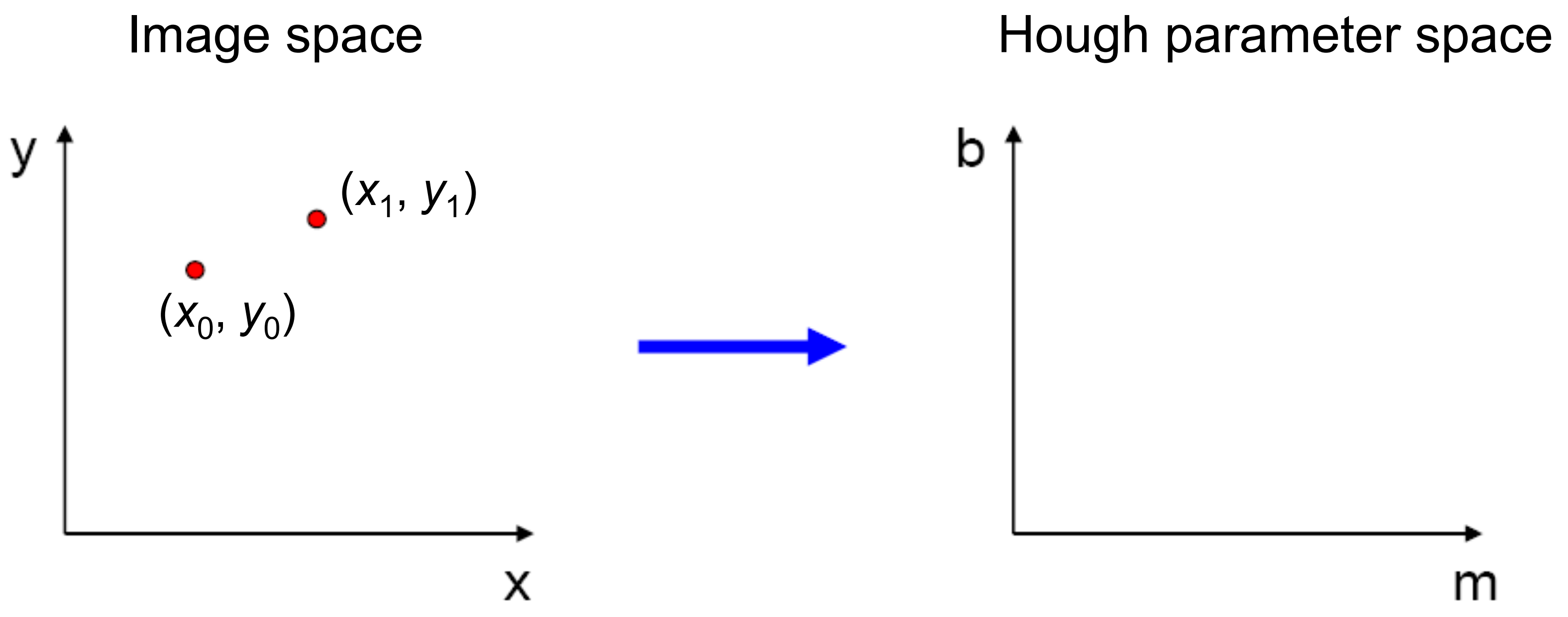
- What does a point (x_0, y_0) in the image space map to in the Hough space?
  - Answer: the solutions of $b = -x_0 m + y_0$, which is a line in Hough space



Source: S. Seitz

Parameter space representation

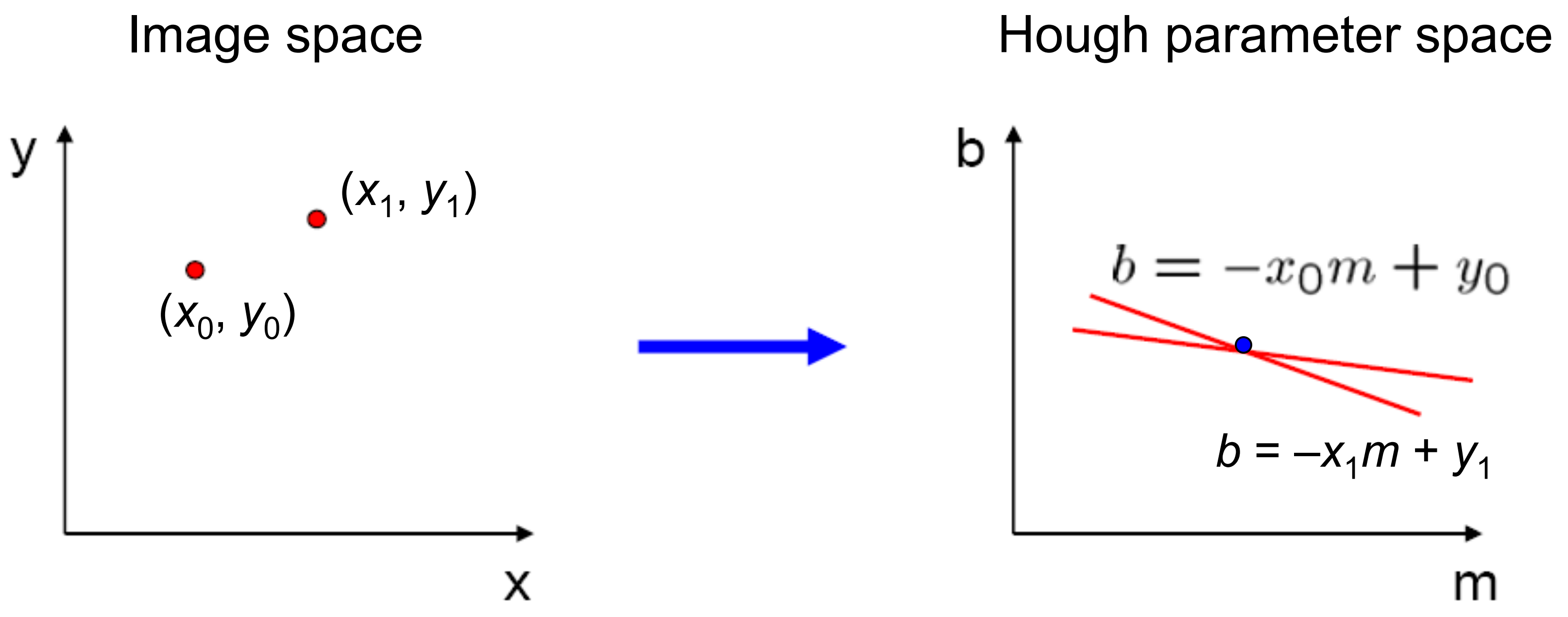
- Where does the line that contains both (x_0, y_0) and (x_1, y_1) map to?



Source: S. Seitz

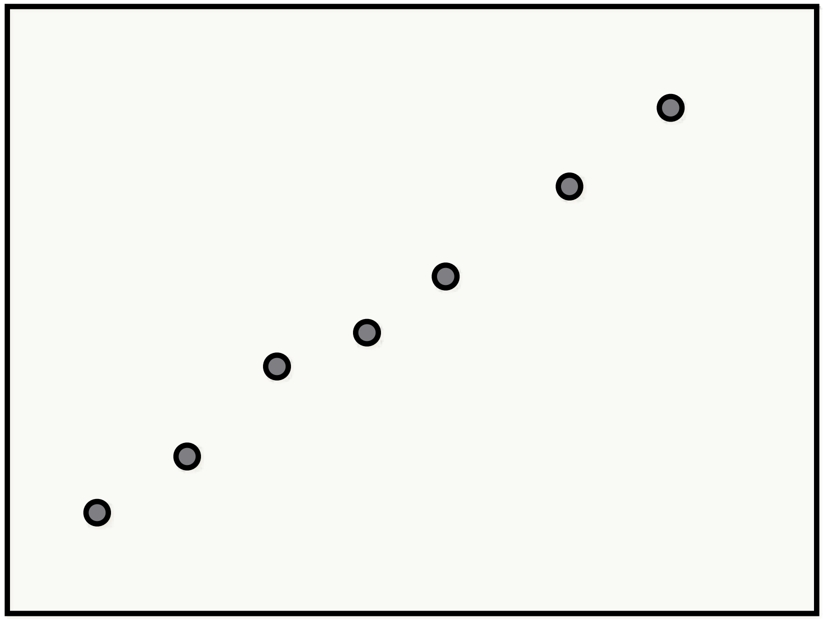
Parameter space representation

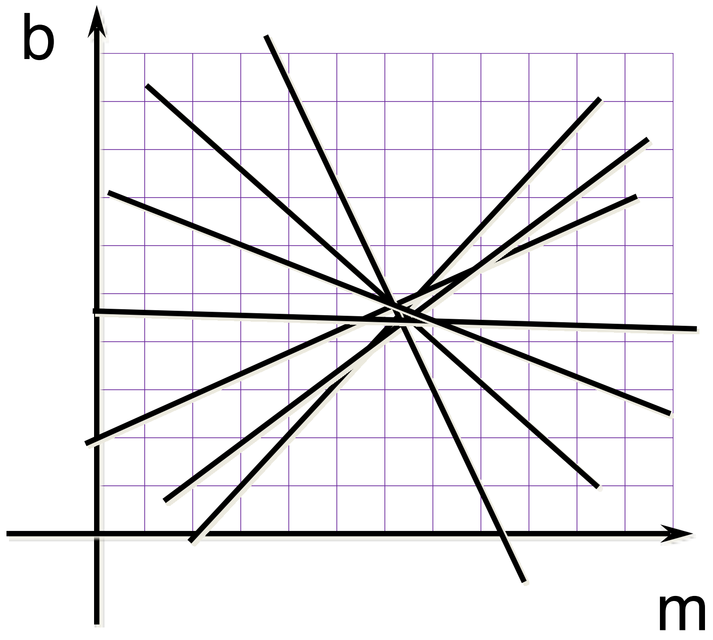
- Where does the line that contains both (x_0, y_0) and (x_1, y_1) map to?
  - It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$



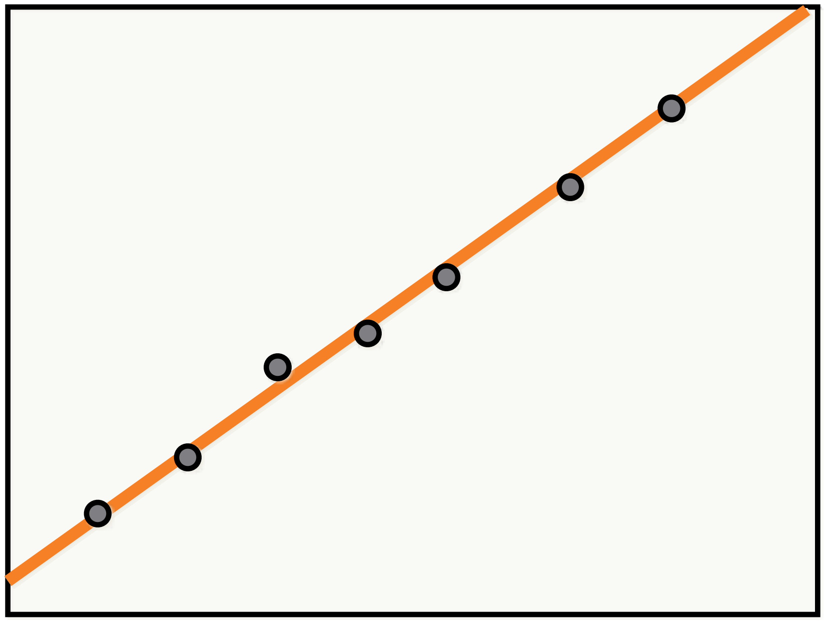
Source: S. Seitz

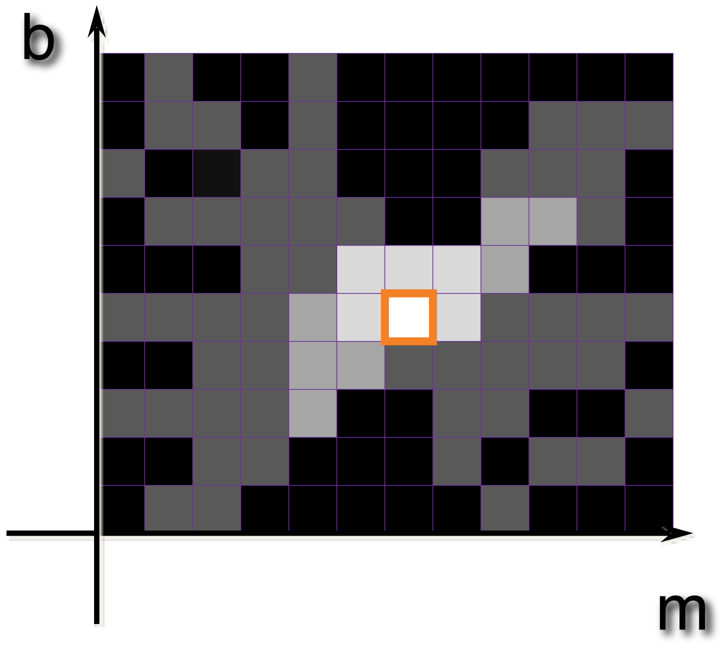
Hough Transform for Lines





Hough Transform for Lines





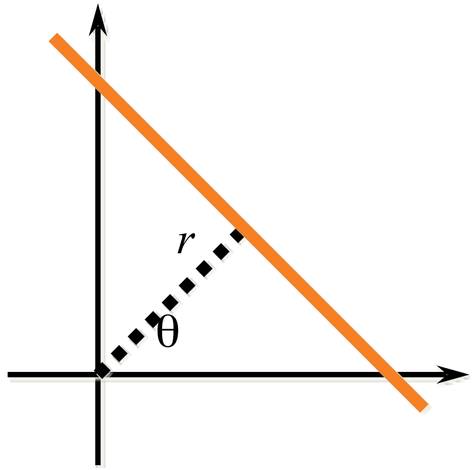
Bucket Selection

- 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

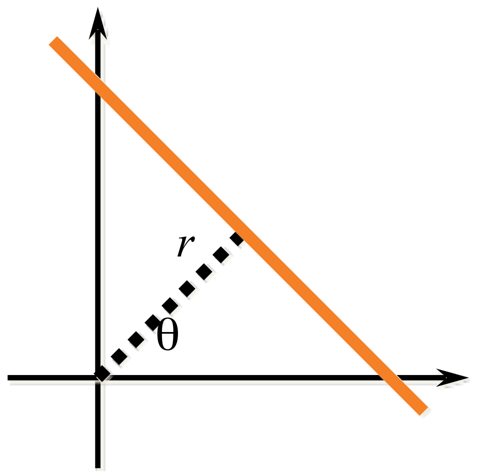
- Slope / intercept parameterization not ideal
  - Non-uniform sampling of directions
  - Can’t represent vertical lines
- Angle / distance parameterization
  - Line represented as (r, θ) where

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

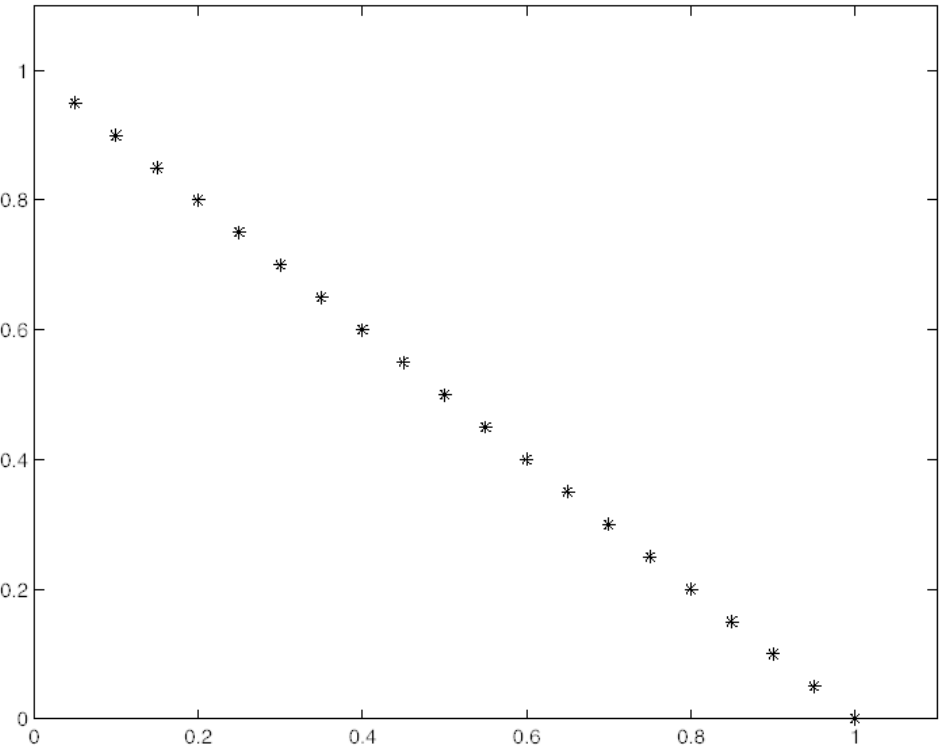


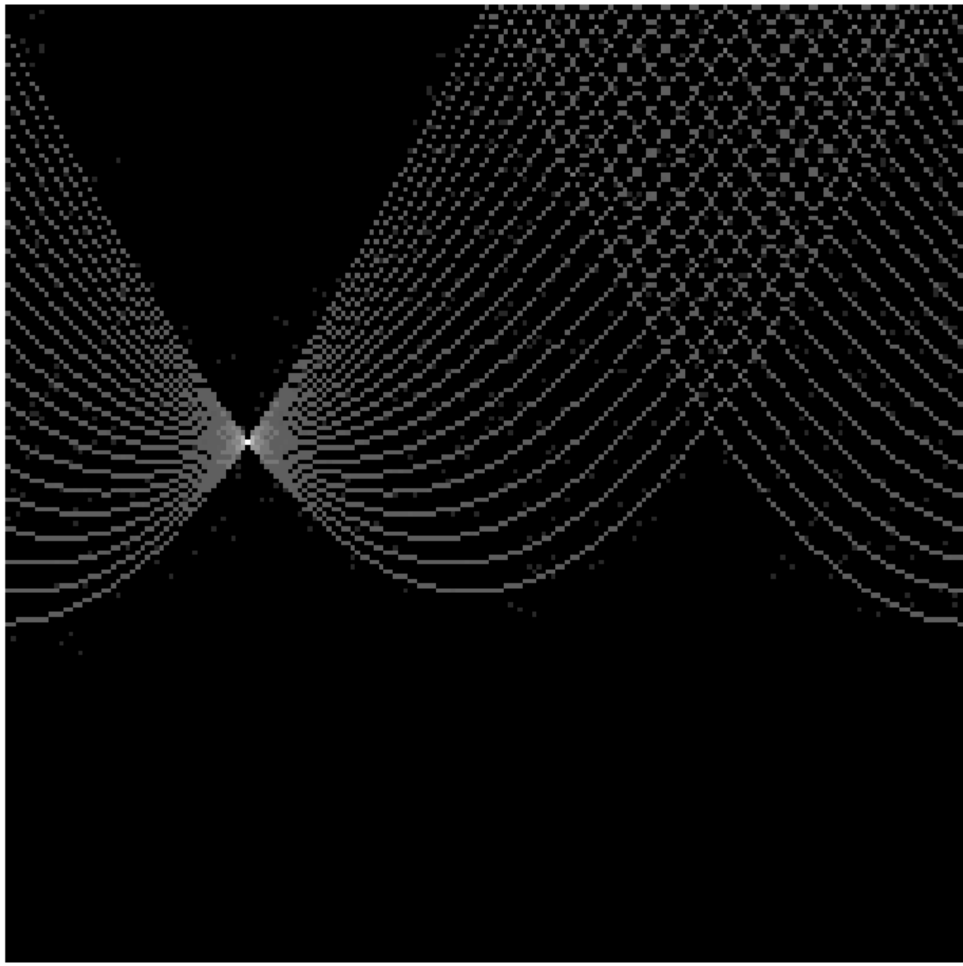
Angle / Distance Parameterization

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



Hough Transform Results

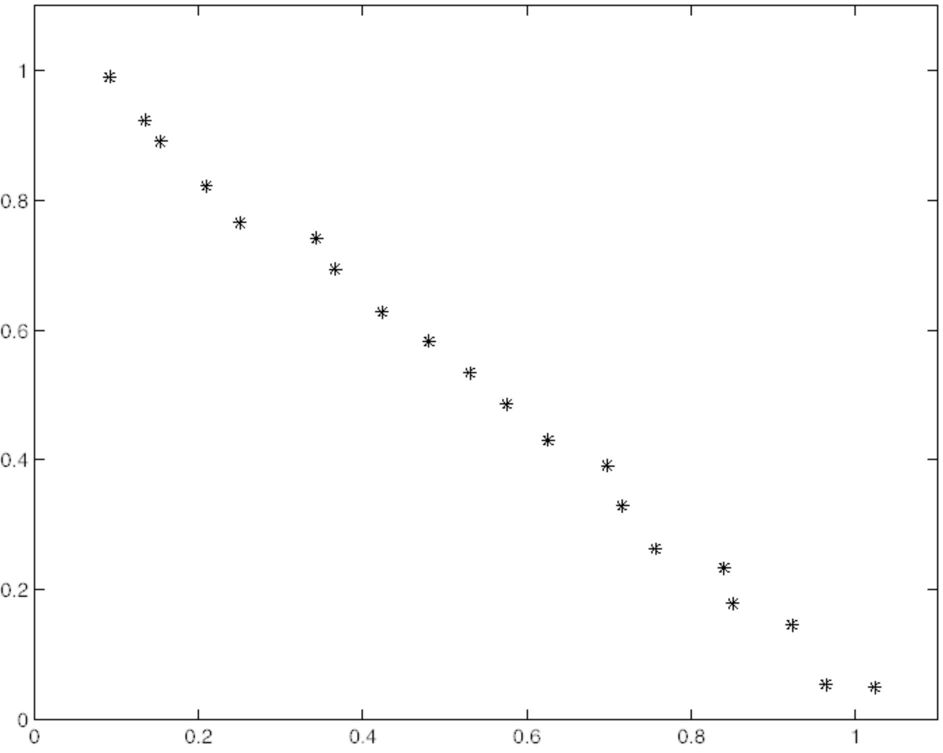


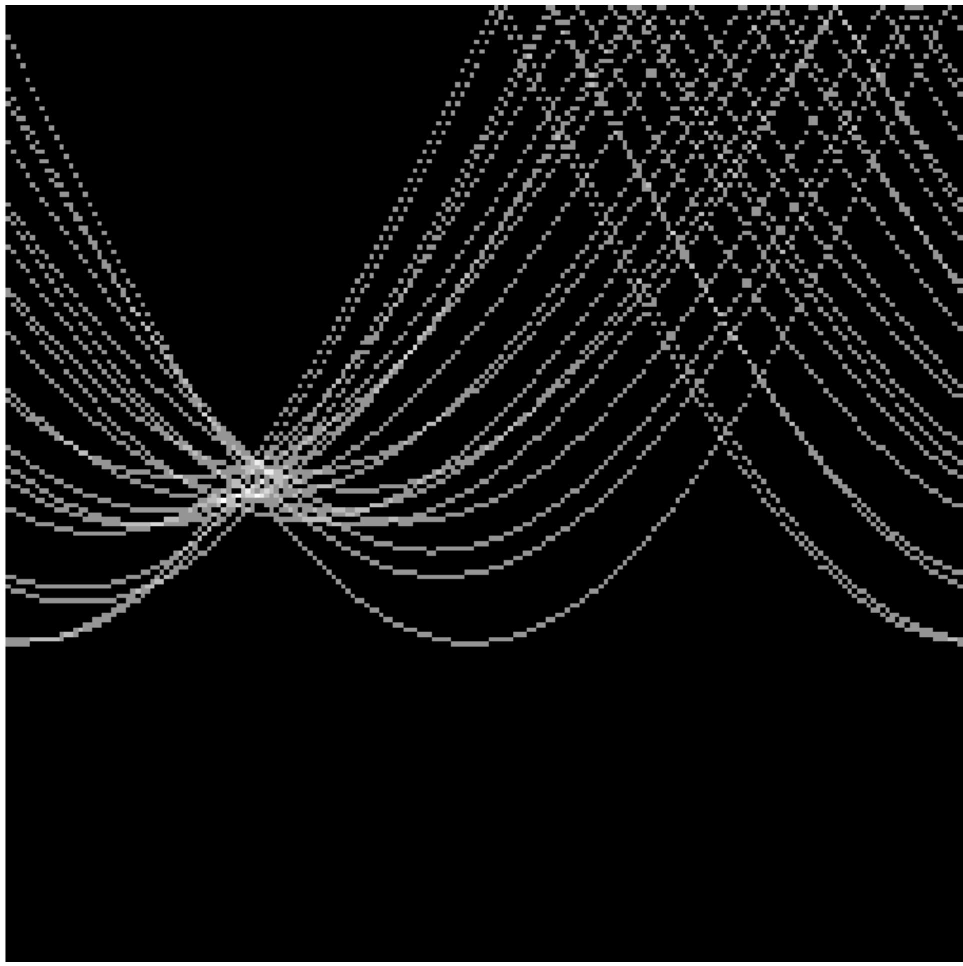


Forsyth & Ponce

Hough Transform with Noise

Peak gets fuzzy and hard to locate

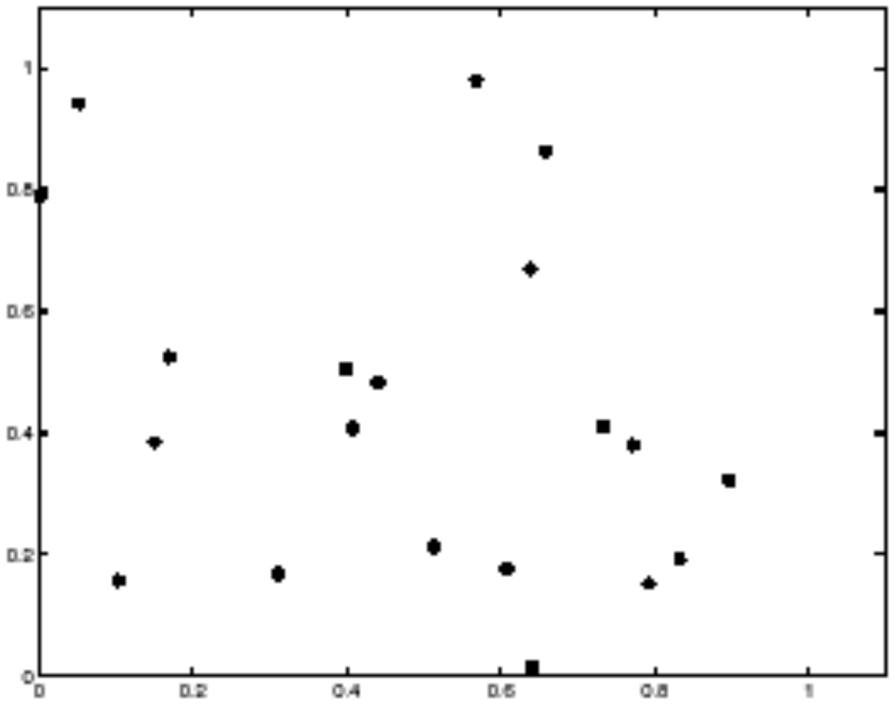


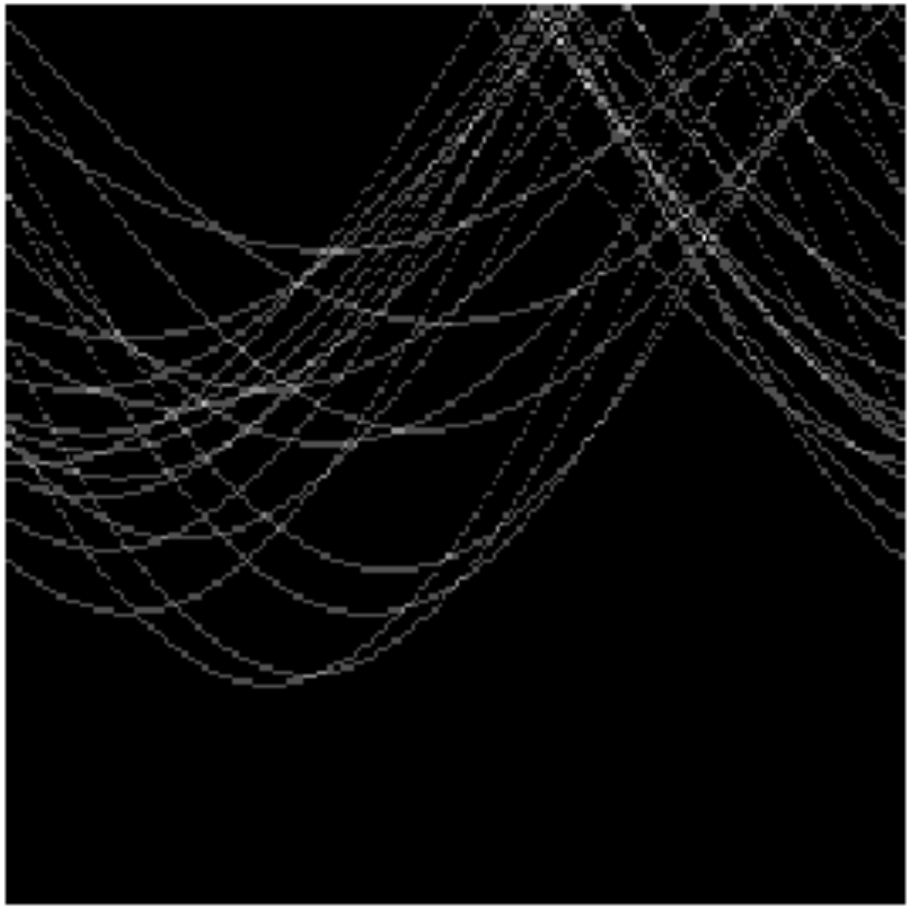


Forsyth & Ponce

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

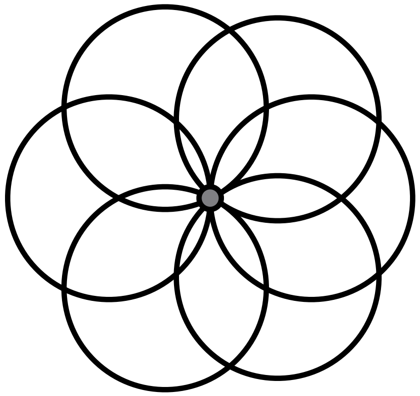
Hough transform beyond lines

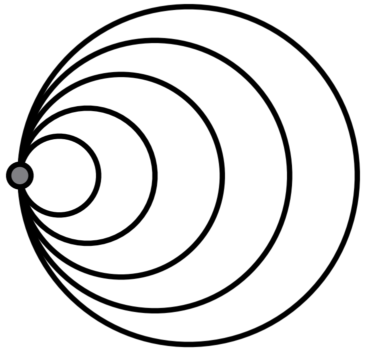
Hough Transform

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

Hough transform for circles

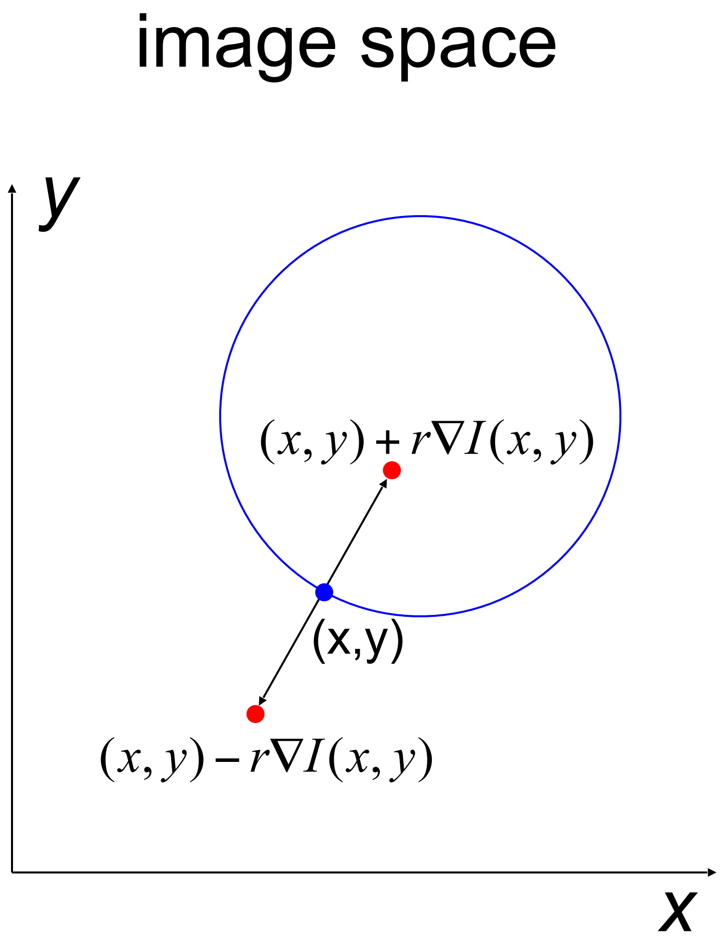
- How many dimensions will the parameter space have?
- Given an edge point, what are all possible bins that it can vote for?
- What about an *oriented* edge point?

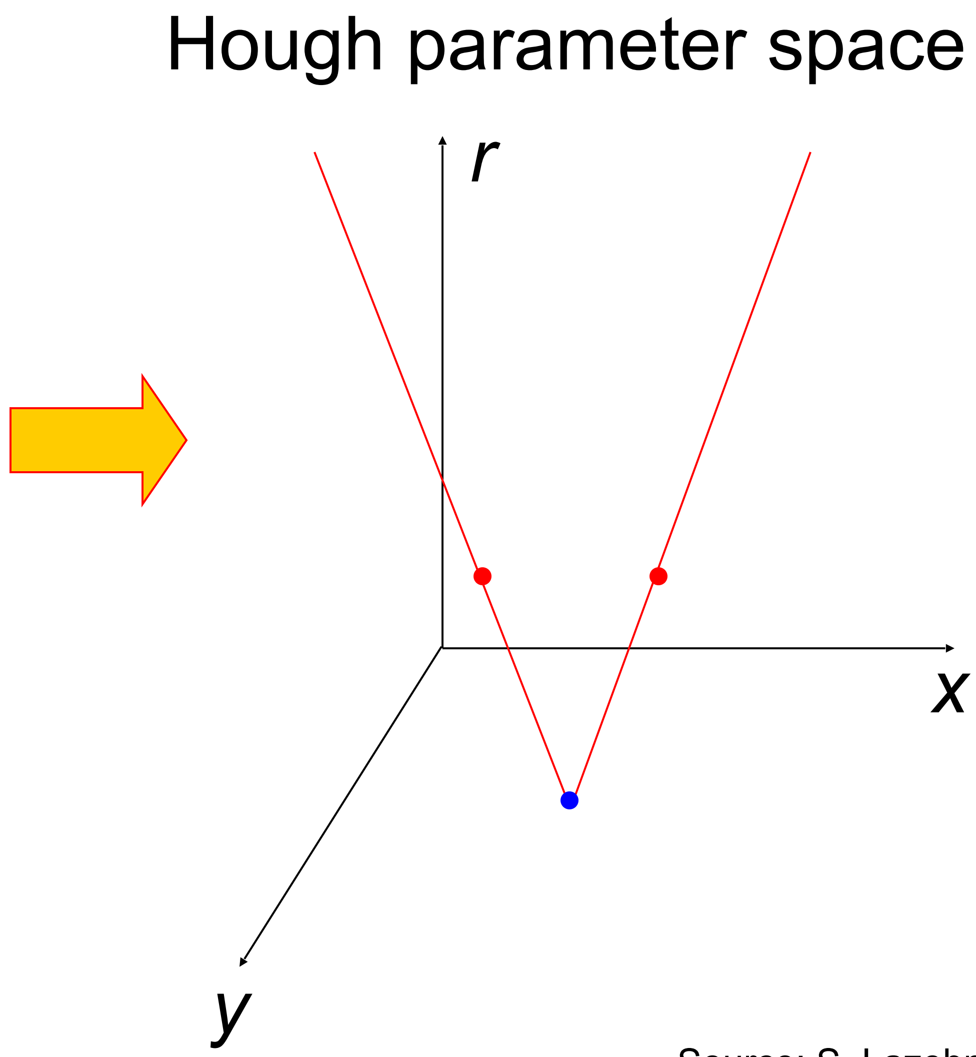




Source: S. Lazebnik

Hough transform for circles



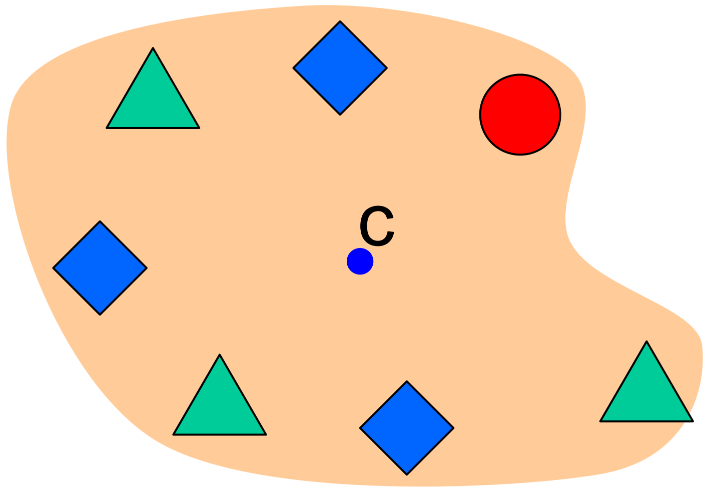


Source: S. Lazebnik

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

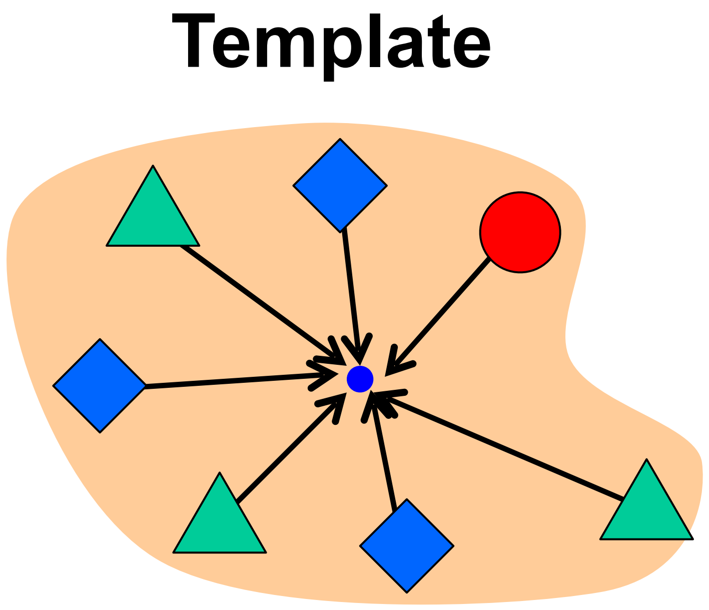
Template

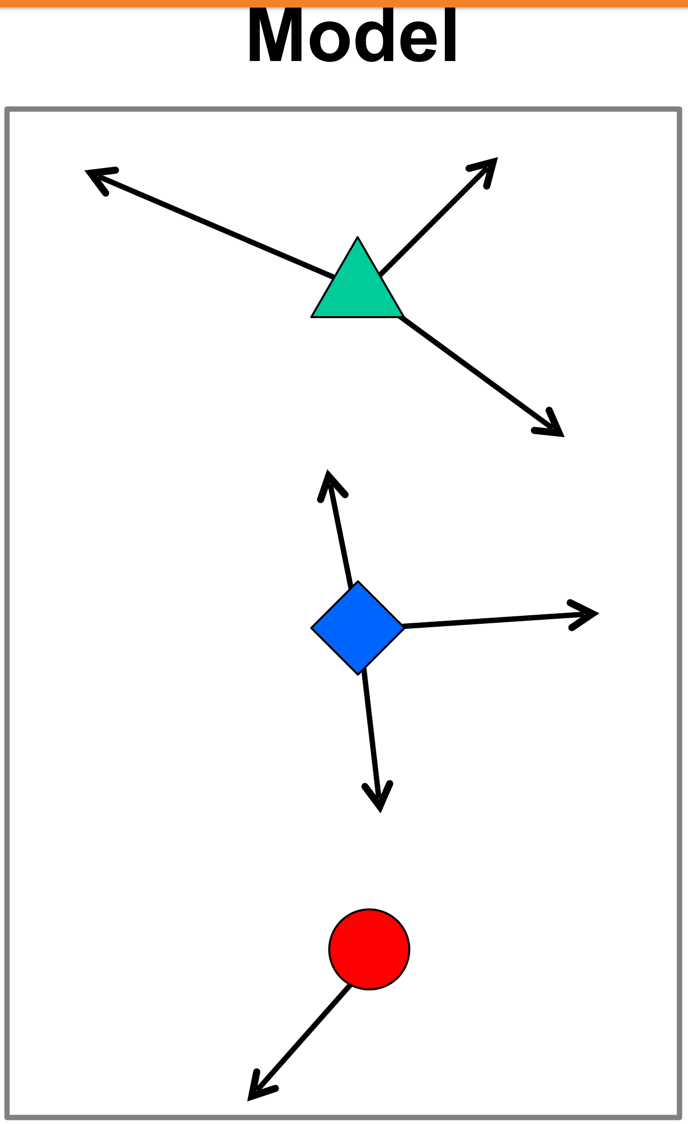


Source: S. Lazebnik

Generalized Hough transform

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



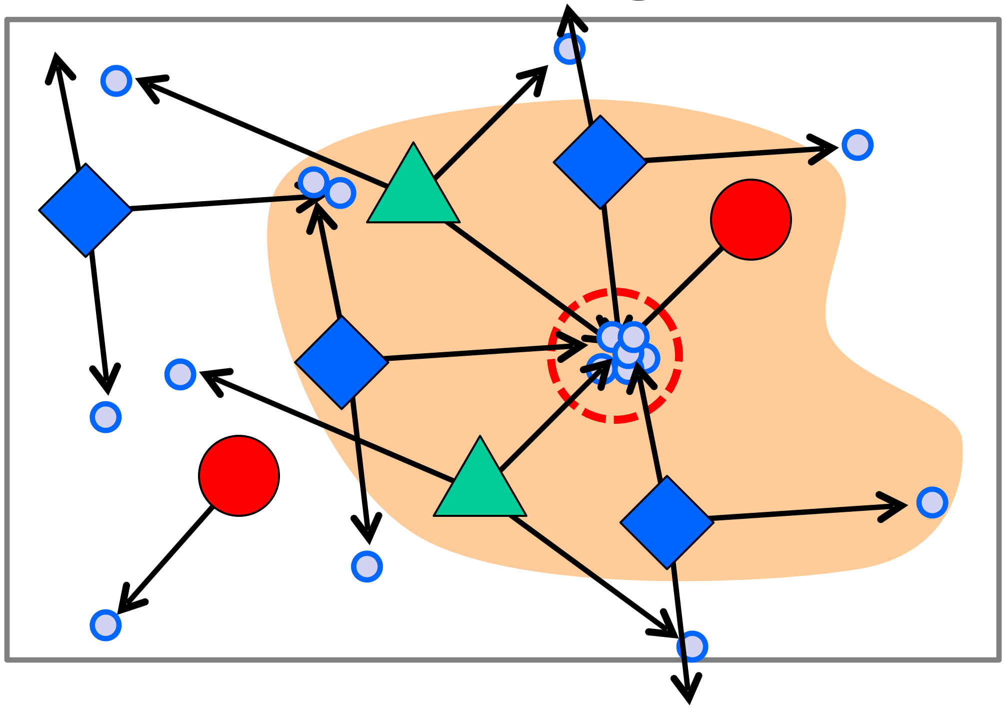


Source: S. Lazebnik

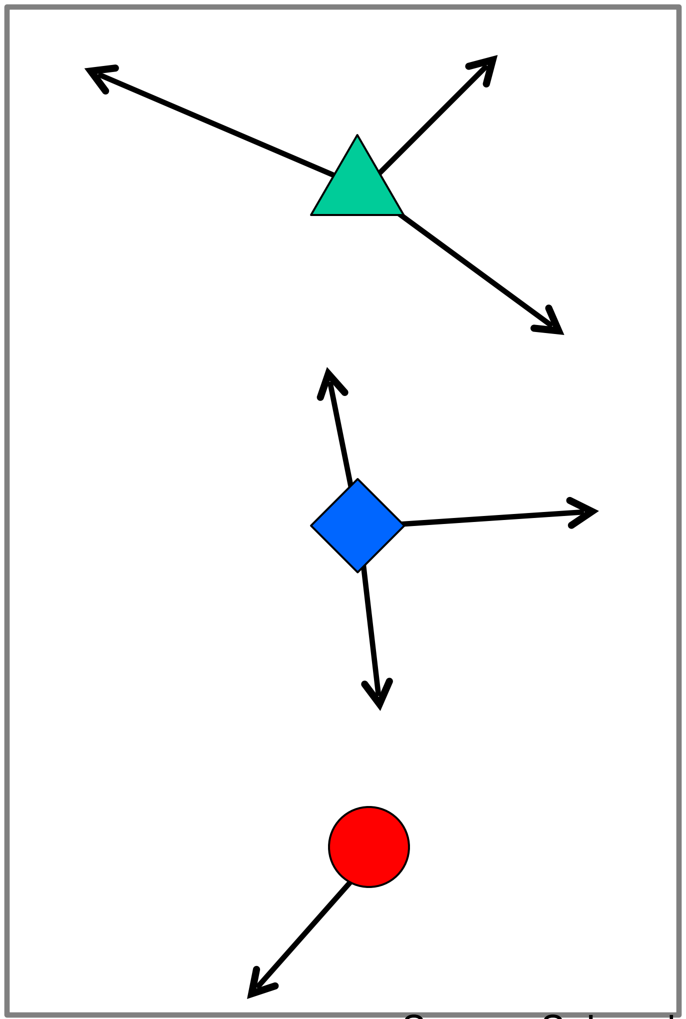
Generalized Hough transform

- 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

Test image



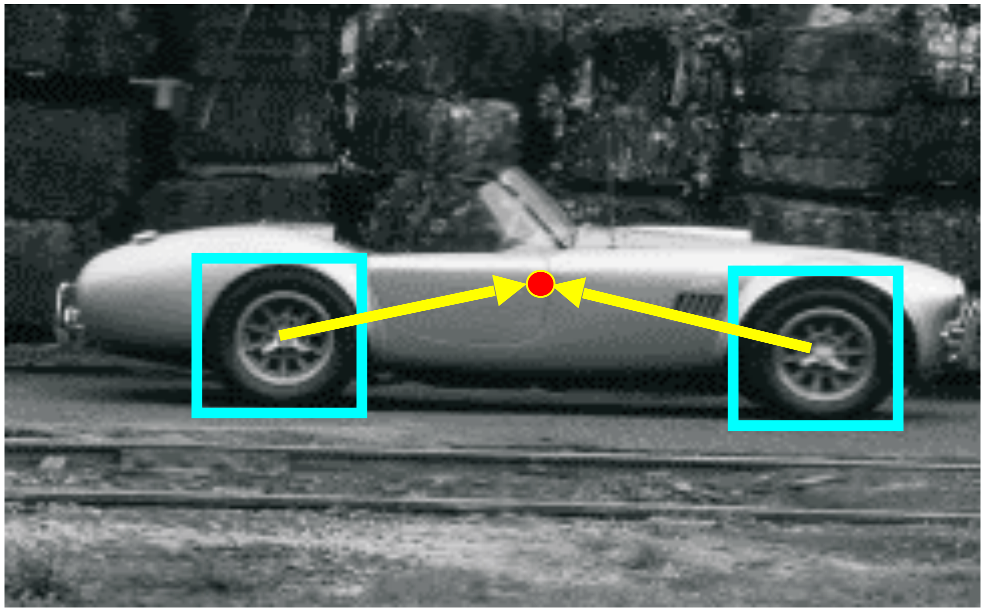
Model



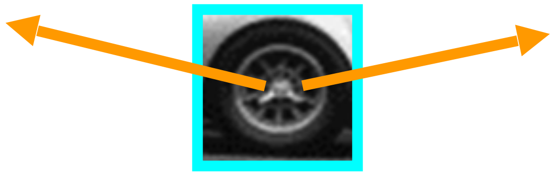
Source: S. Lazebnik

Application in recognition

- Index displacements by “visual codeword”



training image



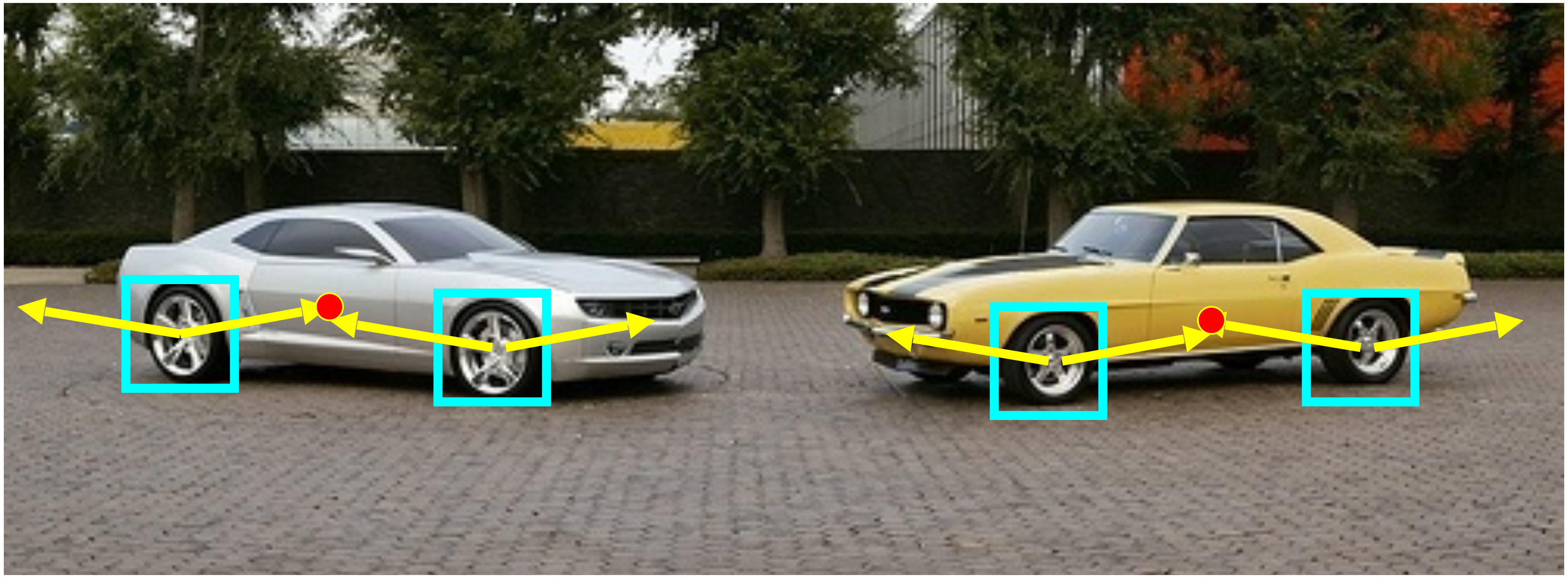
visual codeword with displacement vectors

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