Computer Vision

CS308
Feng Zheng
SUSTech CS Vision Intelligence and Perception
Week 6

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

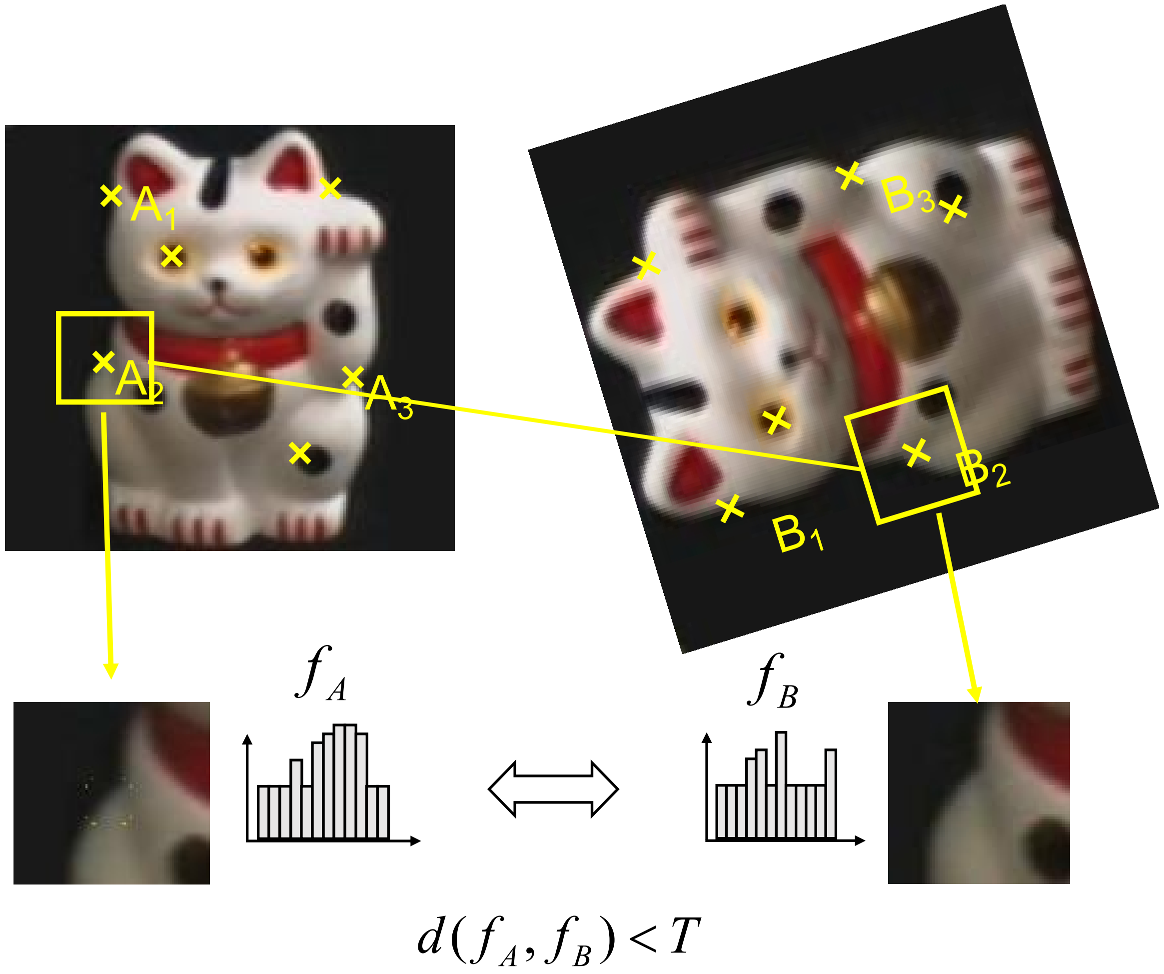
Content

- Brief Review
- Fitting Techniques
  - ➢ Least Squares
  - ➢ Total Least Squares
- Random Sample Consensus (RANSAC)
- Hough Voting
- Image Alignment

Brief Review

Overview of Keypoint Matching

- Steps
  - Find a set of distinctive keypoints
  - Define a region around each keypoint
  - Compute a local descriptor from the region
  - Match local descriptors
- Goals
  - Detect points that are repeatable and distinctive



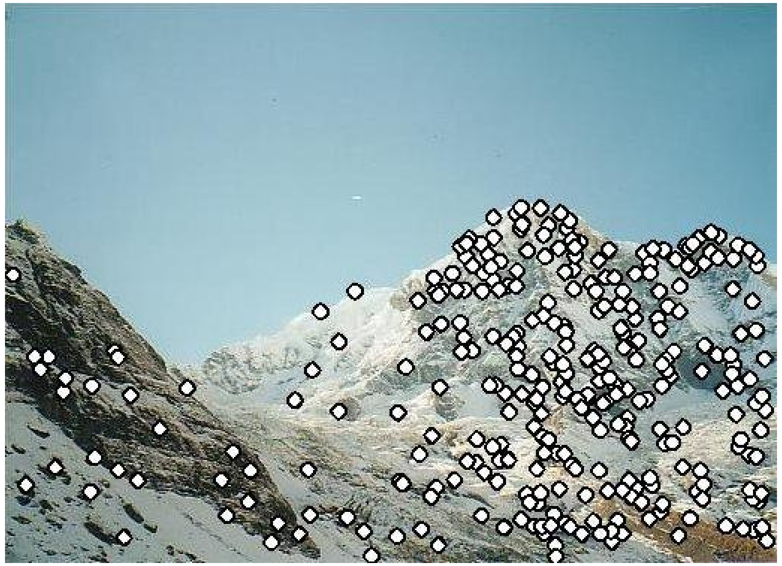
Fitting Techniques

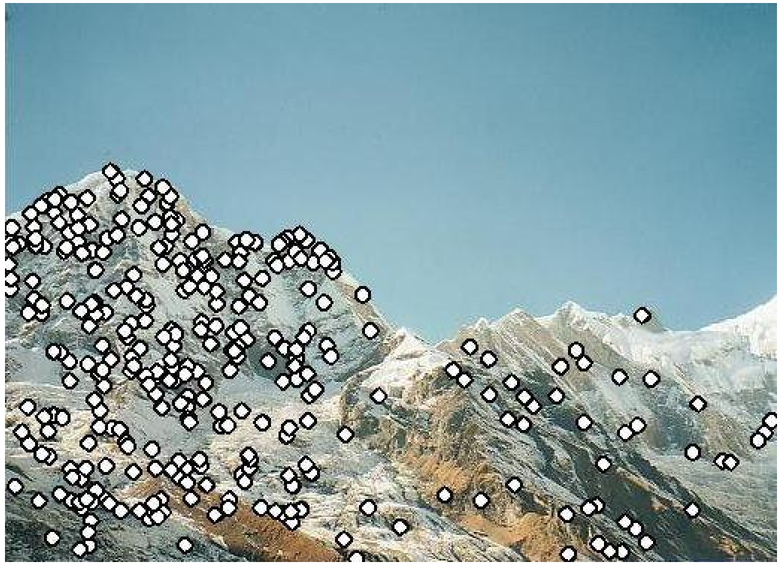
How Do We Build Panorama?

- We need to match (align) images

Matching with Features

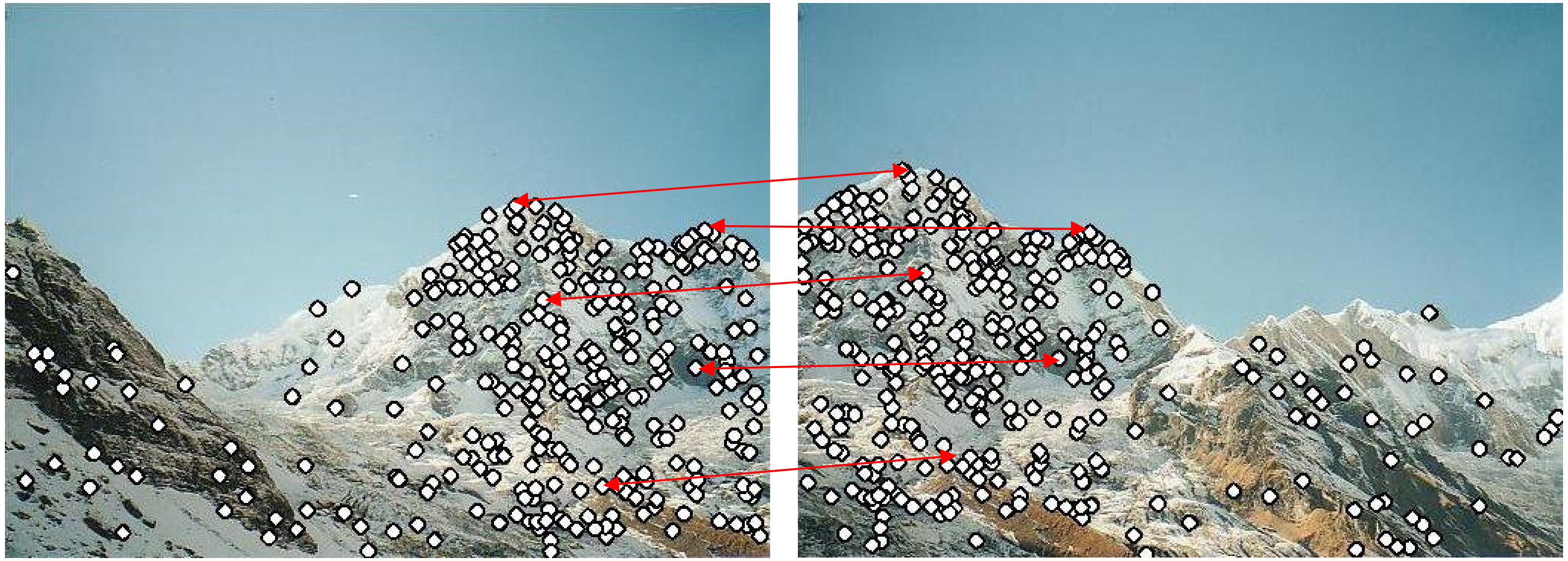
- Steps
  - Detect feature points in both images





Matching with Features

- Steps
  - Detect feature points in both images
  - Find corresponding pairs



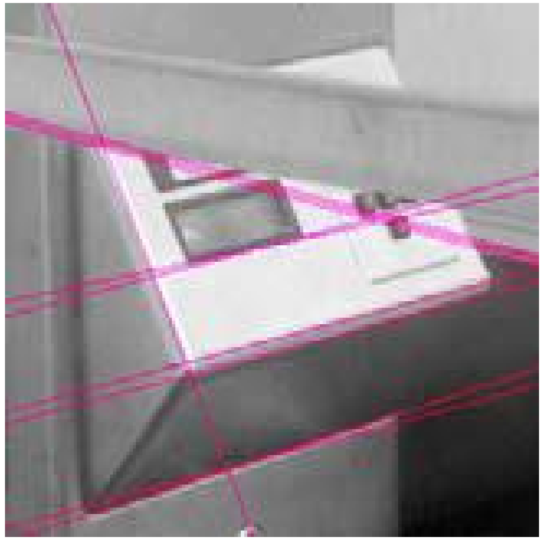
Matching with Features

- Steps
  - ➢ Detect feature points in both images
  - ➢ Find corresponding pairs
  - ➢ Use these pairs to align images

Previous Lecture

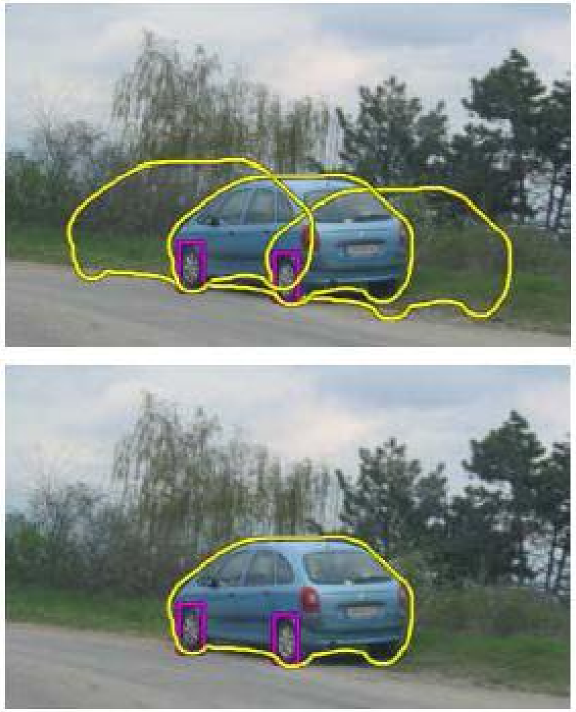
Fitting: Building a Model for a Set of Features

- Choose a parametric model to represent a set of features



Simple model: lines

Simple model: circles



Complicated model: car

Fitting: Issues

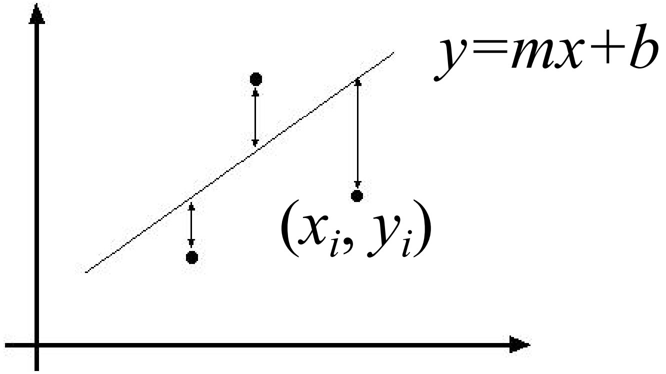
- Case study: Line detection
  - Noise in the measured feature locations
  - Extraneous data: clutter (outliers), multiple lines
  - Missing data: occlusions

Fitting: Issues

- 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

Line Fitting: Ordinary Least Squares

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m\,x_i + b$
- Find (m, b) to minimize

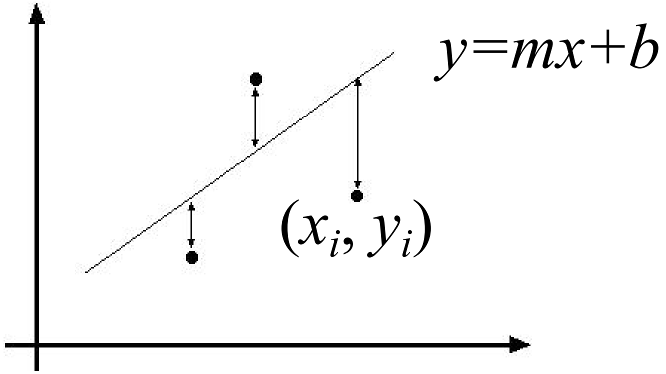


$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$

We know which points belong to the line

Line Fitting: Ordinary Least Squares

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m\,x_i + b$
- Find (m, b) to minimize



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

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

$$= (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

Line Fitting: Ordinary Least Squares

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

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

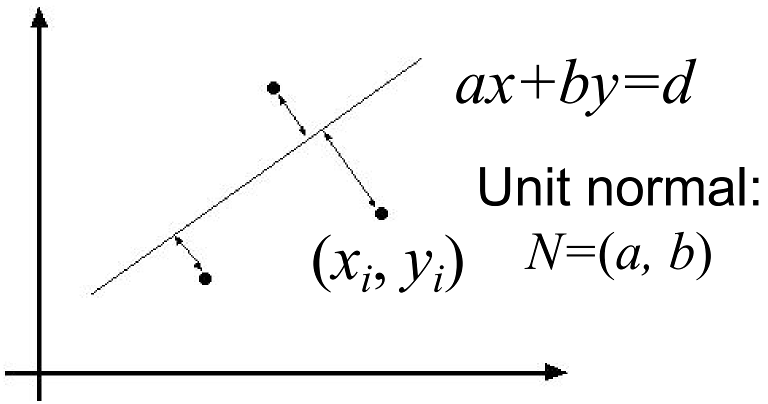
- Problem with "vertical" least squares
  - ➢ Not rotation-invariant
  - ➢ Fails completely for vertical lines $X^T X$

Total Least Squares

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

$$|ax_i + by_i - d|$$

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



Total Least Squares

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

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

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

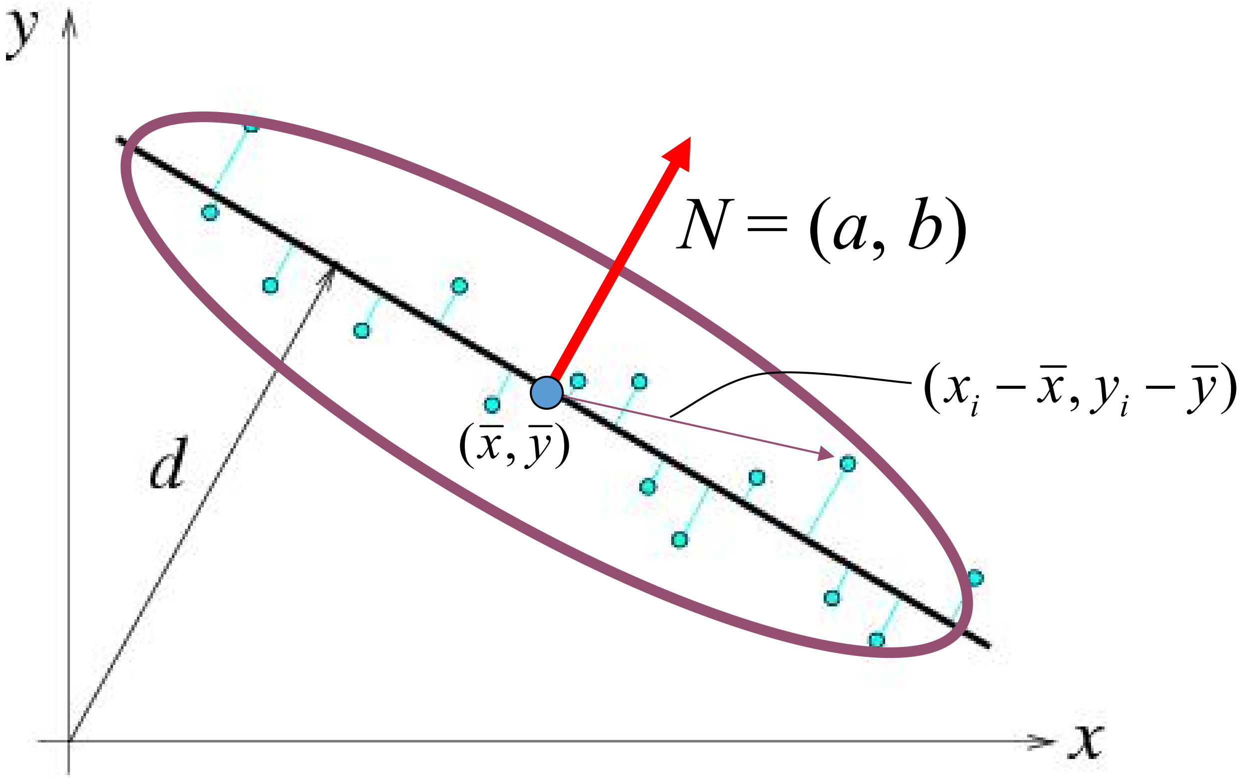
- Solution to $(U^TU)N = 0$, subject to $\|N\|^2 = 1$: eigenvector of U^TU associated with the smallest eigenvalue (least squares solution to *homogeneous linear system* $UN = 0$)

Total Least Squares

- Second moment matrix

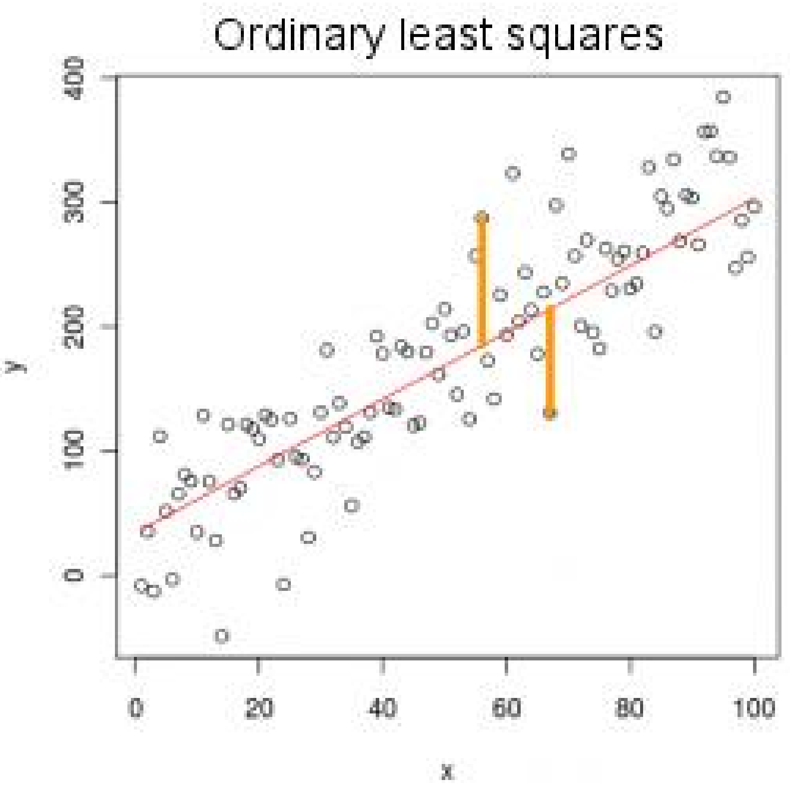
$$U = \begin{bmatrix} x_1 - \overline{x} & y_1 - \overline{y} \\ \vdots & \vdots \\ x_n - \overline{x} & y_n - \overline{y} \end{bmatrix}$$

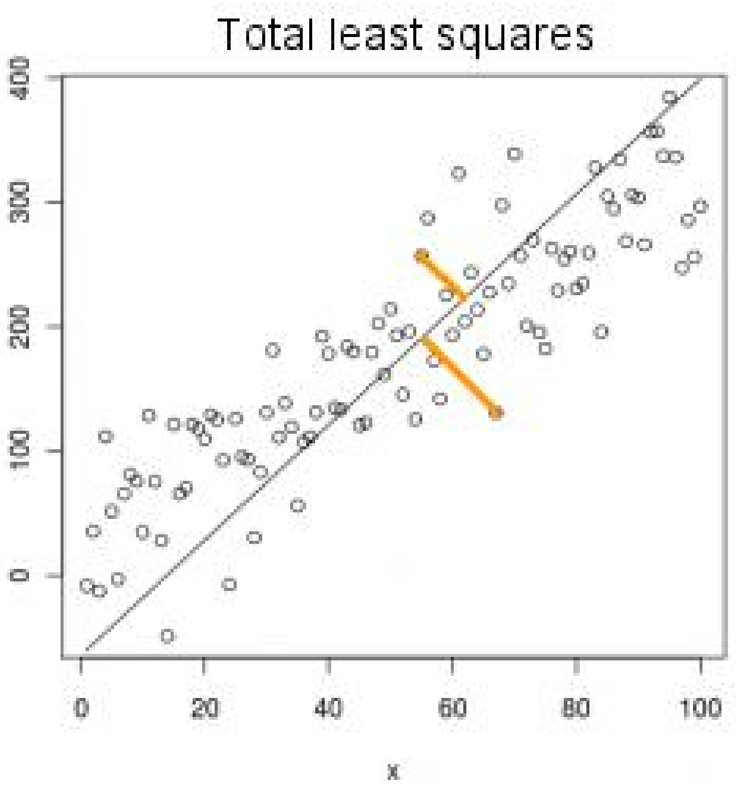
$$U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \overline{x})^2 & \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) \\ \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) & \sum_{i=1}^{n} (y_i - \overline{y})^2 \end{bmatrix}$$



OLS vs. TLS

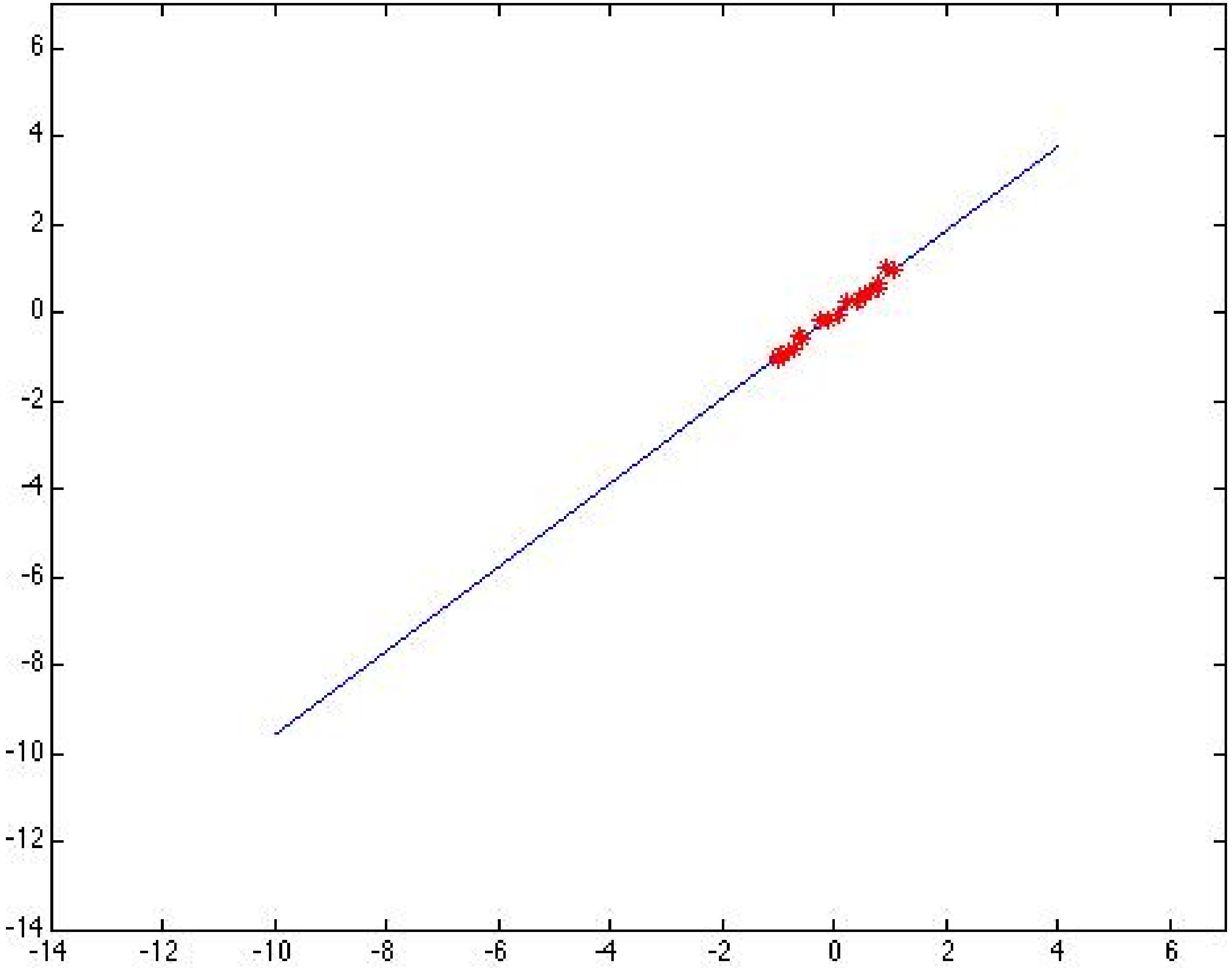
- The difference between standard OLS regression and "orthogonal" TLS regression





Total Least Squares

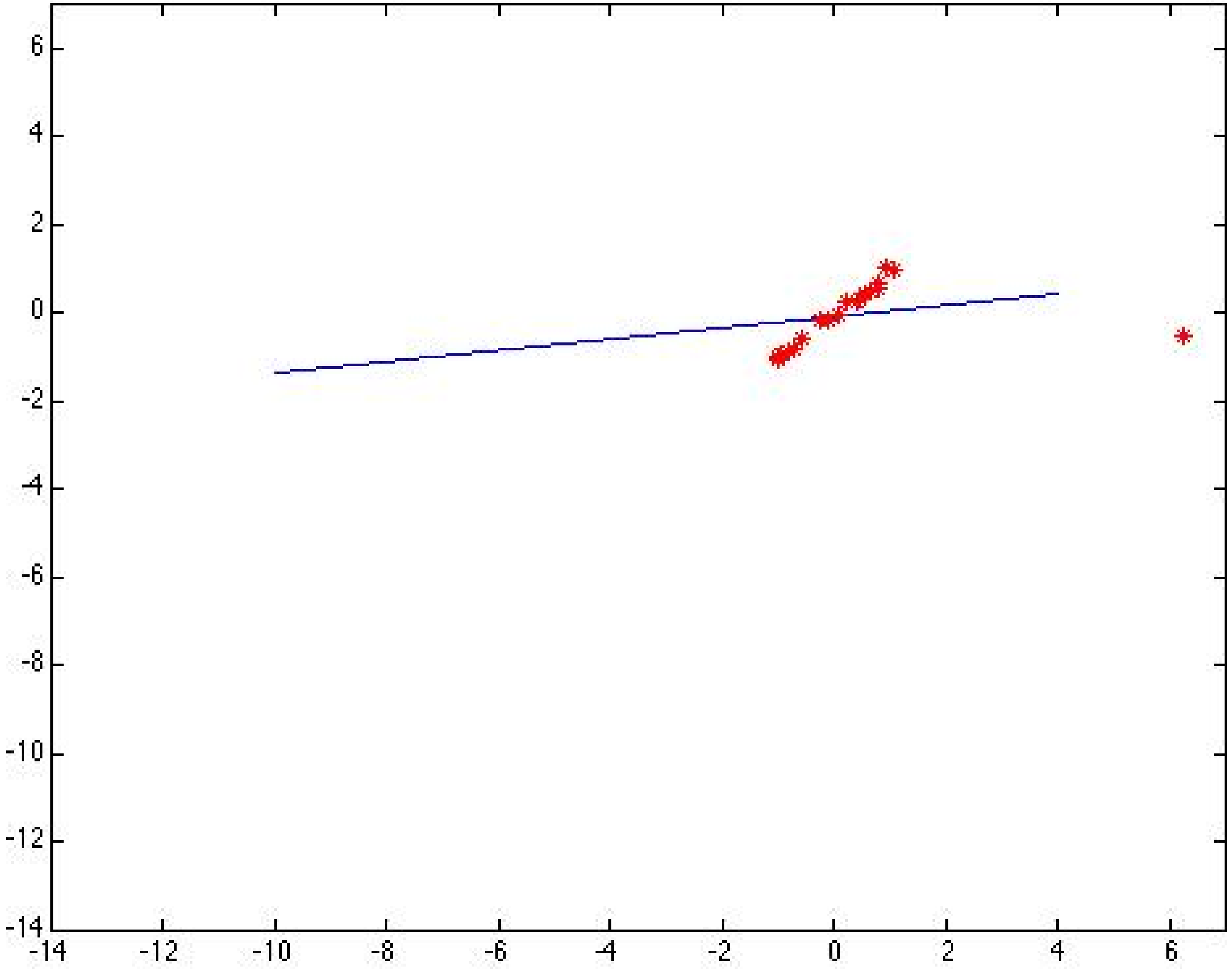
- Robustness to noise: least squares fit to the red points



Total Least Squares

- Robustness to noise: Least squares fit with an outlier

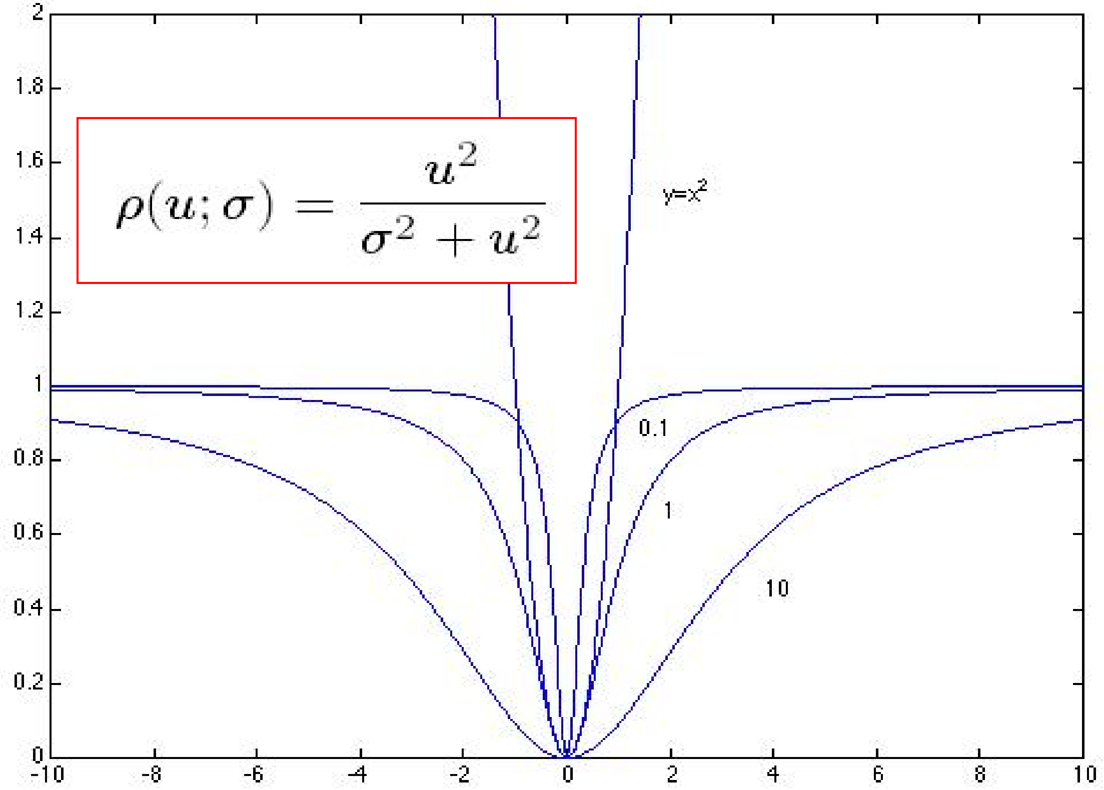
- Problem: squared error heavily penalizes outliers



Robust Estimators

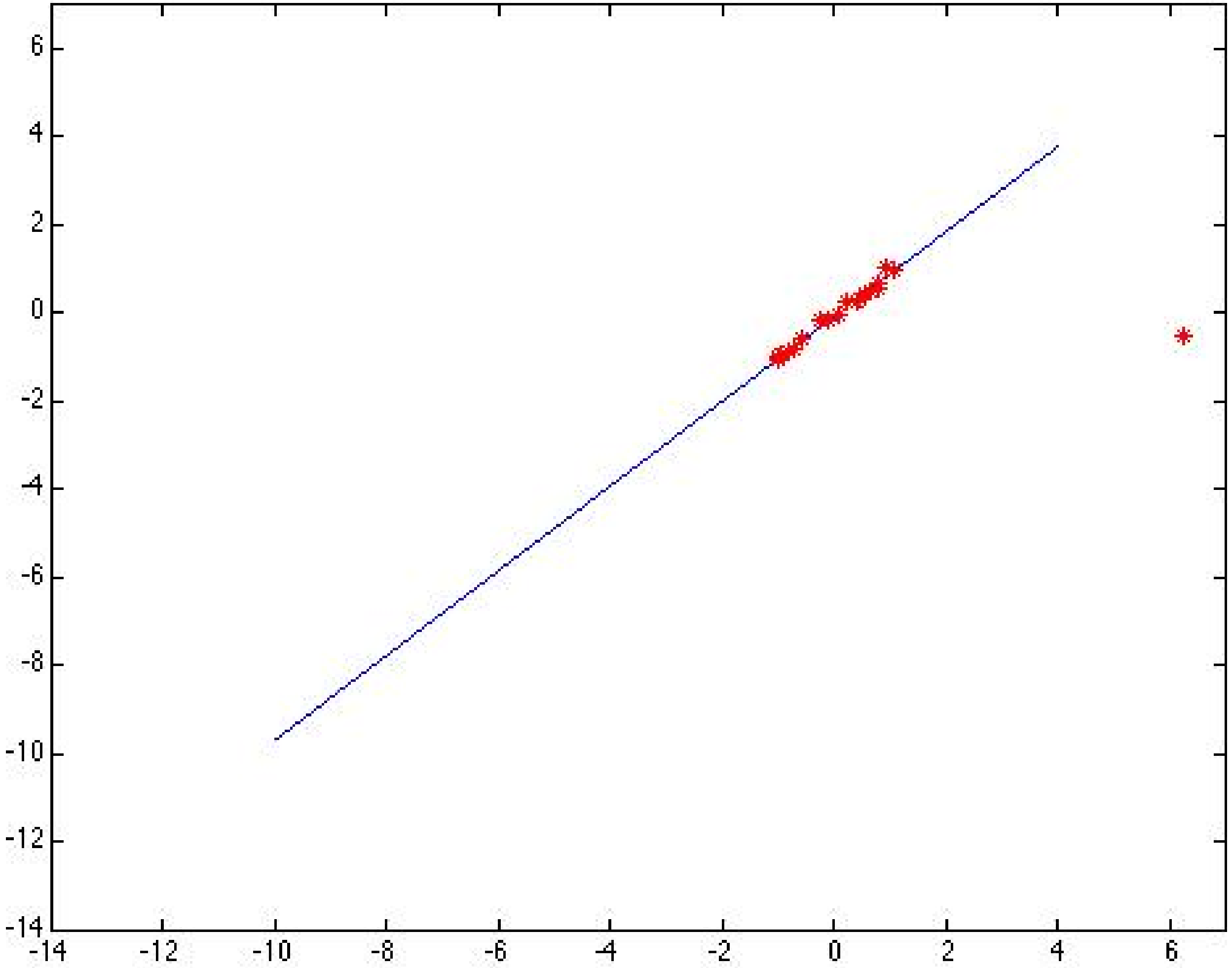
- General approach---minimize: $\sum_i \rho(r_i(x_i, \theta); \sigma)$
  $r_i(x_i, \theta)$ – residual of ith point w.r.t. model parameters θ
  ρ – robust function with scale parameter σ

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



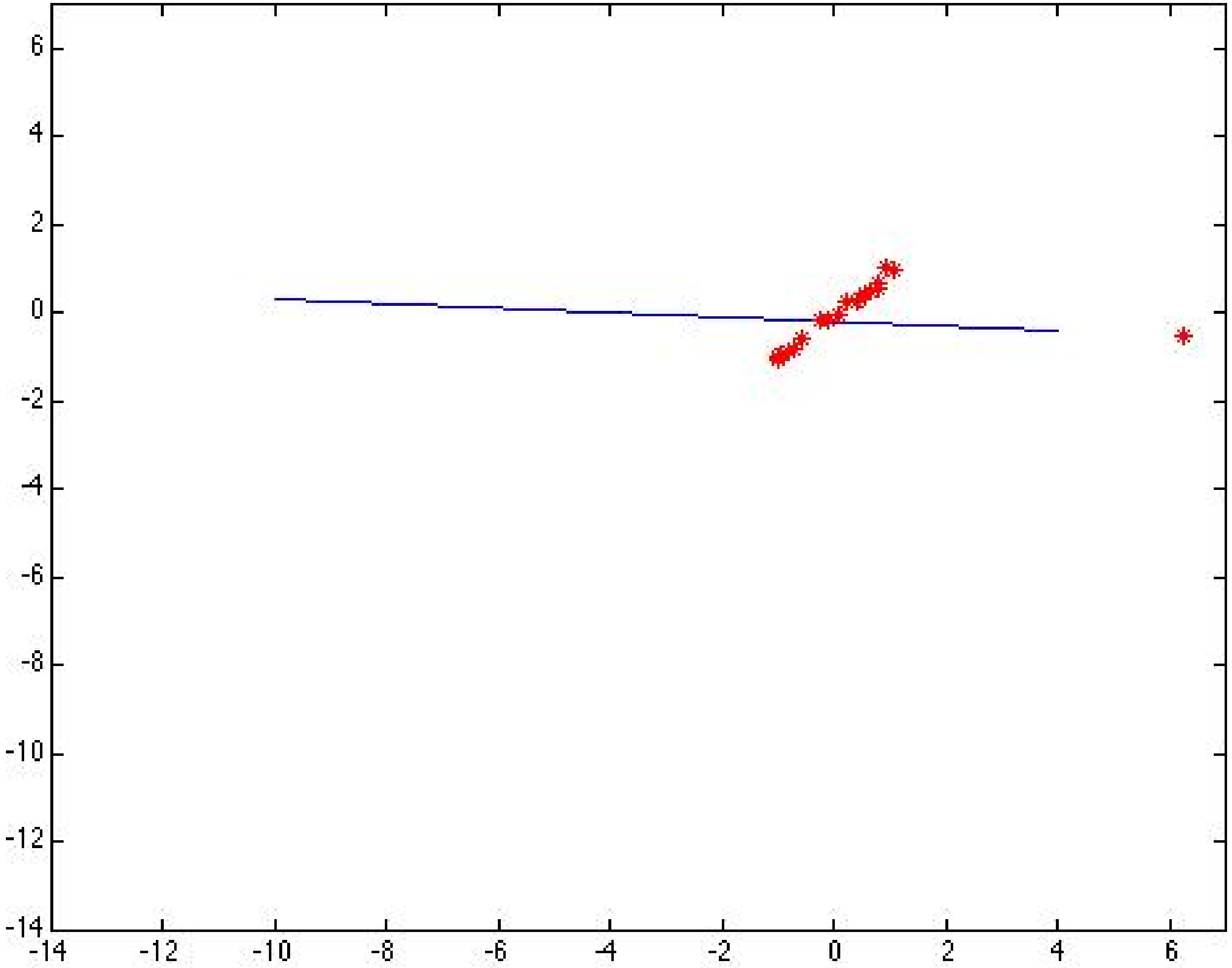
Choosing the Scale: Just Right

- The effect of the outlier is minimized, when choosing a just right scale



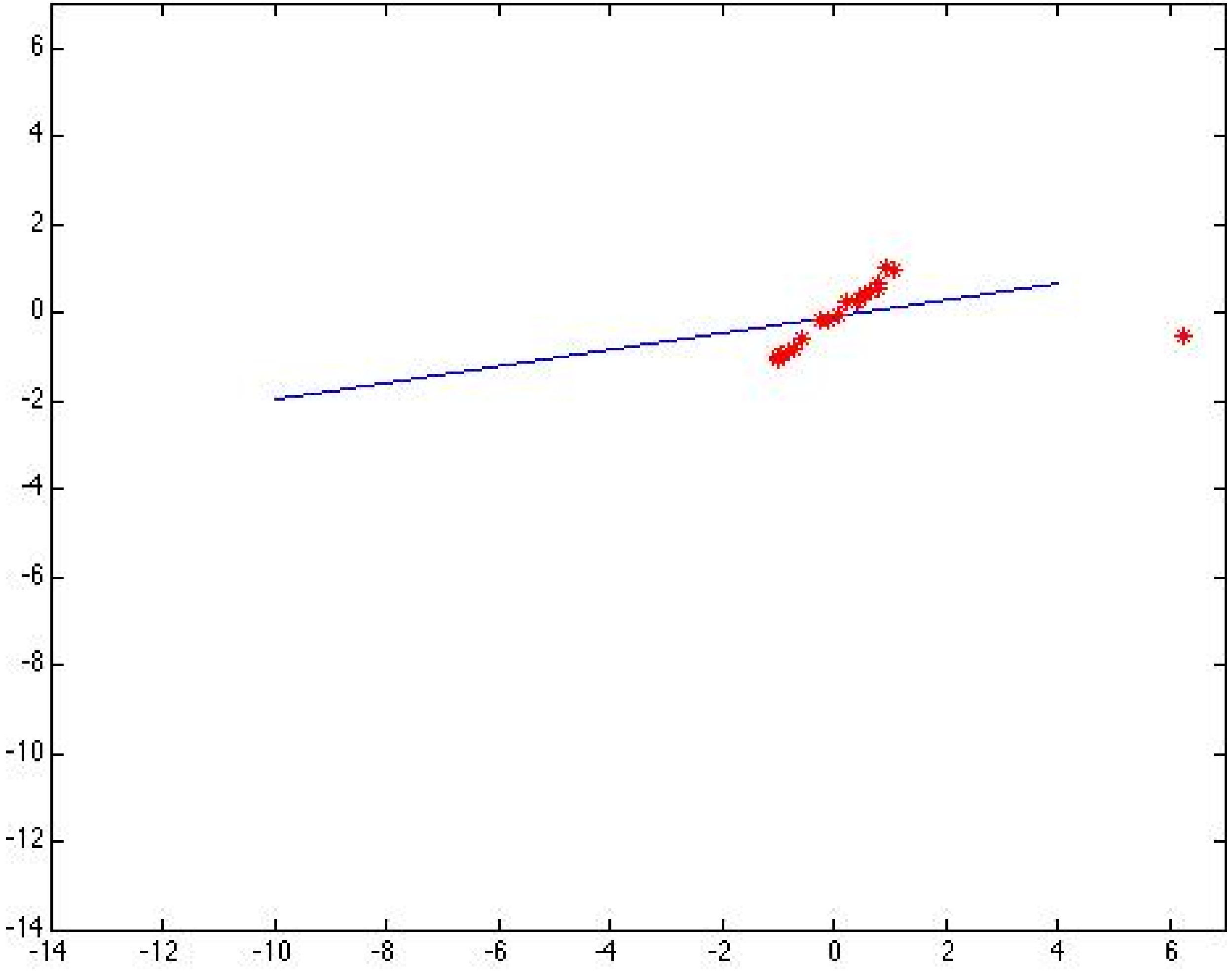
Choosing the Scale: Too Small

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



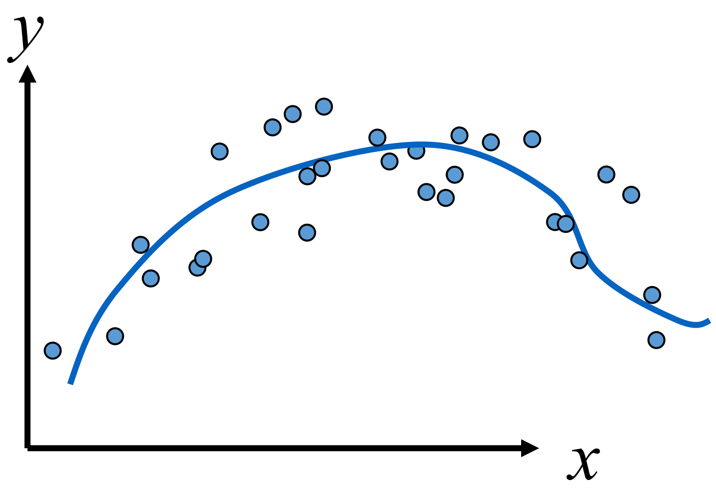
Choosing the Scale: Too Large

- Behaves much the same as least squares



Curve Fitting

- Find Polynomial: $y = f(x) = ax^3 + bx^2 + cx + d$
  - ➢ That best fits the given points (x_i, y_i)



- Minimize: $\frac{1}{N}\sum_i [y_i - (ax_i^3 + bx_i^2 + cx_i + d)]^2$

- Using: $\frac{\partial E}{\partial a} = 0$, $\frac{\partial E}{\partial b} = 0$, $\frac{\partial E}{\partial c} = 0$, $\frac{\partial E}{\partial d} = 0$

- Note: $f(x)$ is LINEAR in the parameters (a, b, c, d)

Random Sample Consensus

RANSAC

算法的基本步骤是：

从数据中随机选择一个小的子集。

对该子集进行模型拟合（例如，拟合一条直线）。

将所有与该模型接近的数据点（即“内点”）找到，其他不符合的点则视为离群点。

重复这个过程多次，选择效果最好的模型。

- Robust fitting (TLS) 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

- Algorithm
- Repeat N times:
  - Draw s points uniformly at random
  - Fit line to these s points (TLS)
  - 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
- End
- Four parameters: s, t, d and N

Choosing the Parameters

- Initial number of points s
  - ➢ Minimum number needed to fit the model
    - ✓2 points

- Distance threshold t
  - ➢ (1) Choose t so probability for inlier is p (e.g. 0.95)
  - ➢ (2) Zero-mean Gaussian noise with standard deviation σ: $t^2 = 3.84\sigma^2$

如果数据服从零均值高斯噪声，且标准差为
σ，则可以选择 t=3.84σ 作为距离阈值。

Choosing the Parameters

算法多次执行，以确保至少有一个随机选择的样本是没有离群点的。

- Number of times N
  - ➢ Choose N so that, with probability p, at least one random sample is free from outliers (e.g. $p=0.99$)

Desired success rate after N times: p

Outlier ratio (Unknown): e

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

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

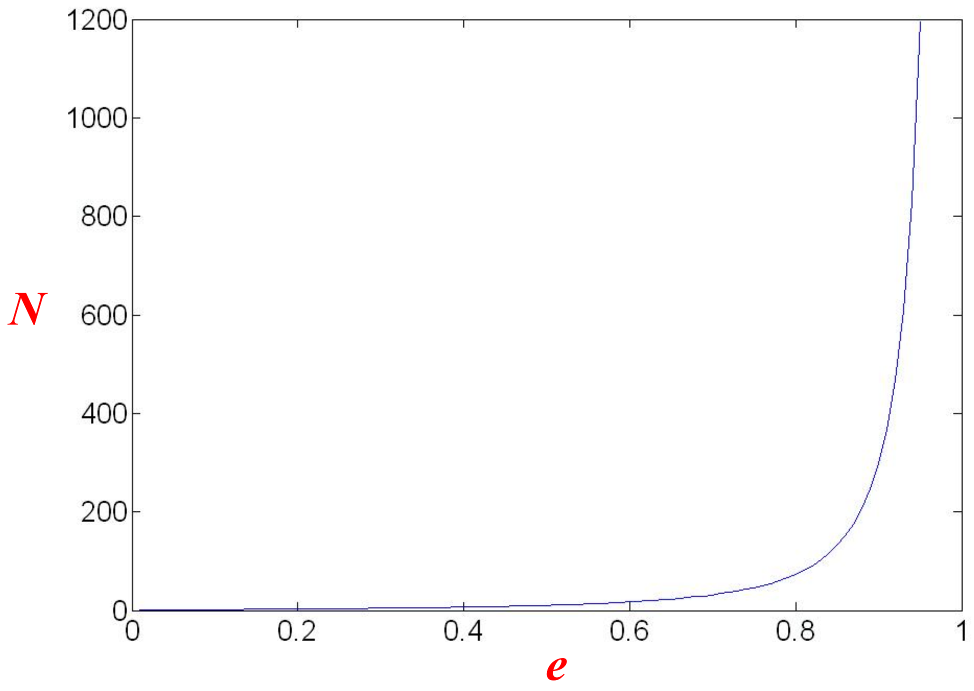
N	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Choosing the Parameters

- Consensus set size d (number of inliers)
  - ➢ Should match expected inlier ratio

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

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



Adaptively determining the number of samples

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

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

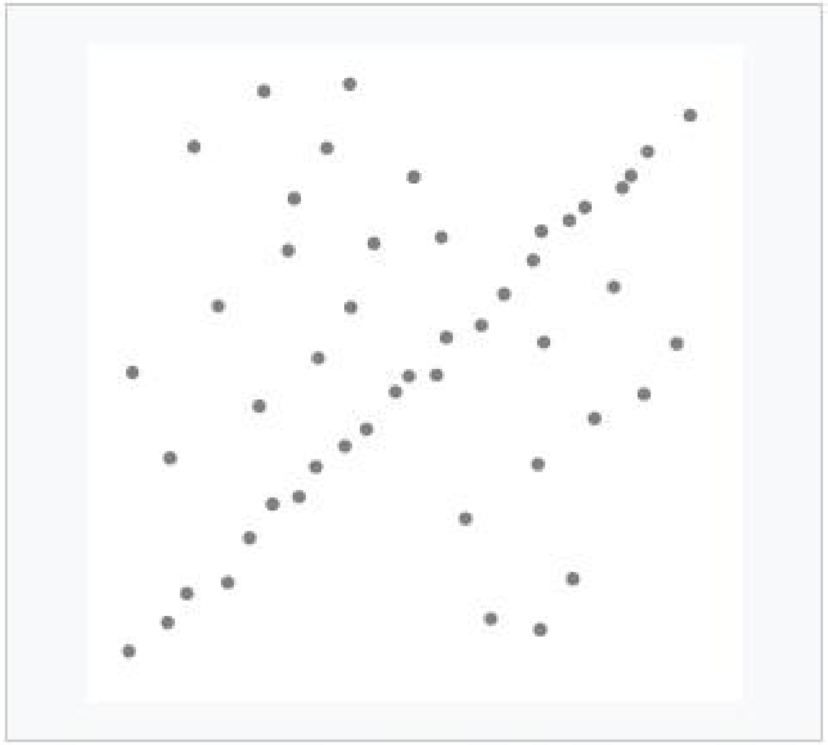
    - ✓ Increment the $sample\_count$ by 1

灵活性：通过动态调整内点比例，算法能根据数据的实际情况优化样本数量，避免过度计算或不足计算。

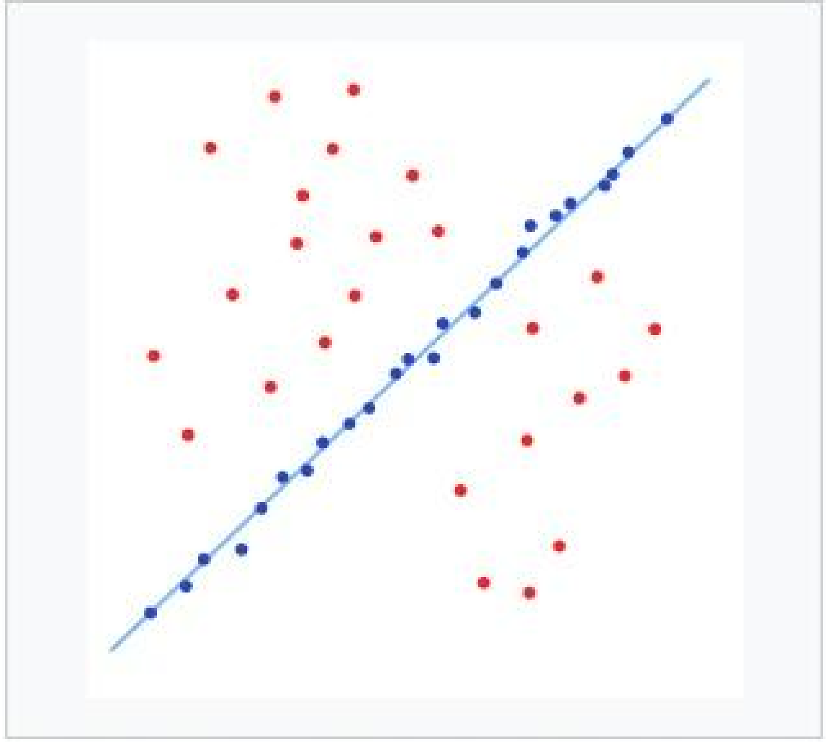
适应性：如果在迭代过程中发现更多的内点（例如内点比例达到80%），算法会自动调整 e 的值，使得后续的样本选择更加有效。

RANSAC

- An example



A data set with many outliers for which a line has to be fitted.



Fitted line with RANSAC; outliers have no influence on the result.

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of parameters to tune
  - Can't always get a good initialization of the model based on the minimum number of samples
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios
  - We can often do better than brute-force sampling

Hough transform

Voting Schemes

- Principal of voting
  - Let each feature (voter) vote for all the models that are compatible with it
  - Hopefully the noise features (voter) will not vote consistently for any single model (nominator)
  - Missing data doesn't matter as long as there are enough features remaining to agree on a good model

Voting Schemes (投票方案)
投票原理：
每个特征点（投票者）对与之兼容的所有模型进行投票。
希望噪声特征（不相关的数据点）不会对任何单一模型（提名者）产生一致的投票。
只要有足够的特征点一致地投票给一个好的模型，缺失数据就不会影响结果。
关键点：
特征点投票：每个特征点会对多个模型进行投票，而不是仅对一个模型。
噪声和缺失数据的鲁棒性：噪声点不会影响最终的结果，缺失数据也不影响，只要有足够多的合适特征点。

Hough Transform

离散化参数空间：将参数空间离散化为多个小的单元（bins）。

为每个特征点投票：对于图像中的每个特征点，都会在参数空间中的每个可能的bin内投票，这些bin代表可以生成该特征点的模型。

找到得票最多的bin：在参数空间中，得票最多的bin对应着最佳的模型参数，即检测到的形状或特征。

- 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

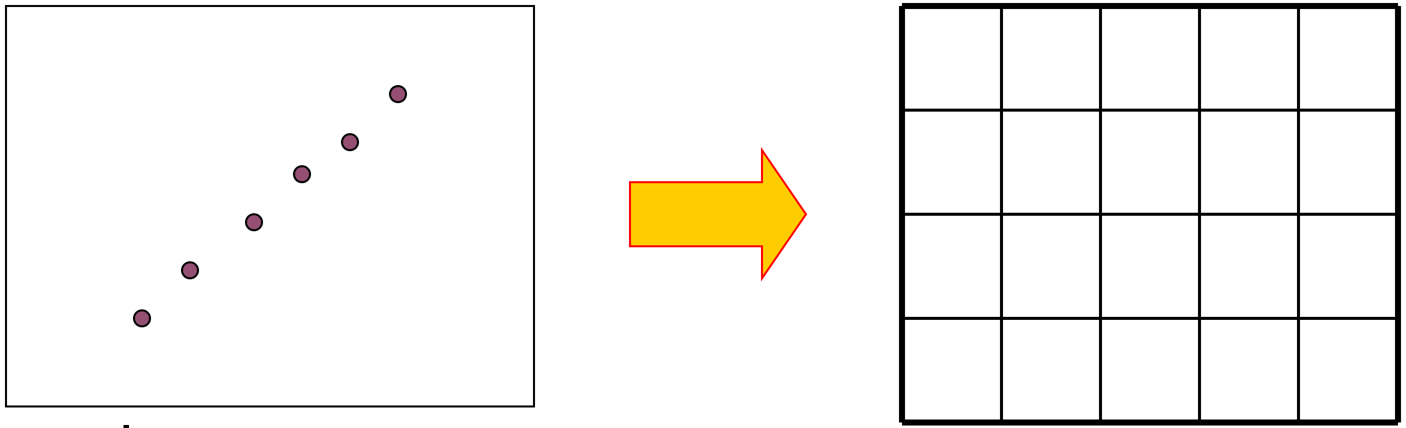
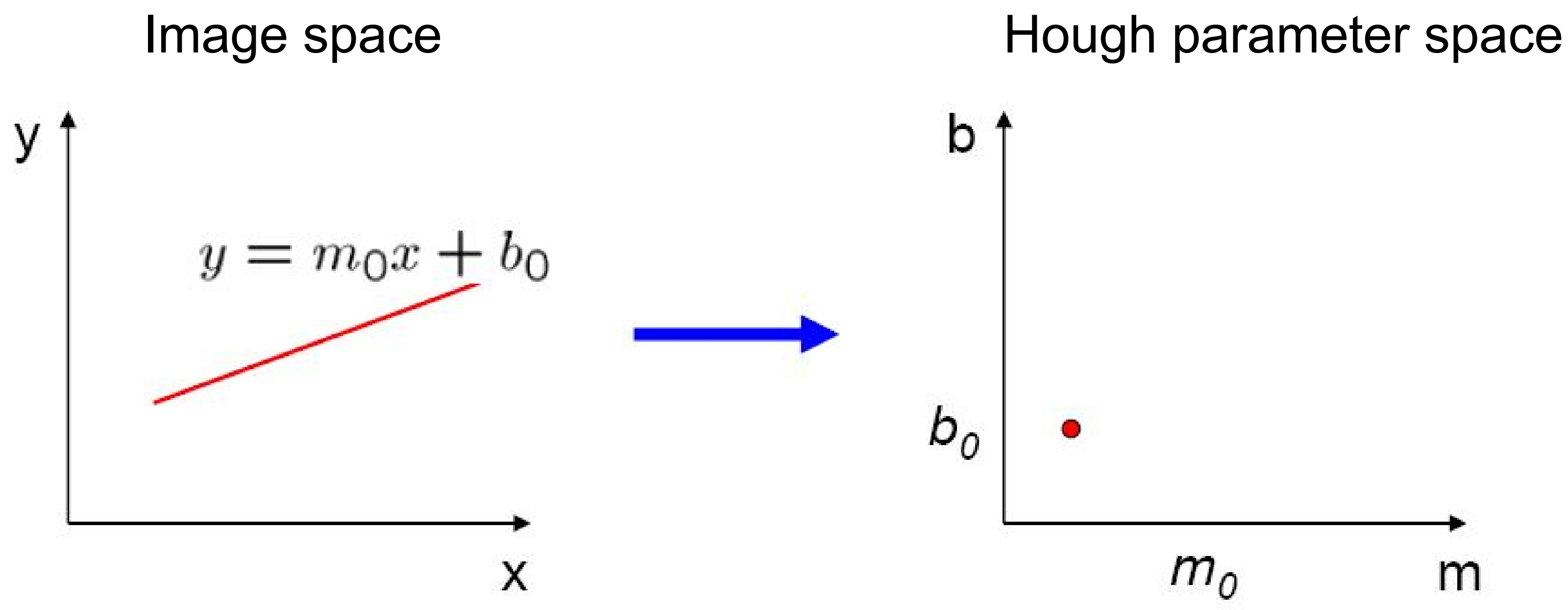


Image space

Hough parameter space

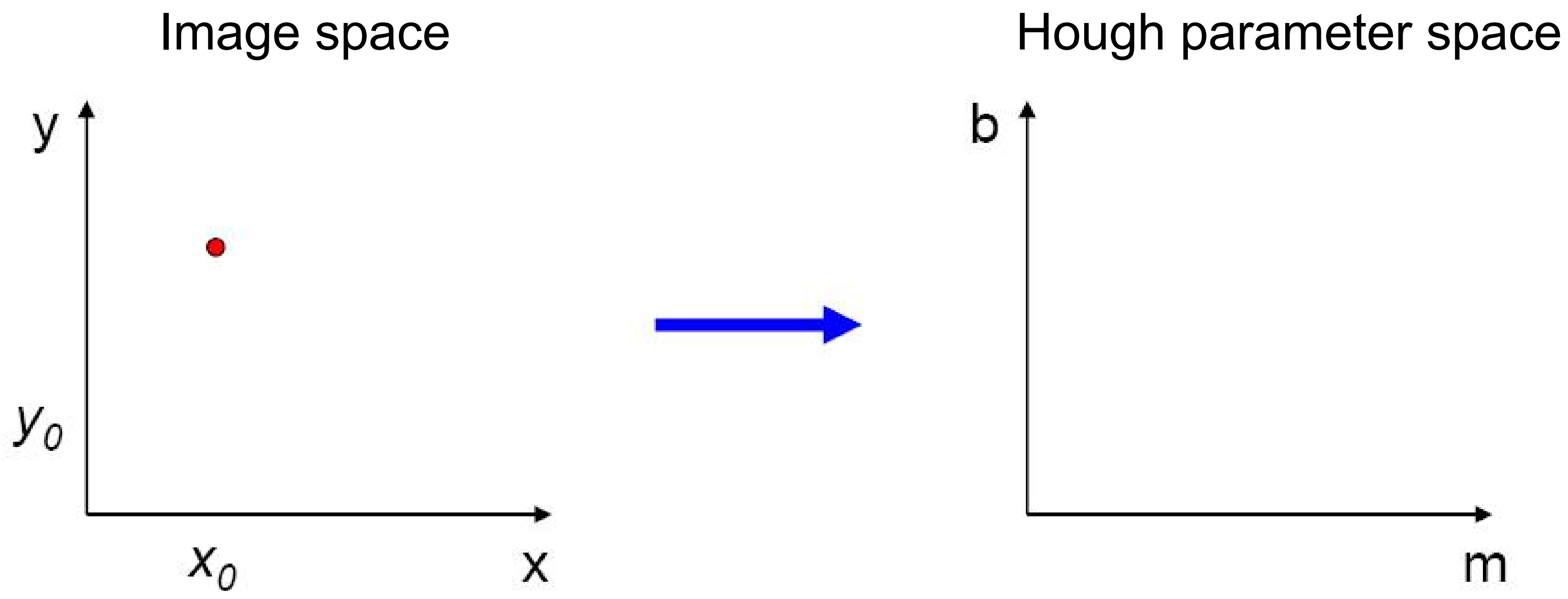
Parameter Space Representation

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



Parameter Space Representation

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



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

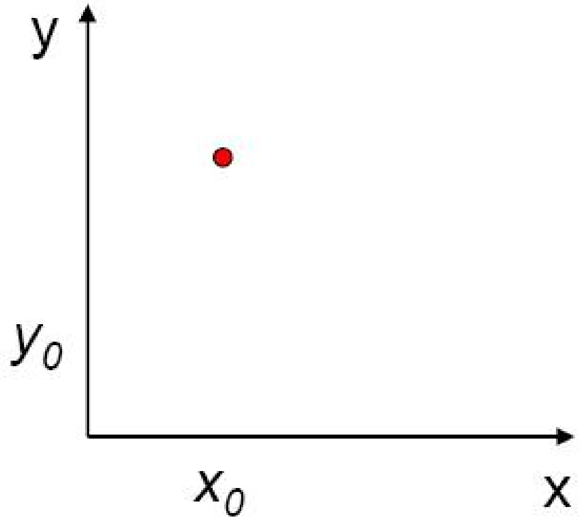
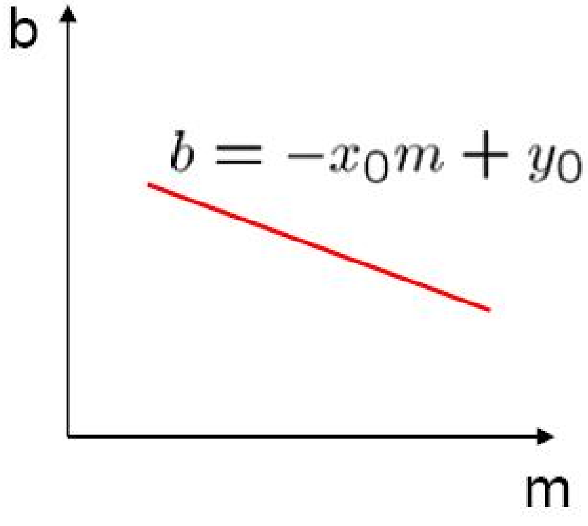


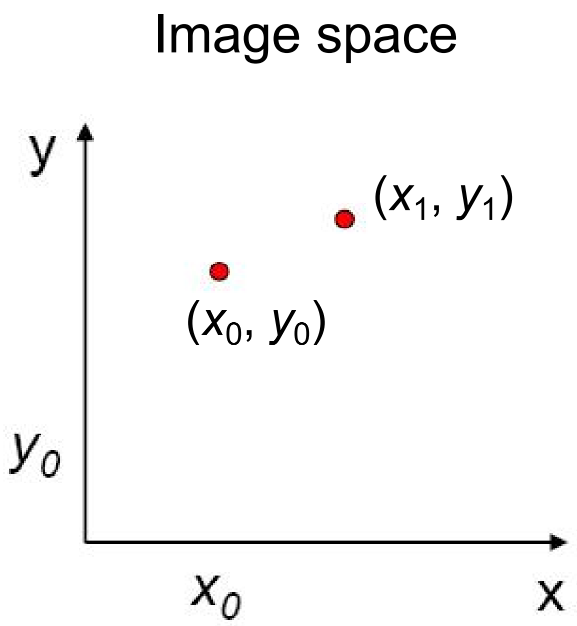
Image space

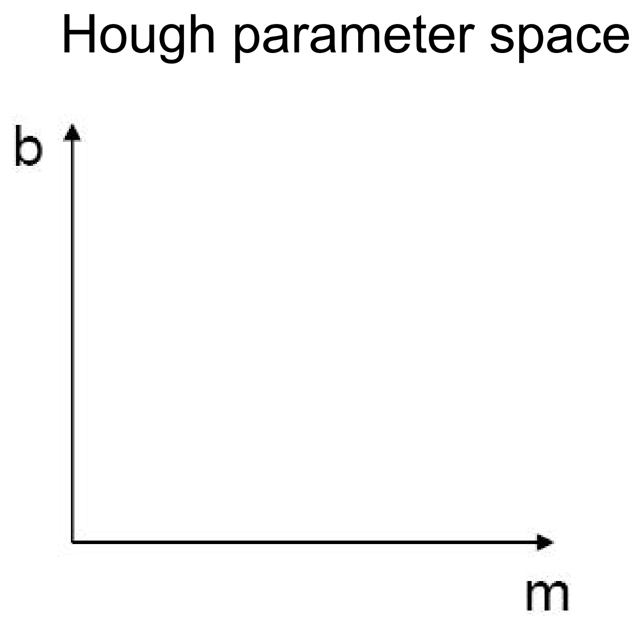


Hough parameter space

Parameter Space Representation

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



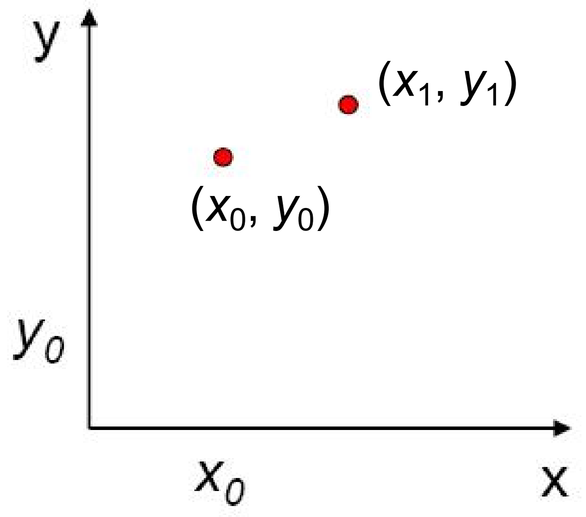


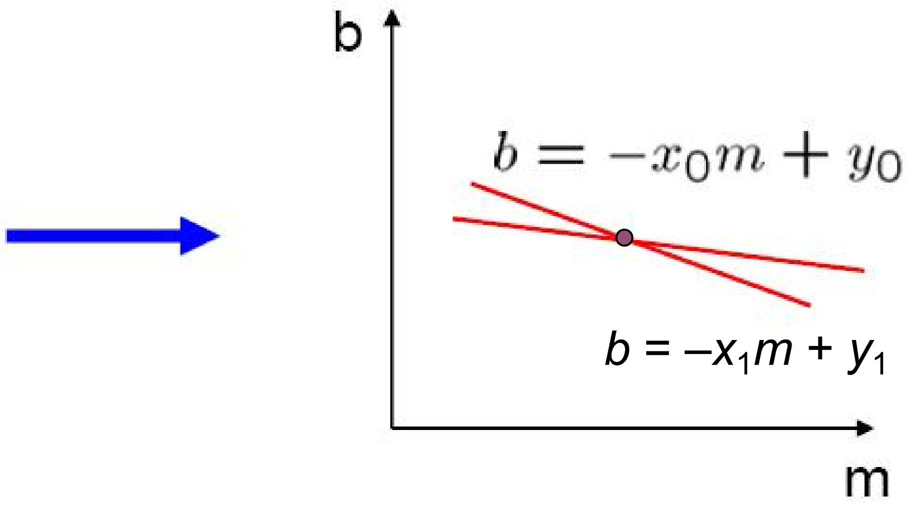
Parameter Space Representation

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

Image space

Hough parameter space

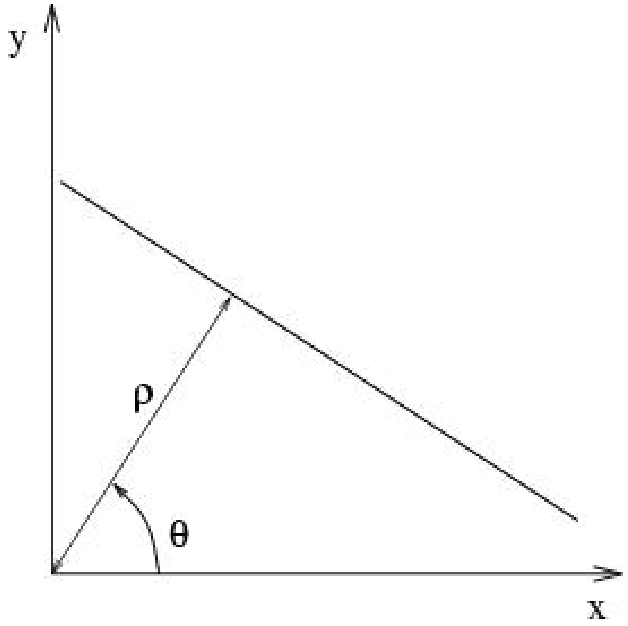




Parameter Space Representation

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

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



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

每个点会在 (θ,ρ) 参数空间中产生一个正弦曲线

Algorithm Outline

- Initialize accumulator H to all zeros
- For **each** edge point (x,y) in the image

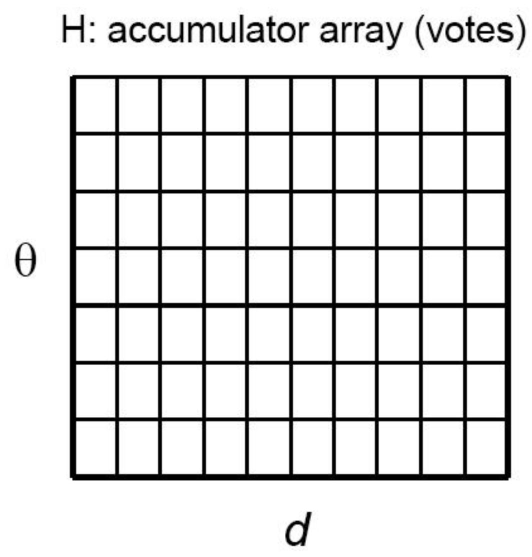
  For $\theta = 0$ to 180

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

  $H(\theta, \rho) = H(\theta, \rho) + 1$

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

每个图像空间中的点会在霍夫空间中产生一条正弦曲线。

Basic Illustration

- A line

Horizontal axis is θ
Vertical is rho.

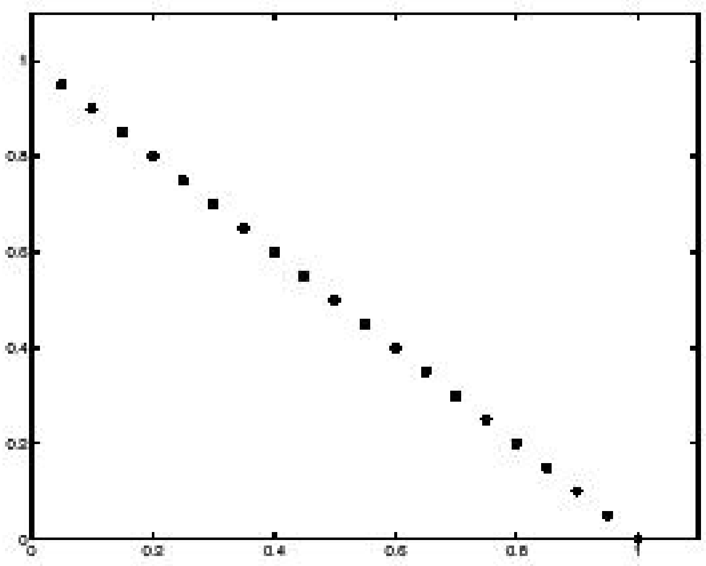
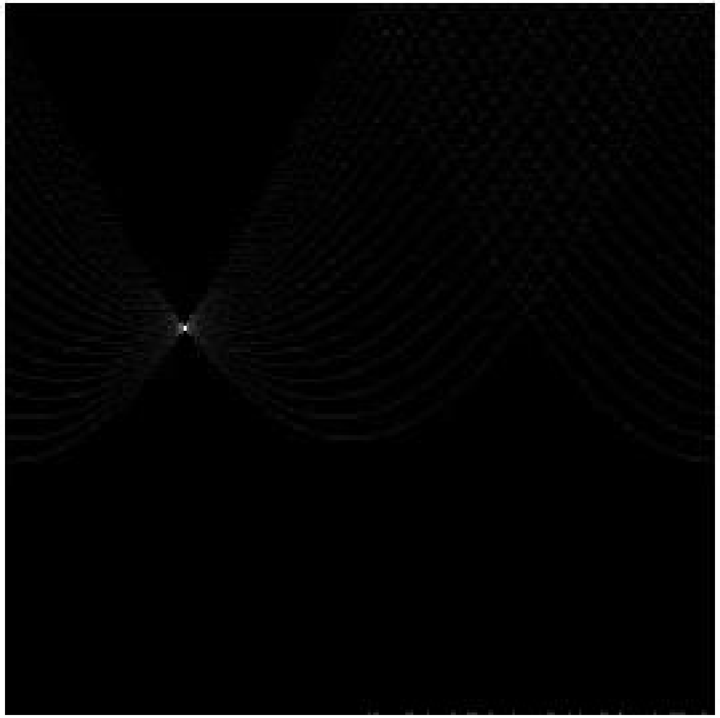


Image space



Votes

Basic Illustration

- A line with noise

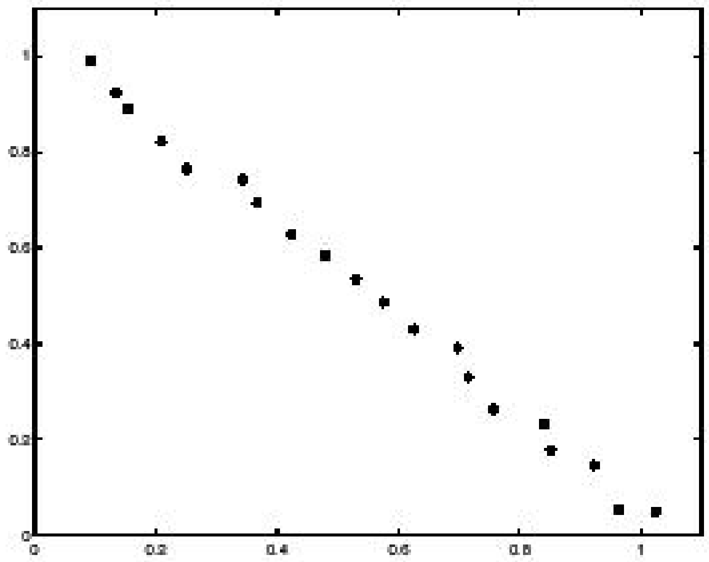
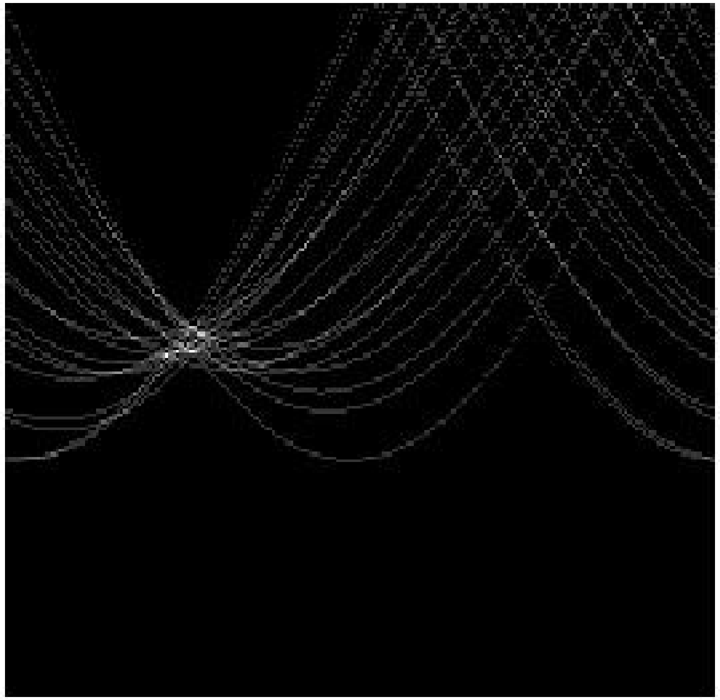


Image space



Votes

Basic Illustration

- Scattered points

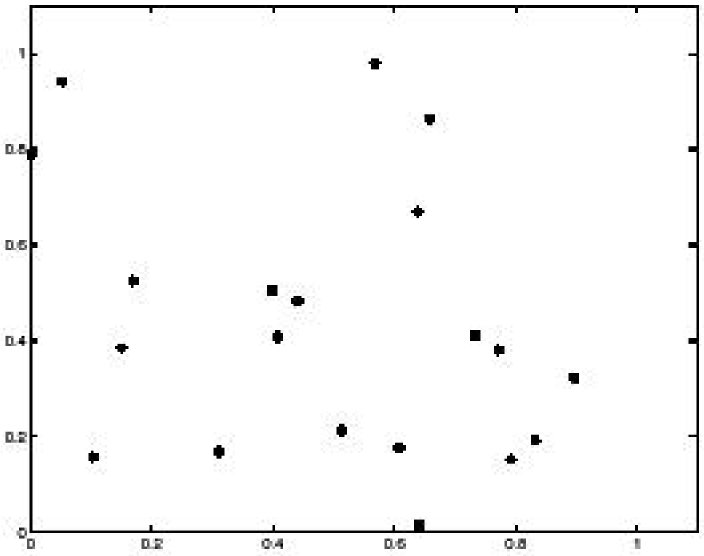
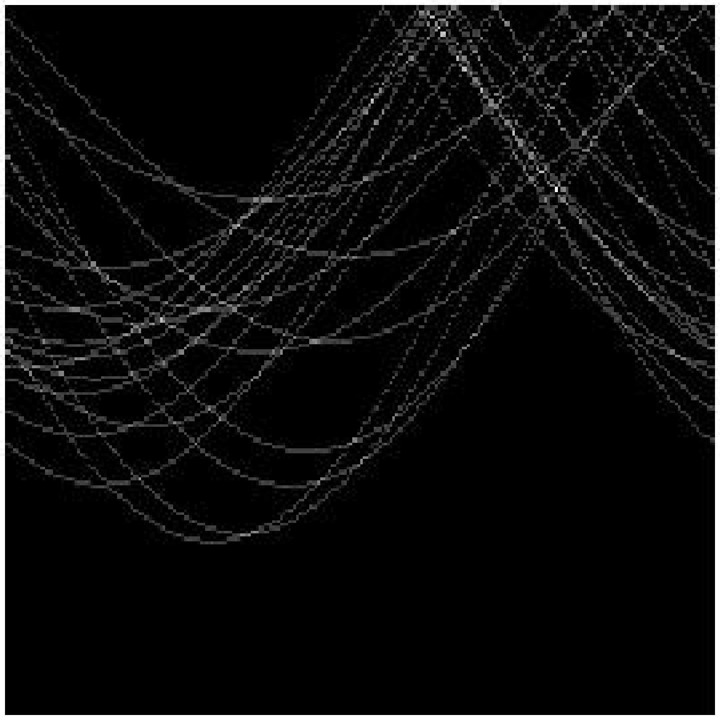


Image space



Votes

Mechanics of the Hough transform

如果单元格太大，可能会将不同的直线合并为一条线。

如果单元格太小，噪声会导致遗漏一些真正的直线，干扰结果。

- Difficulties
  - How big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)
- How many lines?
  - Count the peaks in the Hough array
  - Treat adjacent peaks as a single peak
- Which points belong to each line?
  - Search for points close to the line
  - Solve again for line and iterate

霍夫变换的结果是在参数空间中形成一个数组，其中直线对应的区域会形成局部最大值（峰值）。

对相邻的峰值进行处理：将相邻的峰值视为同一条直线。这样可以避免将由于噪声或细小差异引起的多个峰值误认为是不同的直线。

寻找接近直线的点：

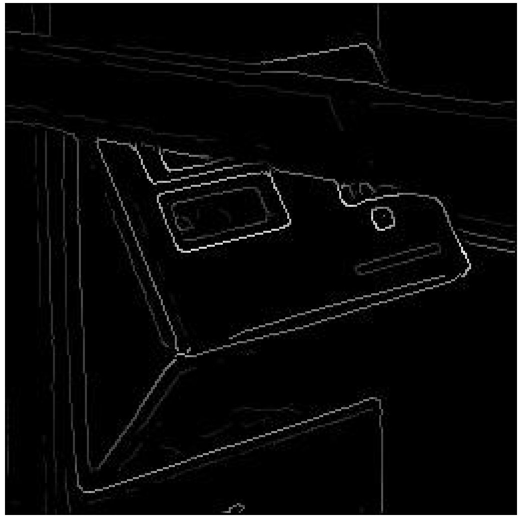
通过计算图像中哪些点与检测到的直线接近，将这些点归属于该直线。

再次解决问题：

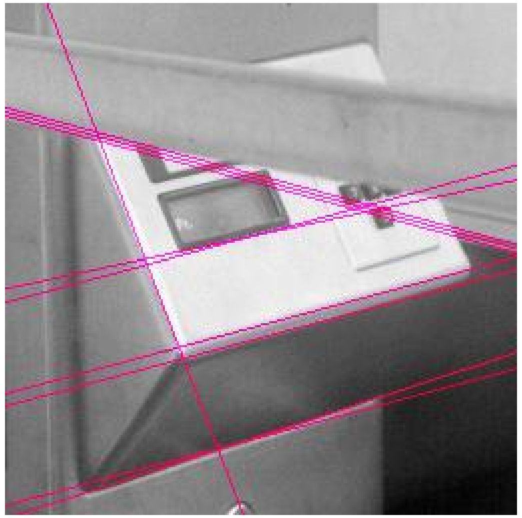
对于每条找到的直线，重新执行霍夫变换来精确确定直线的参数，并进行迭代，直到结果稳定。

Real World Example


Original


Edge Detection

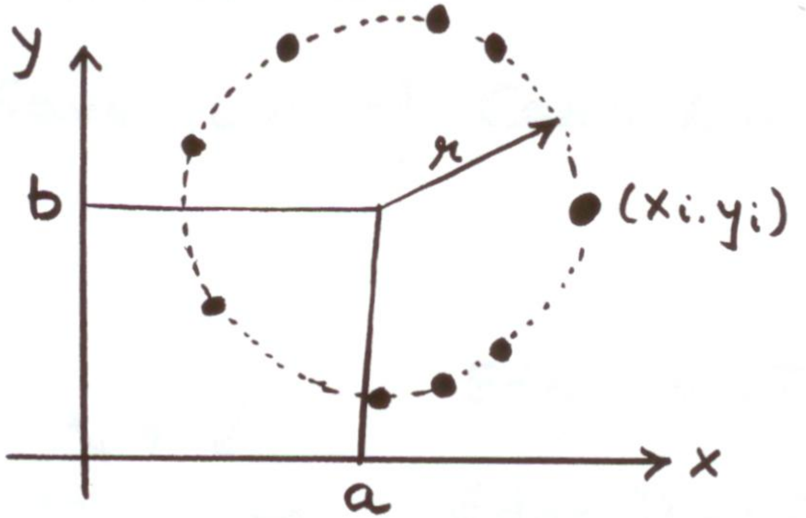

Parameter Space


Found Lines

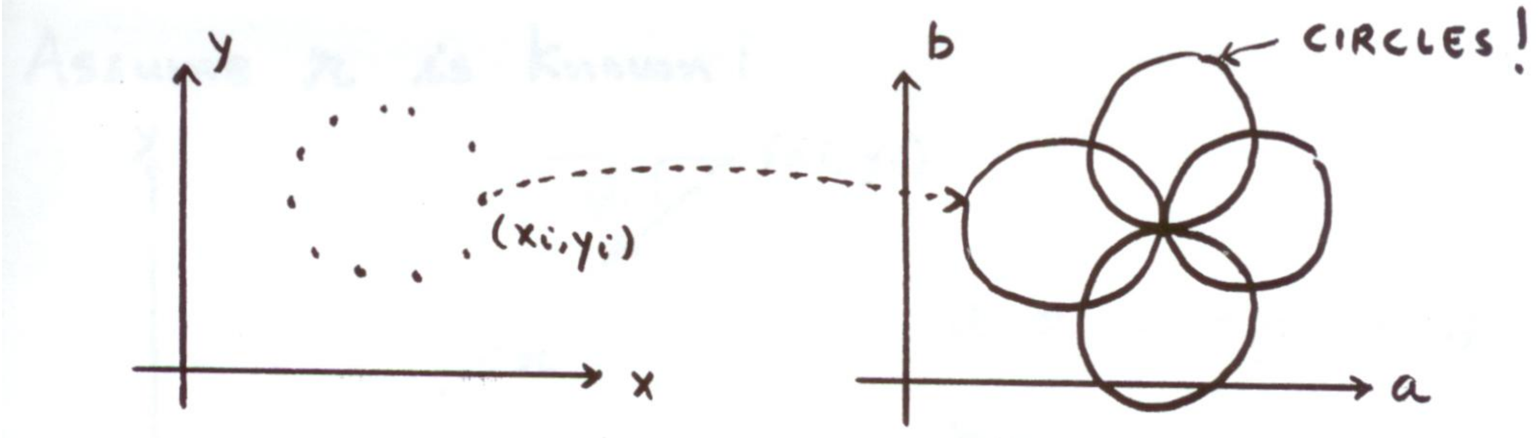
Finding Circles by Hough Transform

- Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$



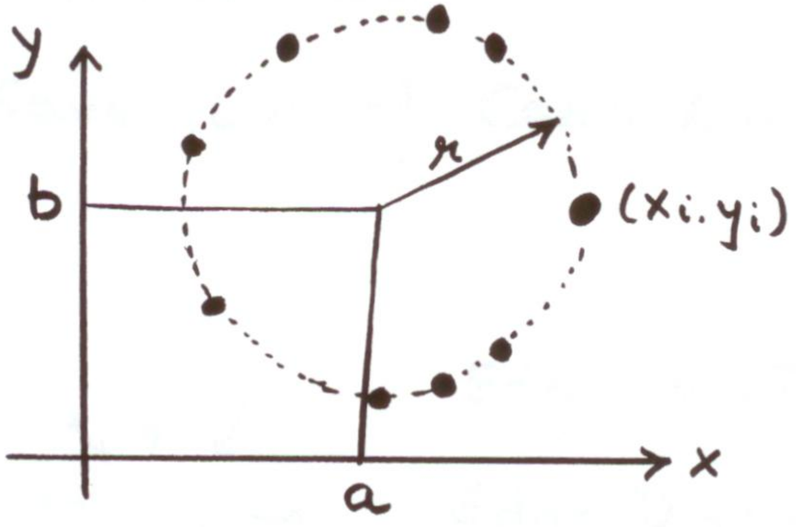
- If radius is known:
  - 2D Hough Space
- Accumulator Array: $A(a,b)$



Finding Circles by Hough Transform

- Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$



- If radius is not known:
  - ➢ 3D Hough space!
- Use Accumulator array: $A(a, b, r)$

- What is the surface in the Hough space?

Finding Circles by Hough Transform

- Hough transform for circles

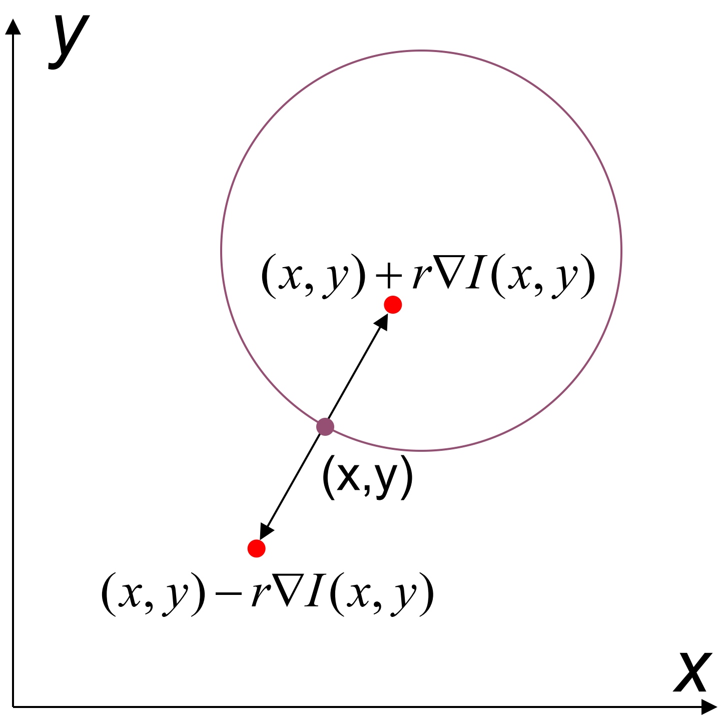
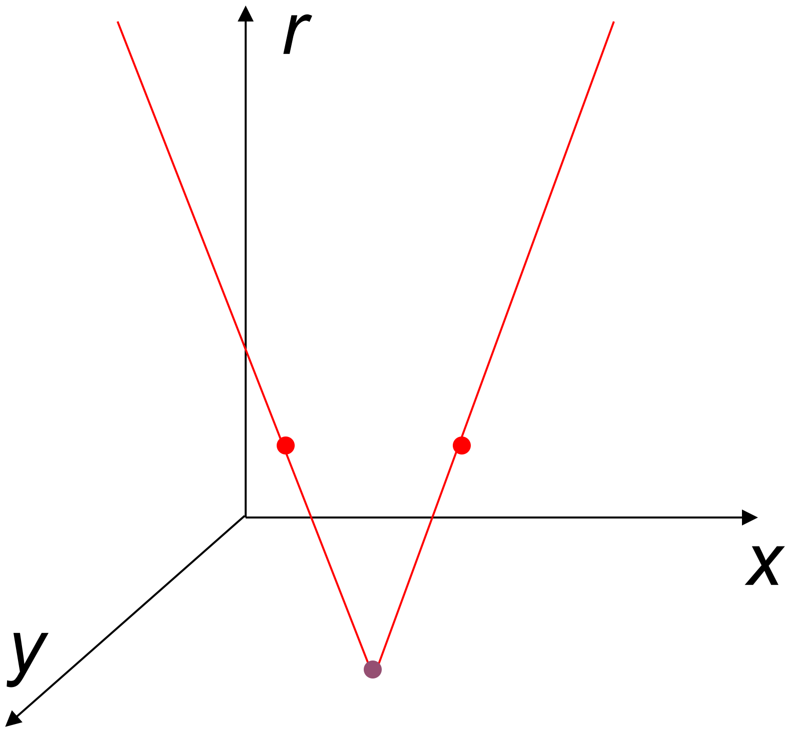


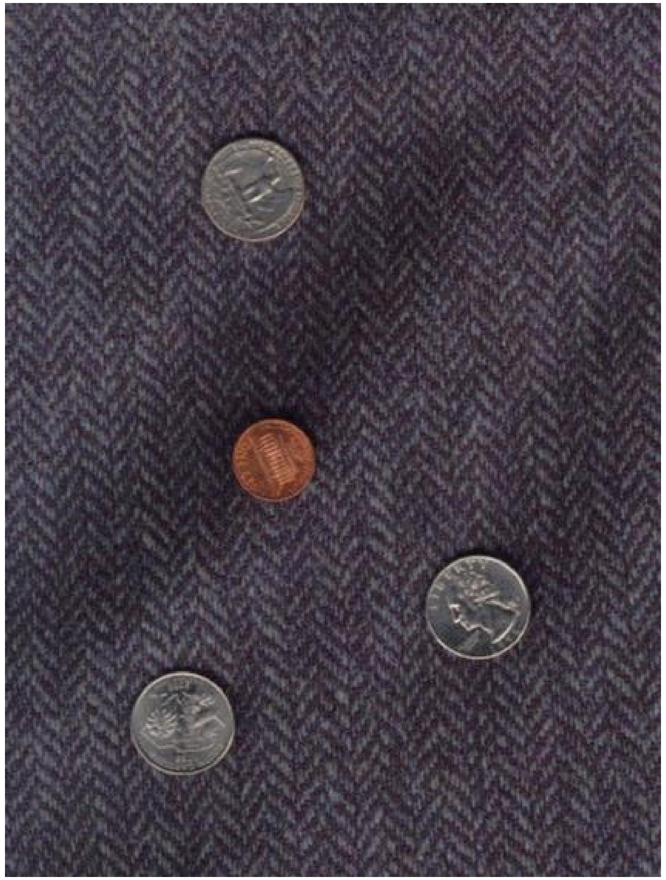
Image space



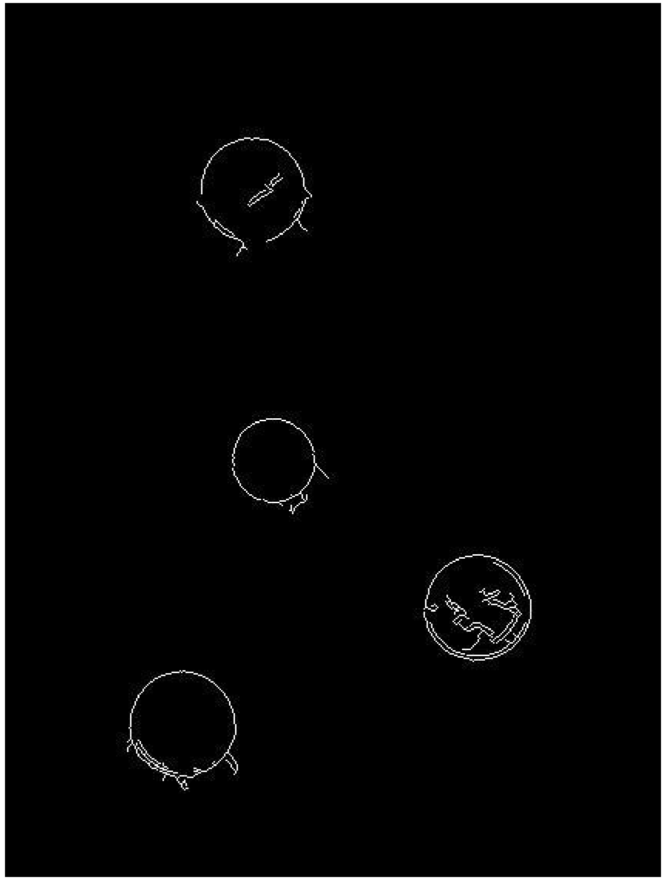
Hough parameter space

Finding Coins

Original



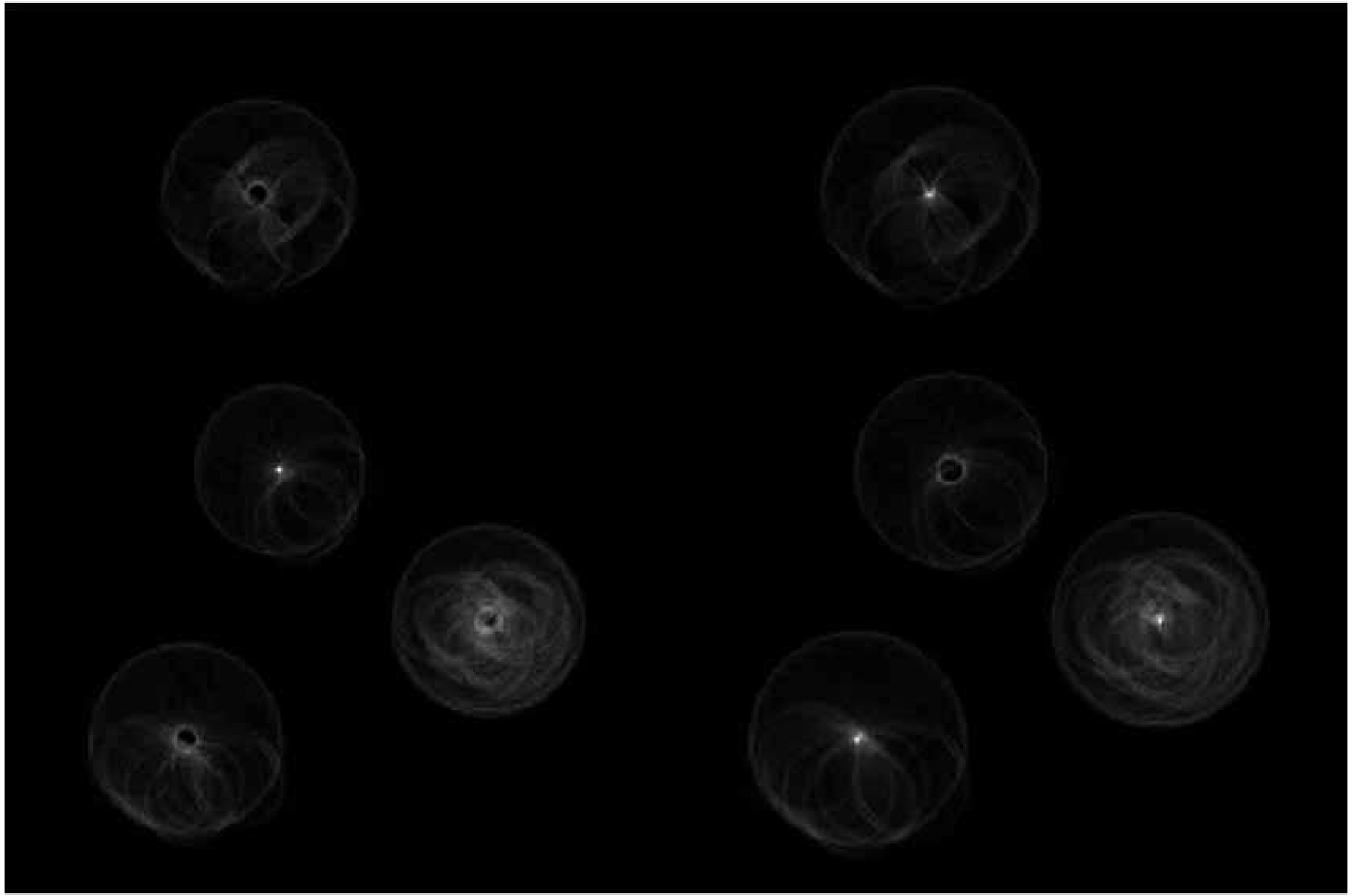
Edges (note noise)



Finding Coins

- Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size

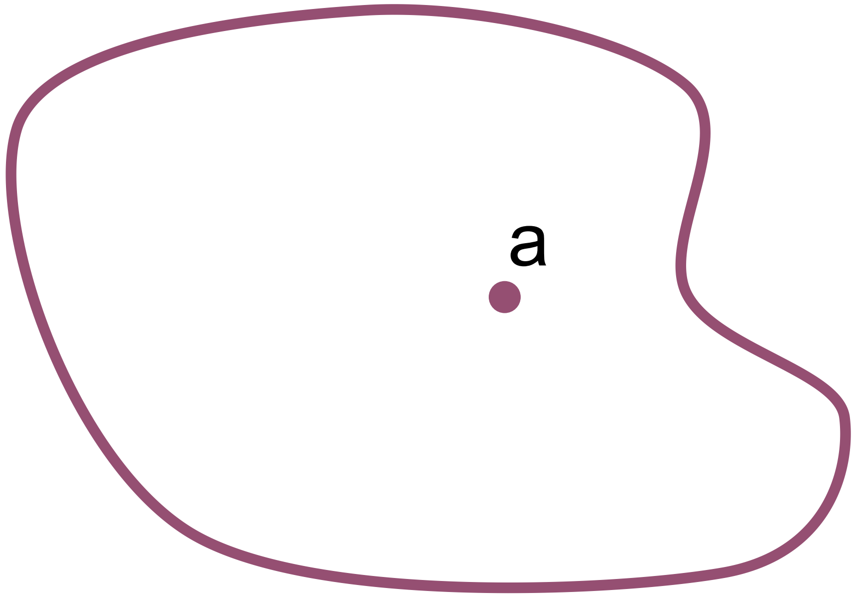
Penny Quarters



Generalized Hough Transform

- We want to find a fixed shape **(known)** defined by its boundary points and a reference point

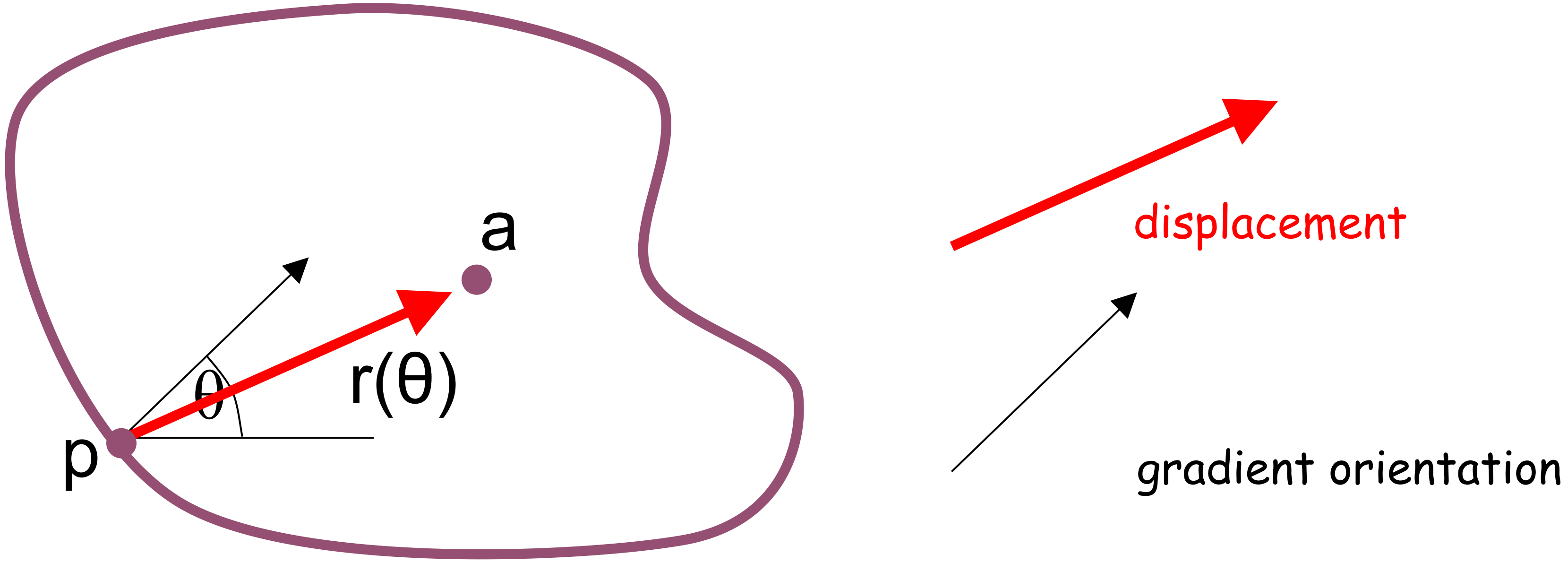
检测一个已知的固定形状，这个形状是通过其边界点和一个参考点来定义的。



D. Ballard, [Generalizing the Hough Transform to Detect Arbitrary Shapes](), PR 13(2), 1981, pp. 111-122.

Generalized Hough Transform

- We want to find a fixed shape (known) defined by its boundary points and a reference point
- For every boundary point p, we can compute the displacement vector $r = a - p$ as a function of gradient orientation θ

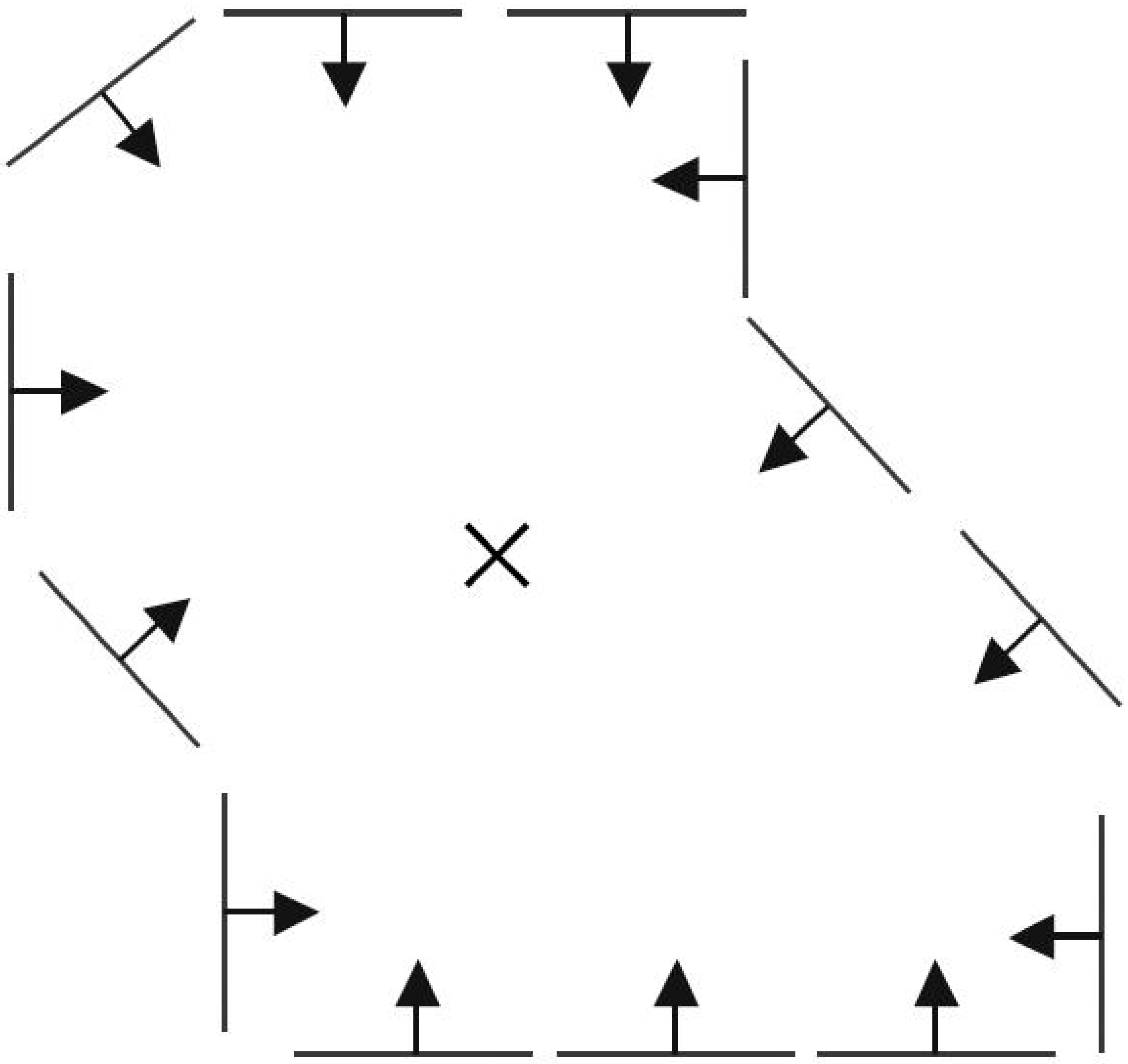


Generalized Hough Transform

- Construct a model for a shape:
  - ➢ Construct a table indexed by θ storing displacement vectors r as function of gradient direction
- Detect using the model
  - ➢ For each edge point p with gradient orientation θ:
    - ✓ Retrieve all r indexed with θ
    - ✓ For each $r(\theta)$, put a vote in the Hough space at $p + r(\theta)$
  - ➢ Peak in this Hough space is reference point with most supporting edges
- Assumption: translation is the only transformation here, i.e., orientation and scale are fixed

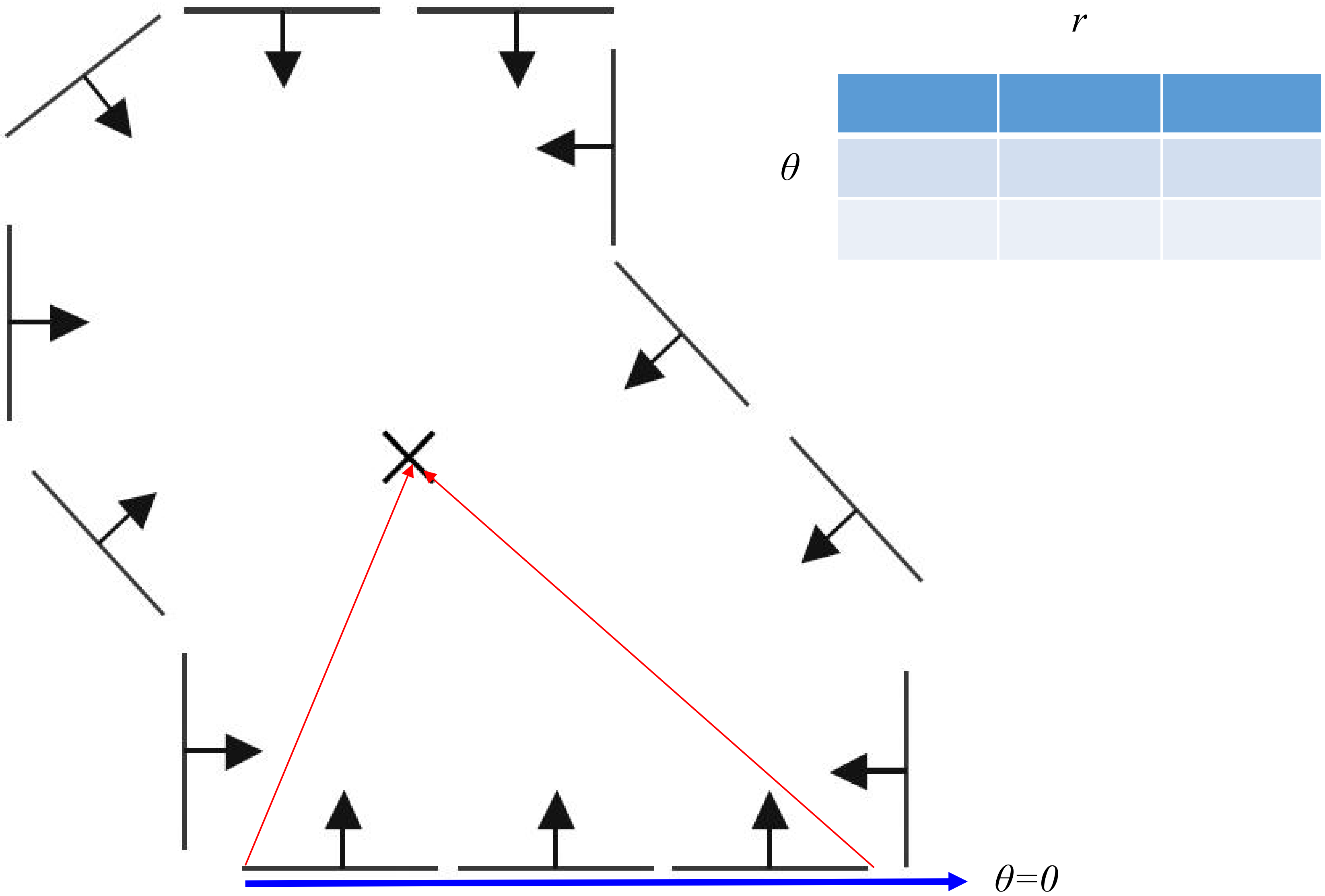
Example: a Known and Fixed Shape

- Model shape
  - Gradient orientation
  - No rotation



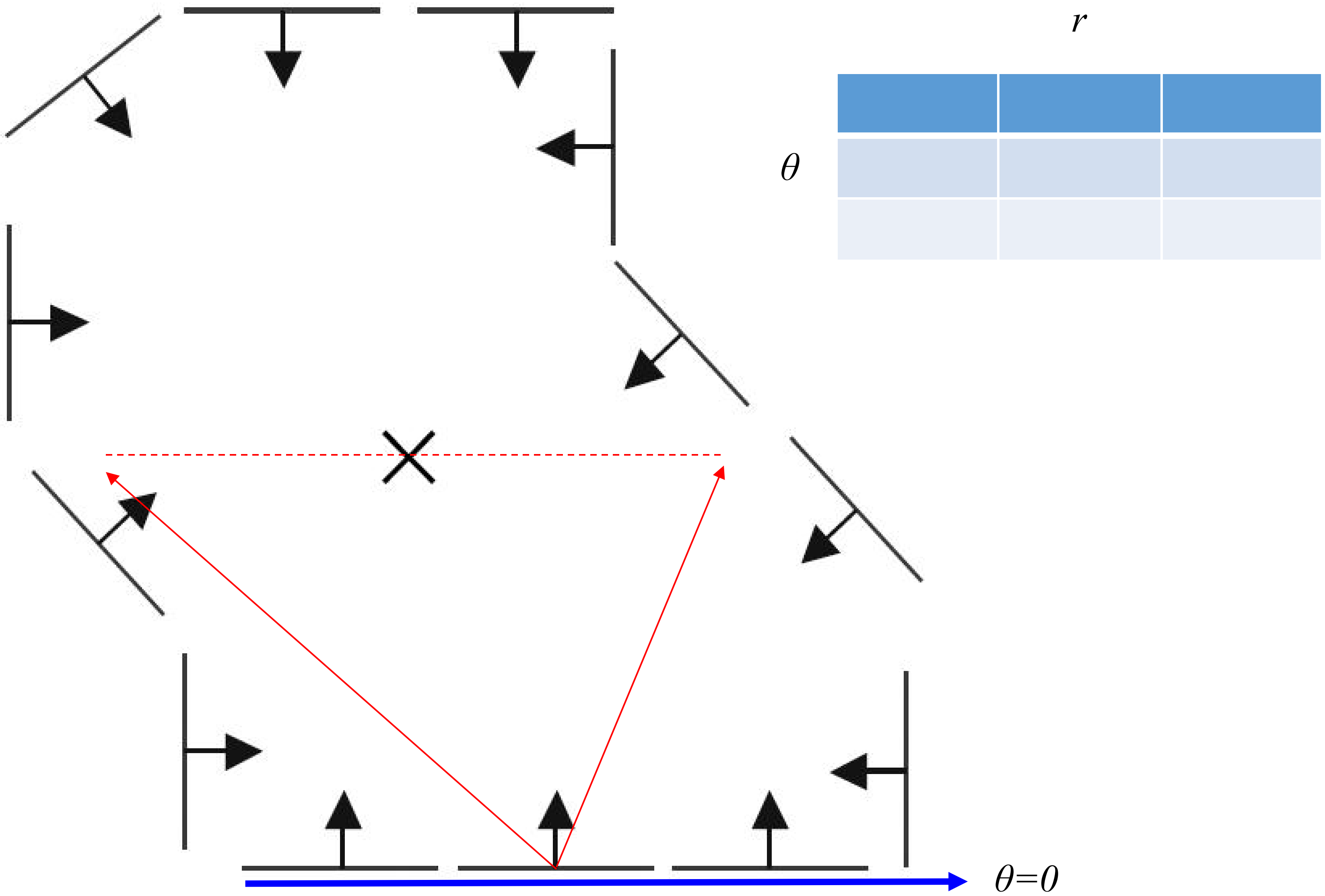
Example: Building a Table

- Displacement vectors for model points



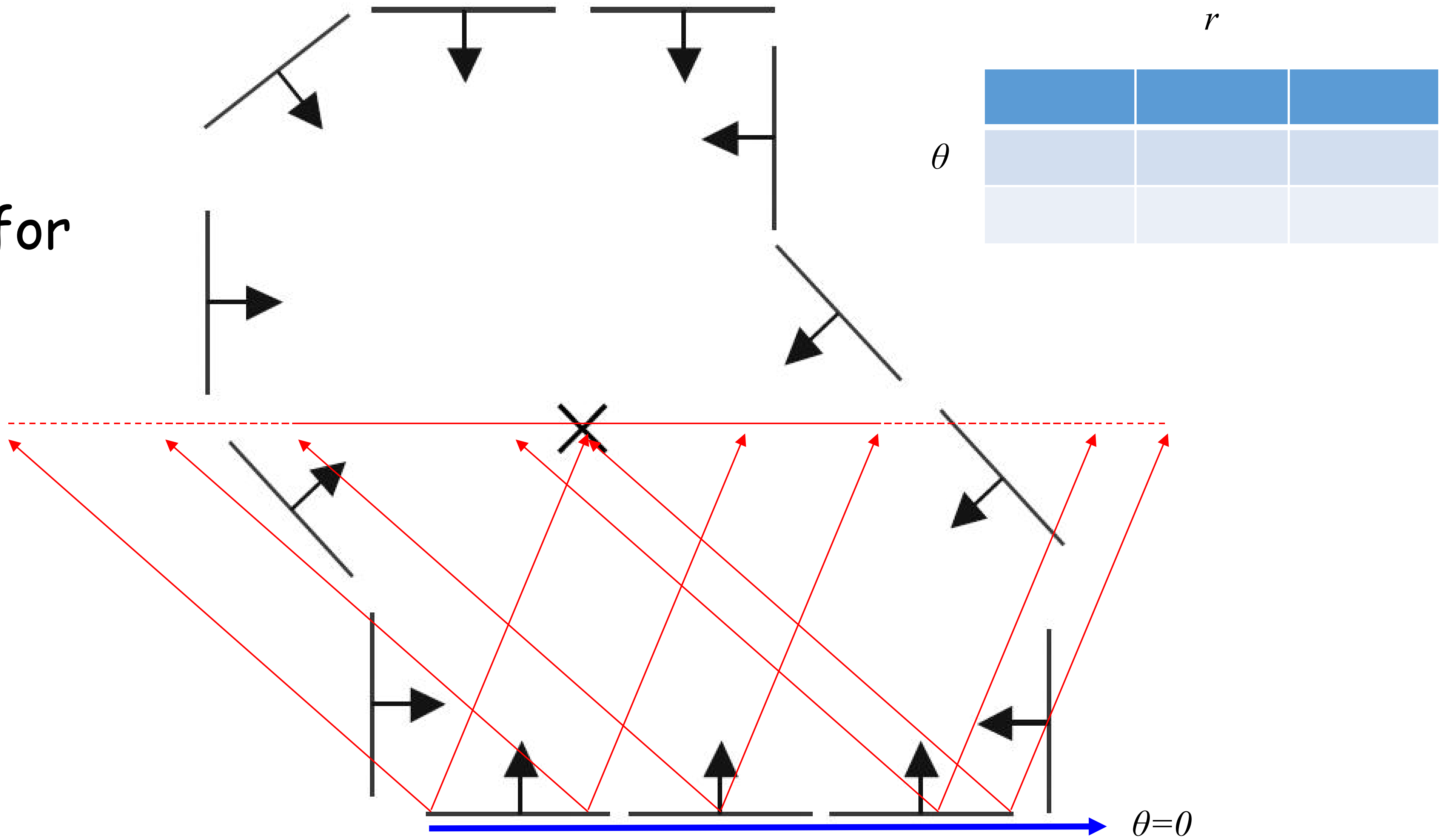
Example: Detection

- Range of voting locations for **test** point



Example: Detection

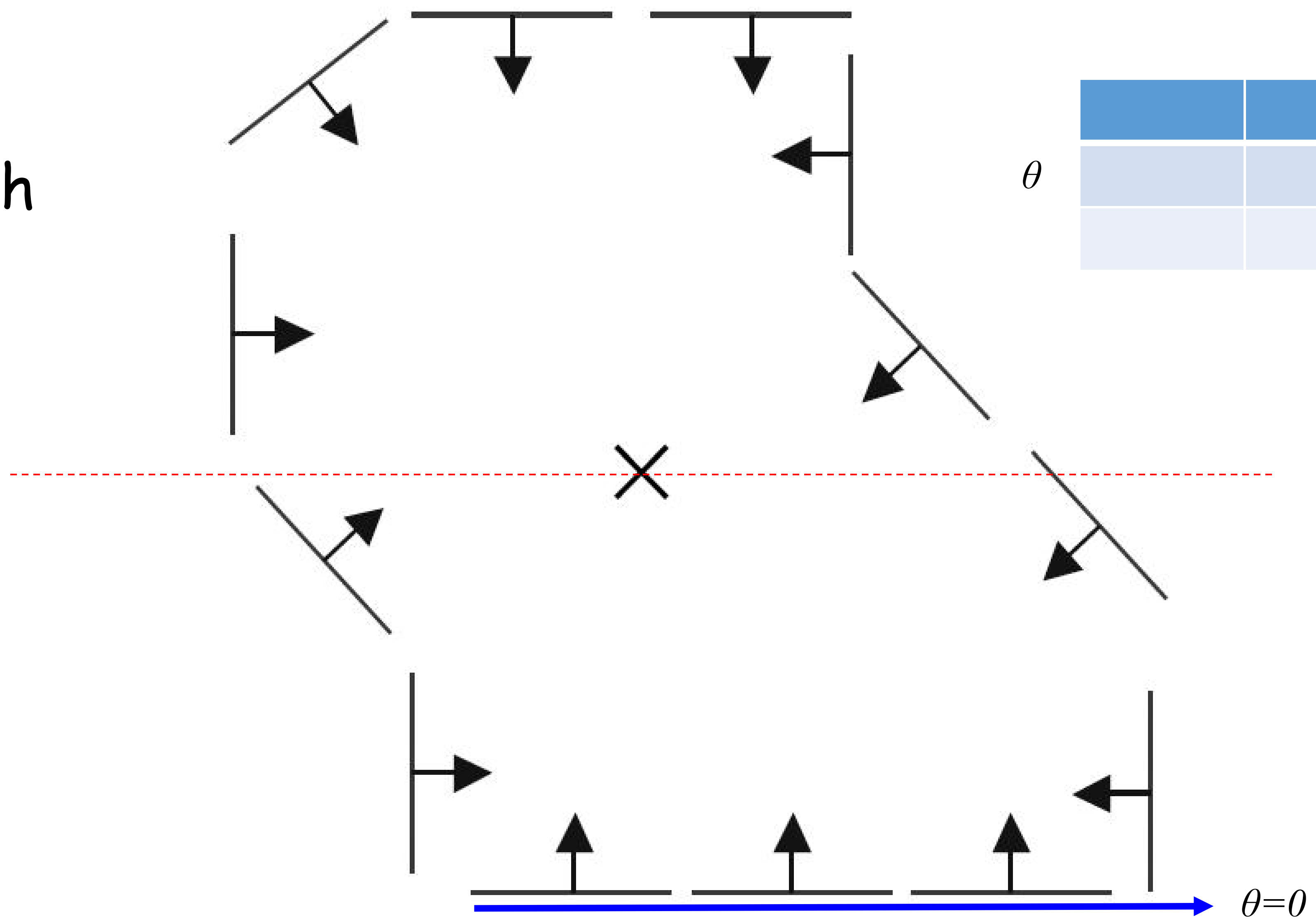
- Range of voting locations for test point



Example: Detection

- Votes for points with

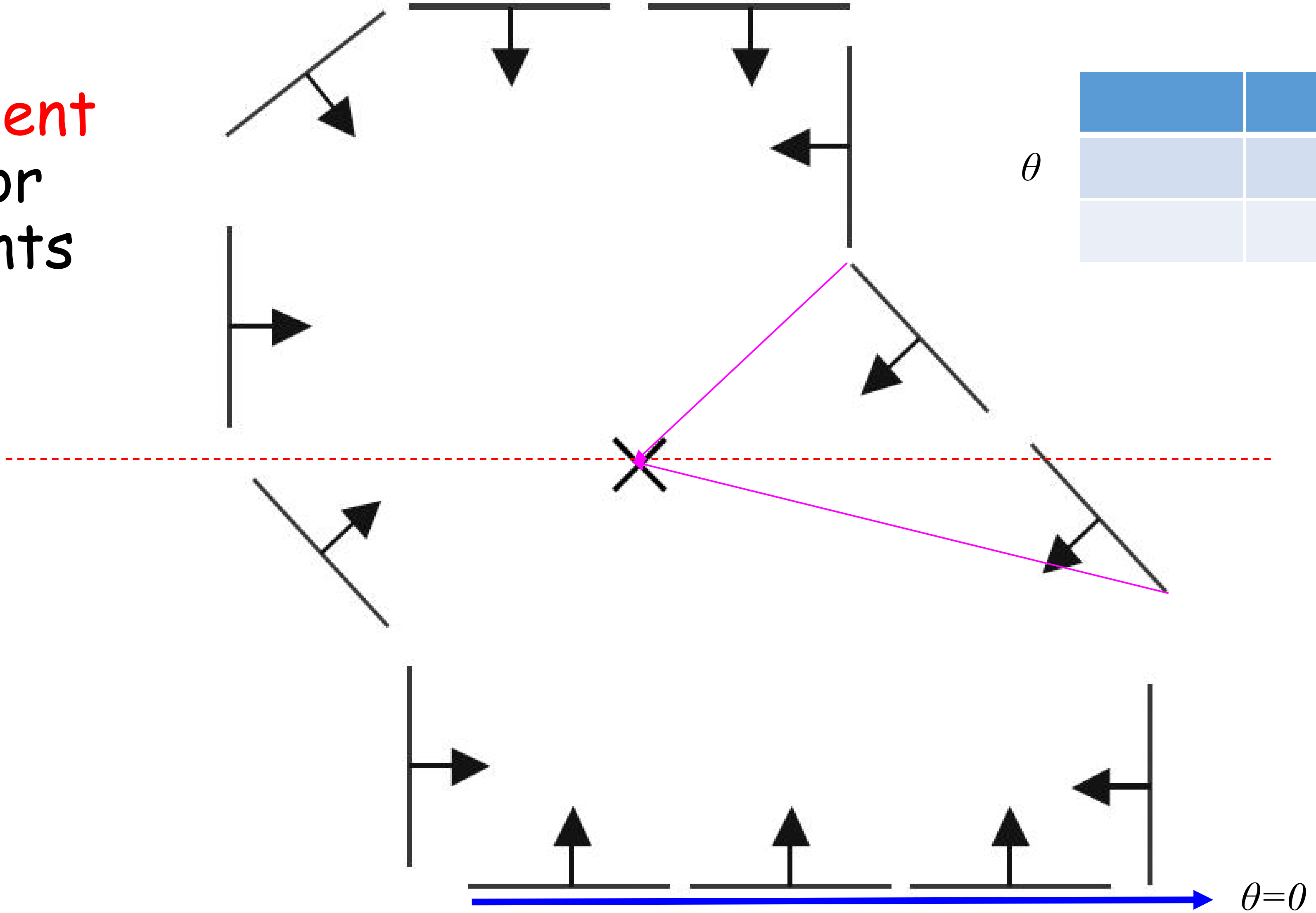
  $\theta = \uparrow$



		r	
θ			

Example: Building a Table

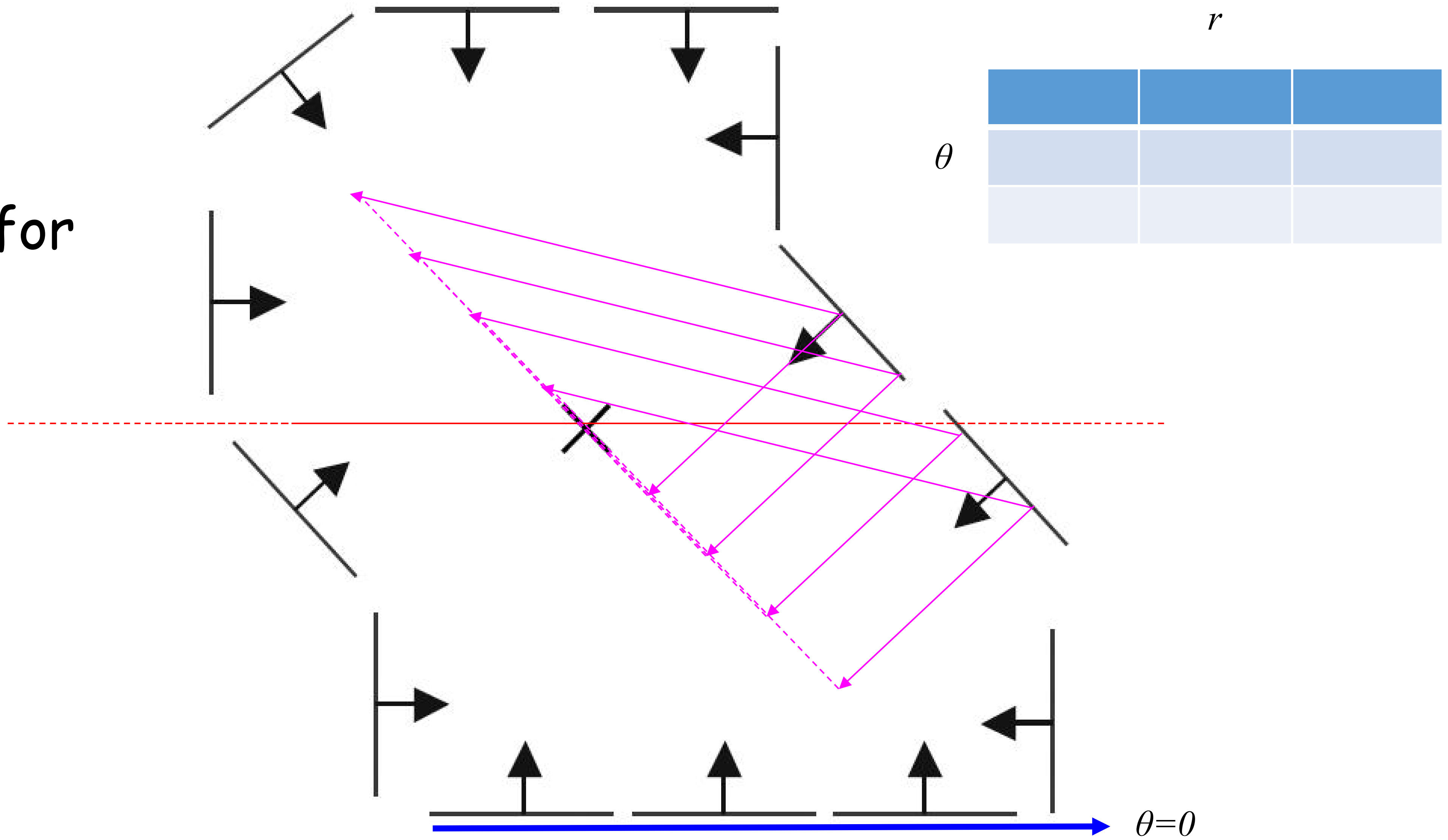
- Displacement vectors for model points



		r	
θ			

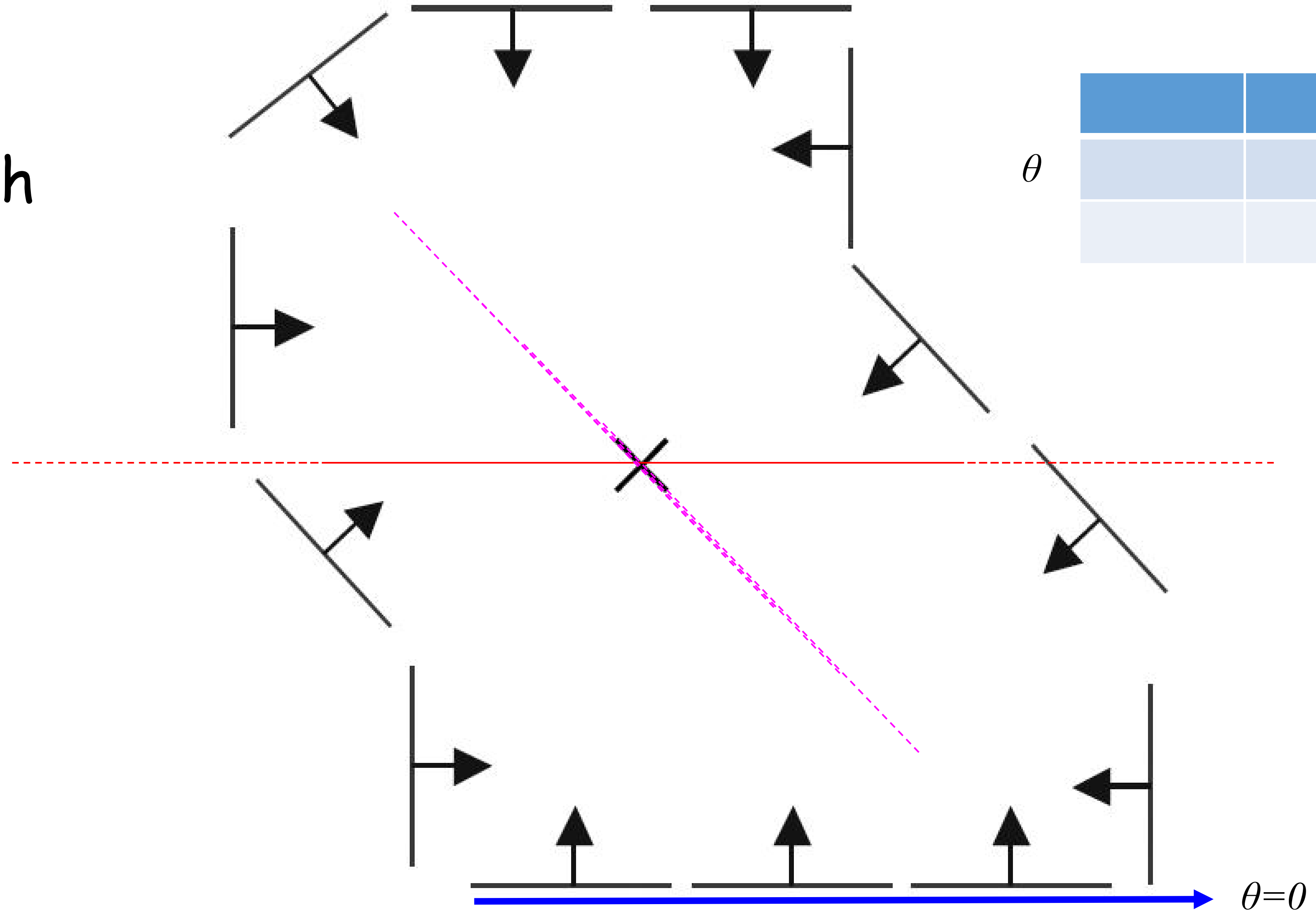
Example: Detection

- Range of voting locations for test point



Example: Detection

- Votes for points with $\theta = \swarrow$

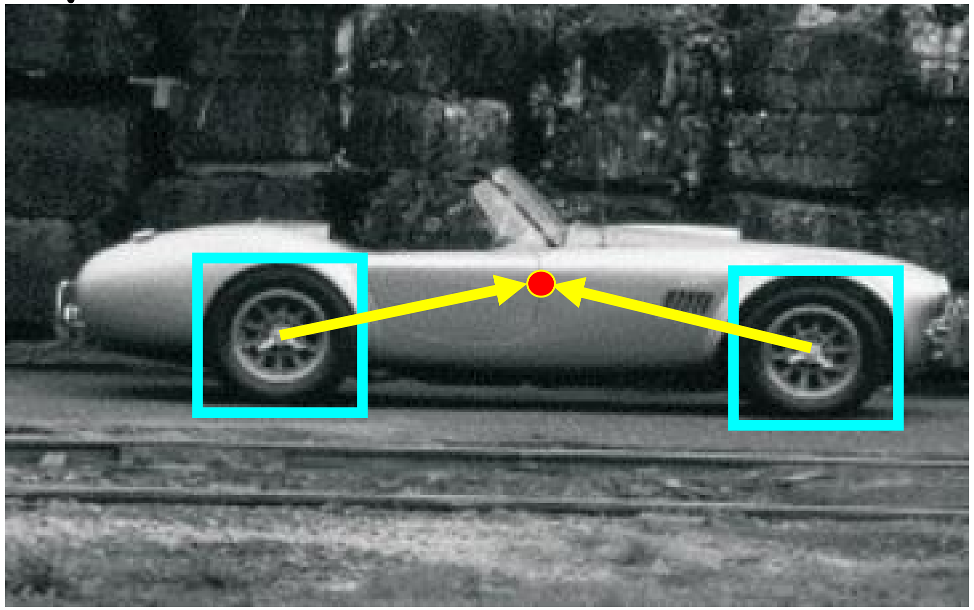


θ \ r			
θ			

Application in Recognition

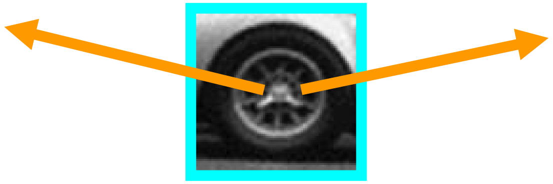
每个特征点通过一个“代码词”（codeword）进行表示，这个代码词可能包含多个特征信息，例如该区域的局部特征描述符或位移向量等。

- Instead of indexing displacements by gradient orientation, index by “visual codeword”



training image

What is the codeword?



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

Application in Recognition

- Instead of indexing displacements by gradient orientation, index 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

Image Alignment

Image Alignment

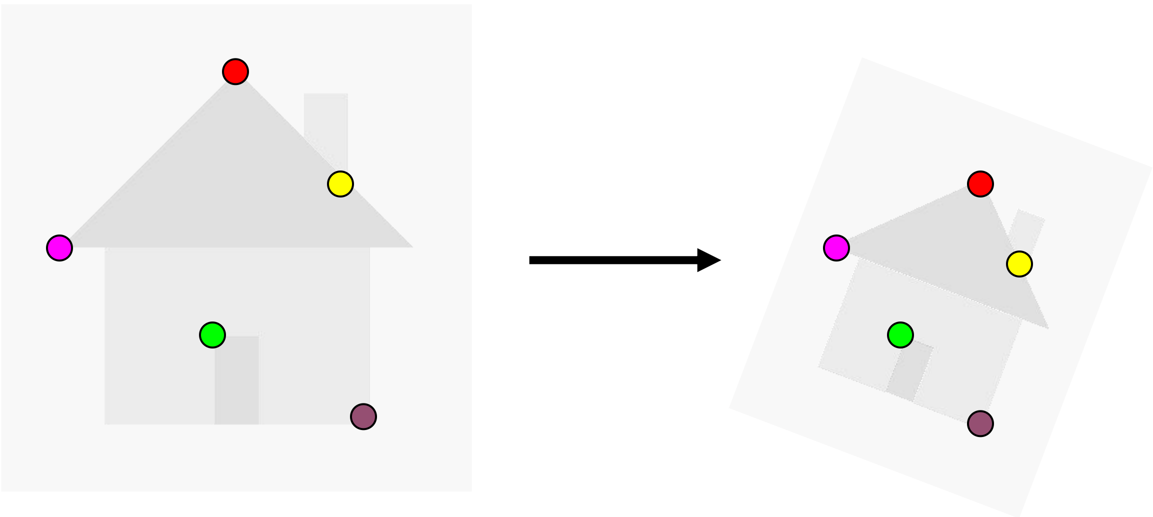
- Two broad approaches:

  通过搜索大多数像素都一致的位置来对齐两张图像。简单来说，直接配准是通过比较图像中的像素来找出最佳对齐位置。

  - ➢ Direct (pixel-based) alignment
    - ✓ Search for alignment where most pixels agree
  - ➢ Feature-based alignment

    通过提取图像的特征点并确保它们的一致性来进行配准。特征点可以是边缘、角点、纹理等，而不仅仅是单纯的像素。

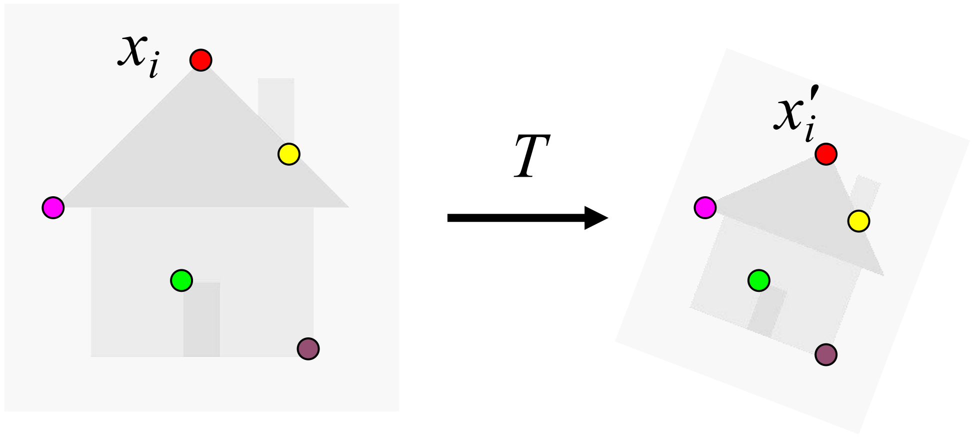
    - ✓ Search for alignment where *extracted features* agree
    - ✓ Can be verified using pixel-based alignment



Alignment as Fitting

配准过程可以看作是将一个模型与两个图像中匹配的特征之间的变换进行拟合。

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



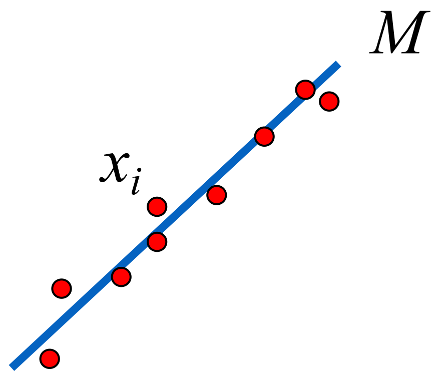
Find transformation T that minimizes

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

Alignment as Fitting

- Previously: fitting a model to features in one image

将模型与单一图像中的特征进行拟合



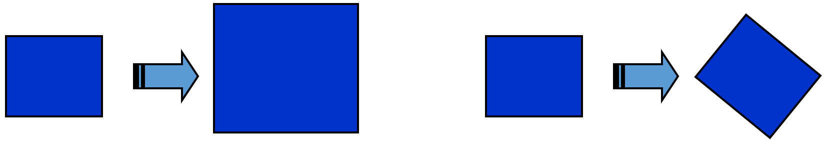
Find model M that minimizes

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

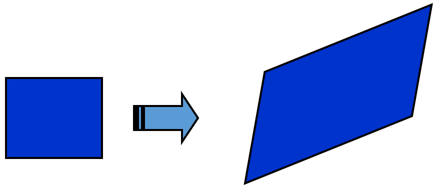
例如，拟合一个直线模型 M 到图像中的特征点。通过最小化每个点到拟合模型的残差，可以找到最佳的直线模型 M。

2D Transformation Models

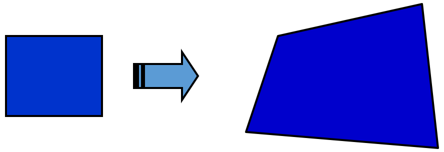
- Similarity (translation, scale, rotation)



- Affine



- Projective (homography)



单应性变换

单应性变换可以将一个图像中的平面（例如，地面或墙面上的一个物体）转换为另一个视角下的图像中的平面。简而言之，单应性变换是描述一个图像到另一个图像的几何变换关系，特别是当相机视角发生变化时。

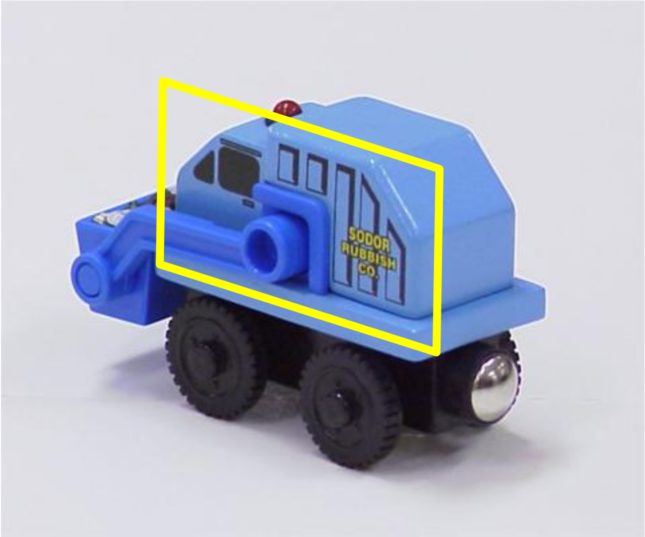
Affine Transformations

1. 仿射变换概述：
简单拟合过程：仿射变换是一种线性最小二乘拟合方法，用于通过图像中的对应点来计算变换矩阵。

视角变化的近似：仿射变换能够近似平面物体的视角变化，尤其适用于大致平面的物体和大致正交的相机（即没有透视变形的相机）。

用于初始化更复杂模型的拟合：仿射变换是处理更复杂形状或变换模型的起点，可以作为更复杂模型拟合的初始步骤。

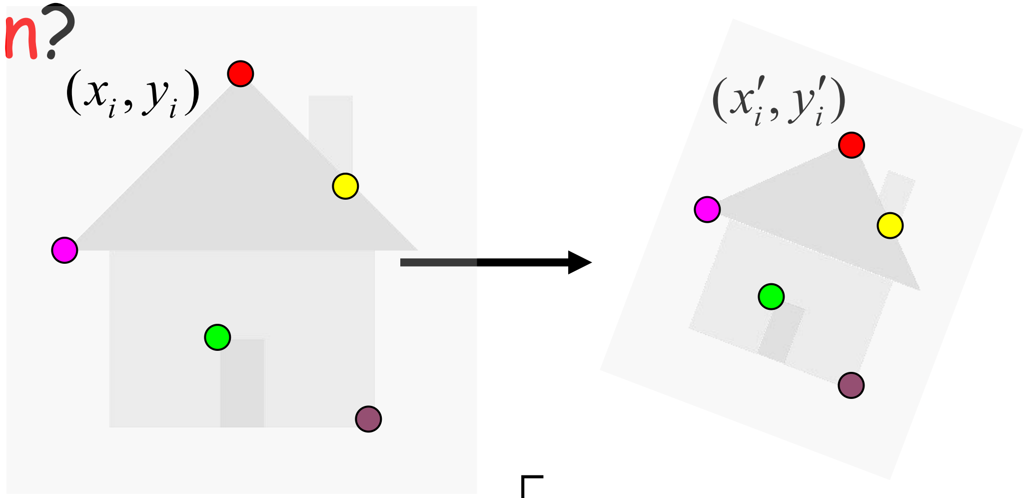
- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used to initialize fitting for more complex models



Affine Transformations

已经知道了两个图像中的对应点，怎么找仿射变换

- Assume we know the correspondences (???), how do we get the transformation?



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

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

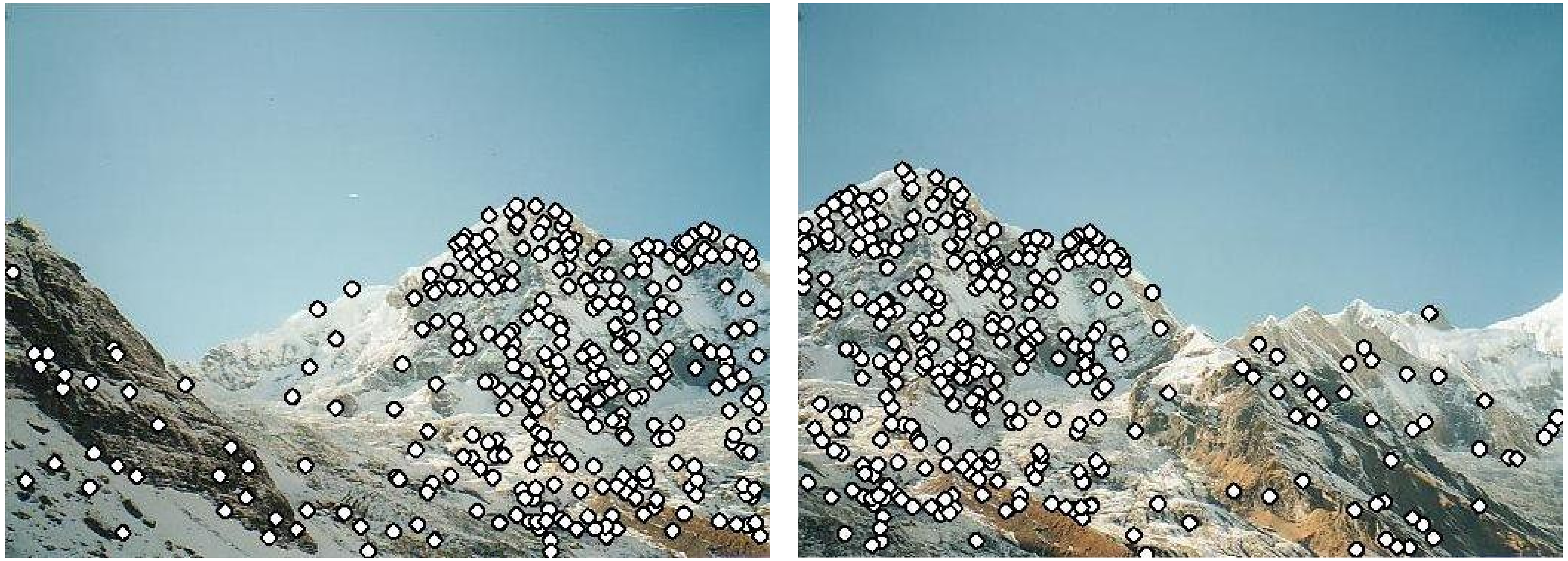
Affine Transformations

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters

仿射变换涉及六个未知数，包括旋转、缩放、平移等参数，通常通过一组匹配点来计算这些未知数。

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

Feature-Based Alignment Outline

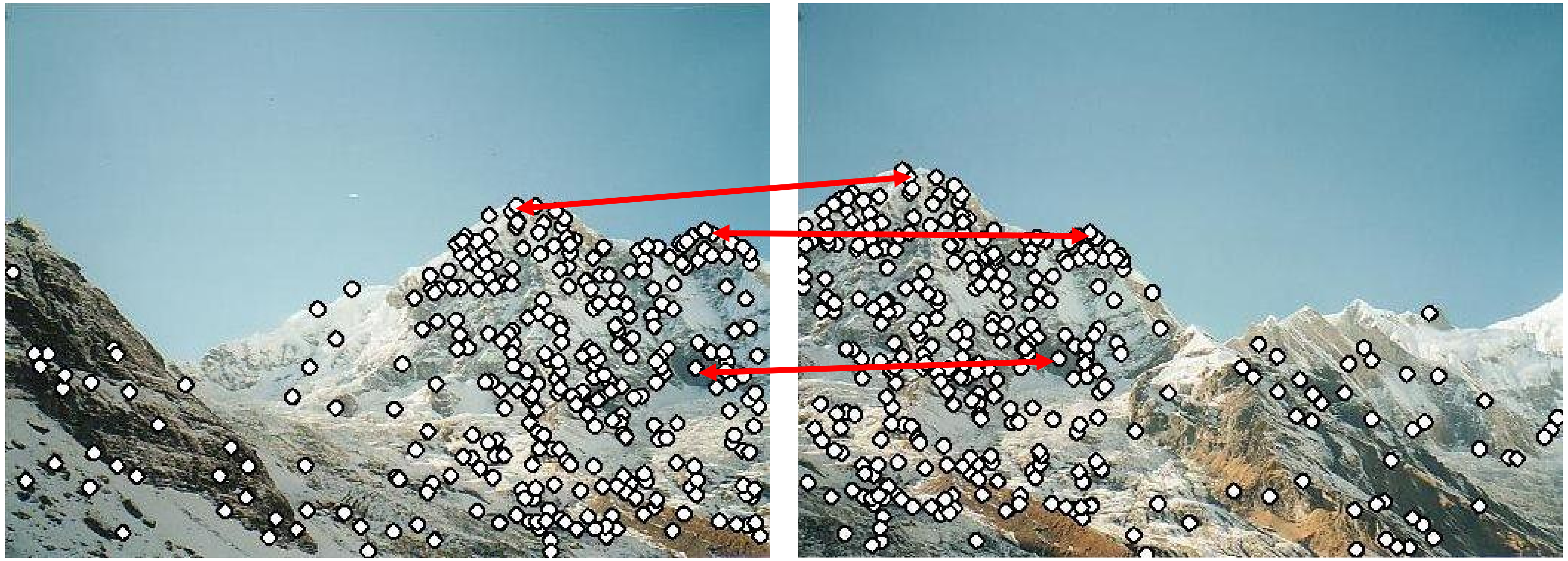


- Extract features

Feature-Based Alignment Outline

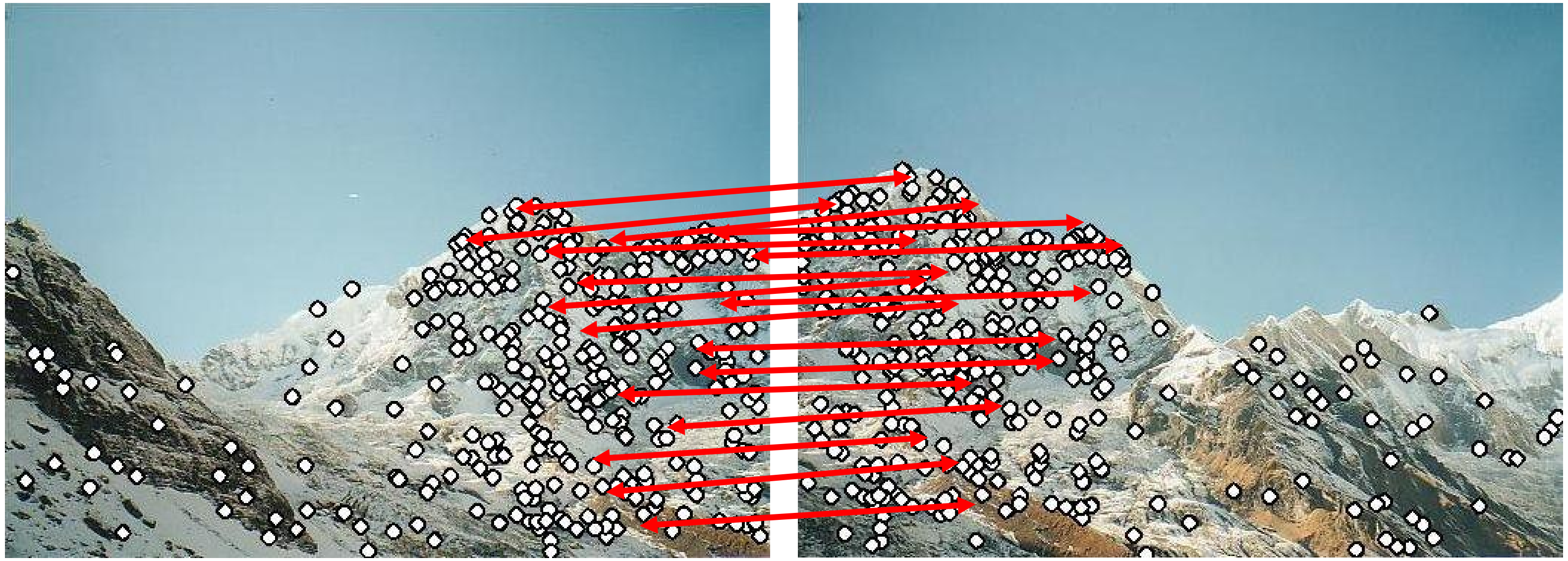
- Extract features
- Compute *putative matches*

Feature-Based Alignment Outline



- Extract features
- Compute *putative matches*
- Loop:
  - ➢ *Hypothesize* transformation *T*

Feature-Based Alignment Outline



- Extract features
- Compute *putative* matches
- Loop:
  - ➢ *Hypothesize* transformation *T*
  - ➢ *Verify* transformation (search for other matches consistent with *T*)

我们计算所有特征点的匹配对。这些匹配对称为“潜在匹配”，它们可能是正确的匹配，也可能包含错误的匹配。

Feature-Based Alignment Outline

- Extract features
- Compute *putative matches*
- Loop:
  - ➢ *Hypothesize* transformation *T*
  - ➢ *Verify* transformation (search for other matches consistent with *T*)

Conclusions

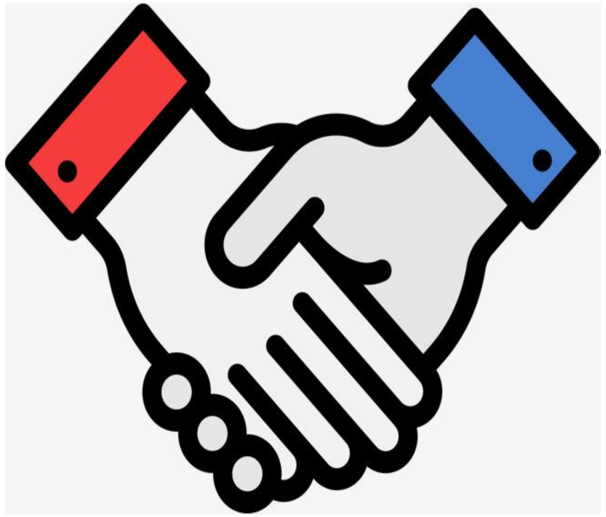
Conclusion

- Fitting techniques
  - Least Squares
  - Total Least Squares

- RANSAC

- Hough Voting

- Alignment as a fitting problem



Thanks

zhengf@sustc.edu.cn