计算机视觉

邬向前

计算学部

多模态智能及应用研究中心

电子邮箱: xqwu@hit.edu.cn

Fitting & Matching

How do we build panorama?

- We need to match (align) images

Matching with Features

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

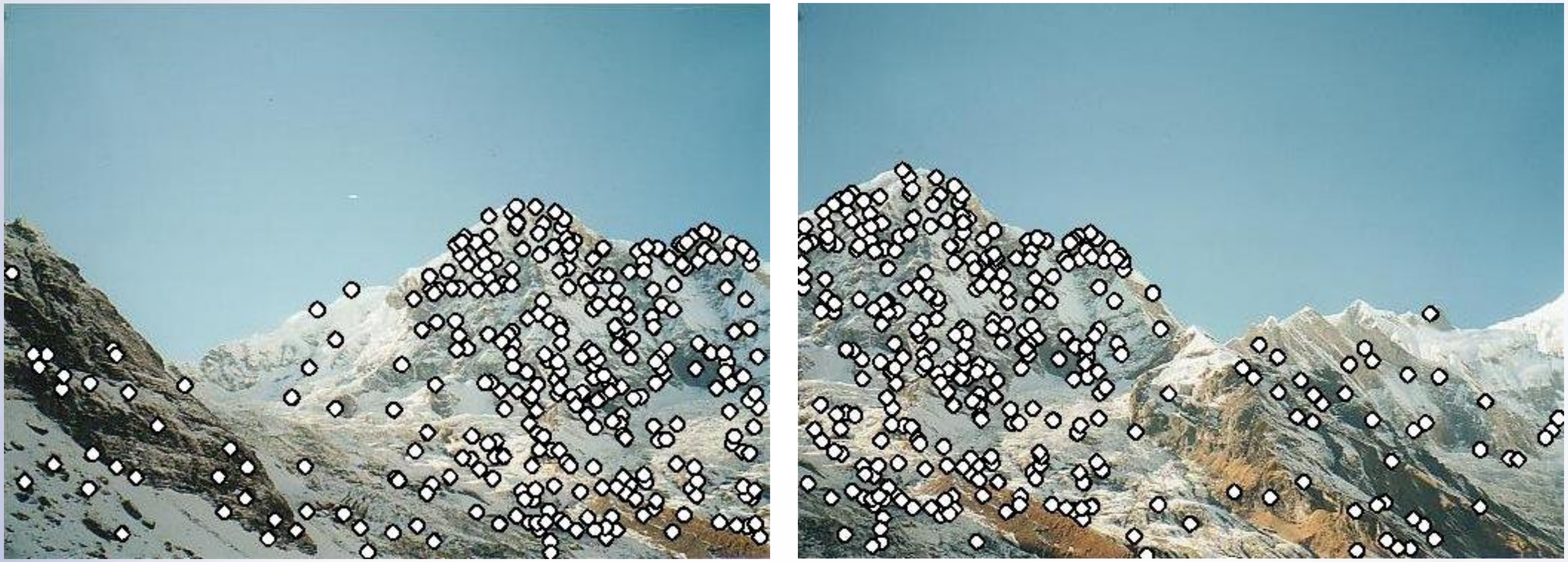
Matching with Features

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

Previous lecture

Scale Invariant Feature Transform(SIFT)

Scale Invariant Feature Transform(SIFT)



Scale Invariant Feature Transform(SIFT)

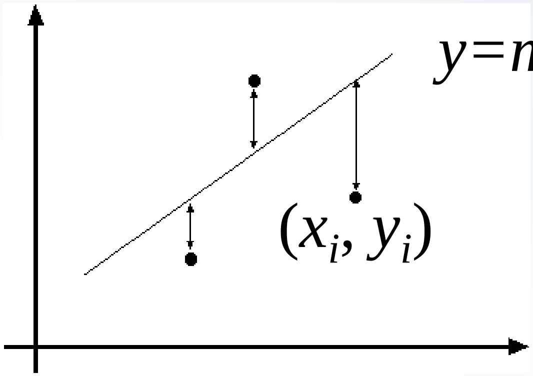
Overview

- Fitting techniques
  - Least Squares
  - Total Least Squares
  - Robust Fitting
- RANSAC
- Hough Voting
- Alignment as a fitting problem

Least squares line fitting

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

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



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

$X \in \mathbb{R}^{n\times 2}$
$B \in \mathbb{R}^{2\times 1}$
$Y \in \mathbb{R}^{n\times 1}$

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

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

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

Equation solution: $B = (X^T X)^{-1} X^T Y$

Slide: S. Lazebnik

Problem with “vertical” least squares

- 无法拟合垂直线，且由于误差采用的是垂直误差，导致越接近垂直线，拟合效果越差。
- 对噪声的鲁棒性不好，受噪声影响较大。

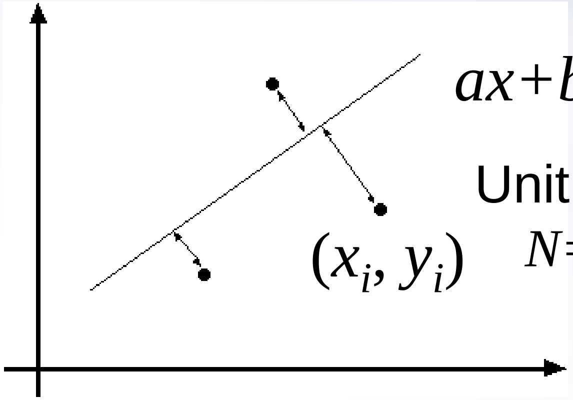
Overview

- Fitting techniques
  - Least Squares
  - Total Least Squares
  - Robust Fitting
- RANSAC
- Hough Voting
- Alignment as a fitting problem

Total least squares

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

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



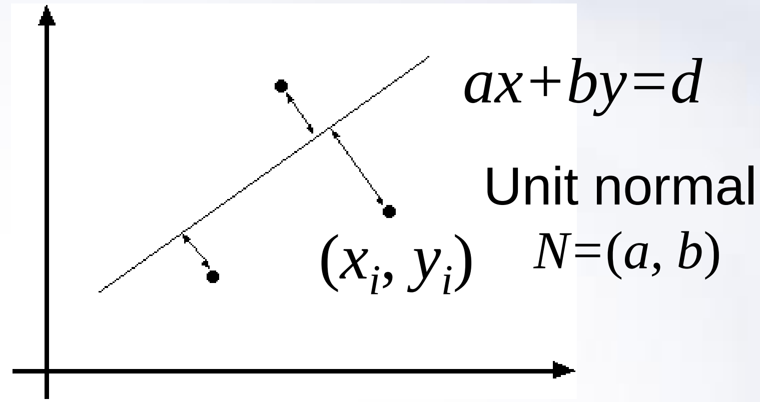
设直线 L 的方程为 $Ax+By+C=0$，点 P 的坐标为（$x0, y0$），则点 P 到直线 L 的距离为:

$$\frac{|Ax_0 + By_0 + C|}{\sqrt{A^2+B^2}}$$

可以解决无法拟合垂直直线问题

Total least squares

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



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

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

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

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

Slide: S. Lazebnik

Total least squares

Solution to $(U^TU)N = 0$, subject to $||N||^2 = 1$: eigenvector of U^TU associated with the smallest eigenvalue

$$(U^TU)N = 0$$

$$A = U^TU$$

$$AX = \lambda X \rightarrow (A - \lambda \mathrm{E})\mathrm{X} = 0$$

拉格朗日乘数法

$$N^TN=1$$

$$L = E - \frac{1}{2}\lambda(N^TN - 1)$$

$$\frac{\partial L}{\partial N} = \frac{\partial E}{\partial N} - \lambda N = 0$$

$$(U^TU - \lambda E)N = 0$$

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

转化为求U^TU 特征值的特征向量。

Slide: S. Lazebnik

Overview

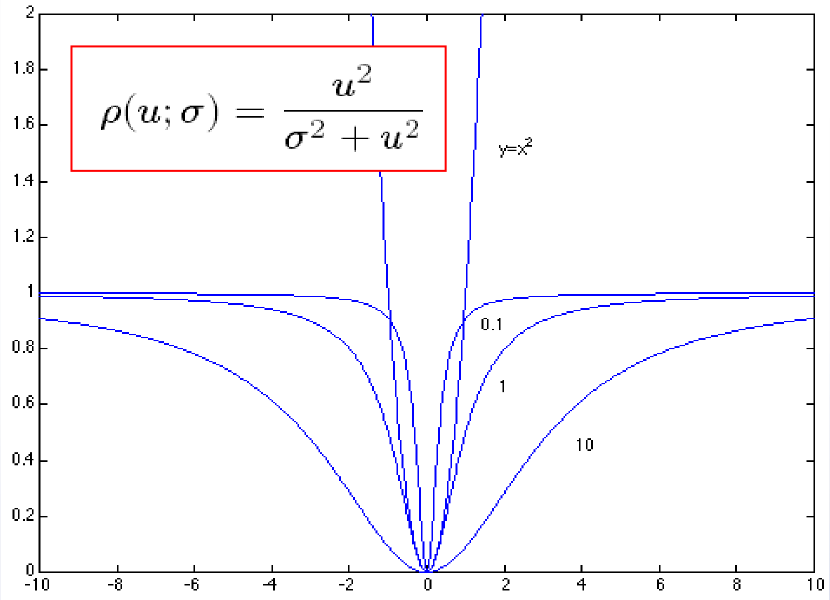
- Fitting techniques
  - Least Squares
  - Total Least Squares
  - Robust Fitting
- RANSAC
- Hough Voting
- Alignment as a fitting problem

Least squares: Robustness to noise

考虑到最小二乘法与总体最小二乘法
均受噪声影响较大，使用鲁棒估计进行改进

- General approach: minimize $\sum_i \rho\left(u; \sigma\right)$

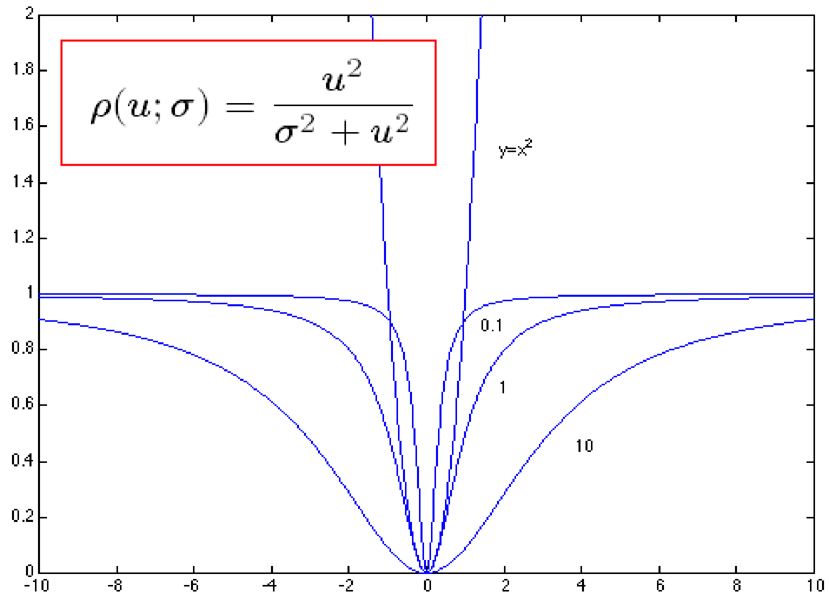
u – 拟合误差
ρ – 经过σ尺度缩放后的拟合误差



Least squares: Robustness to noise

- General approach: minimize $\sum_i \rho\left(u; \sigma\right)$

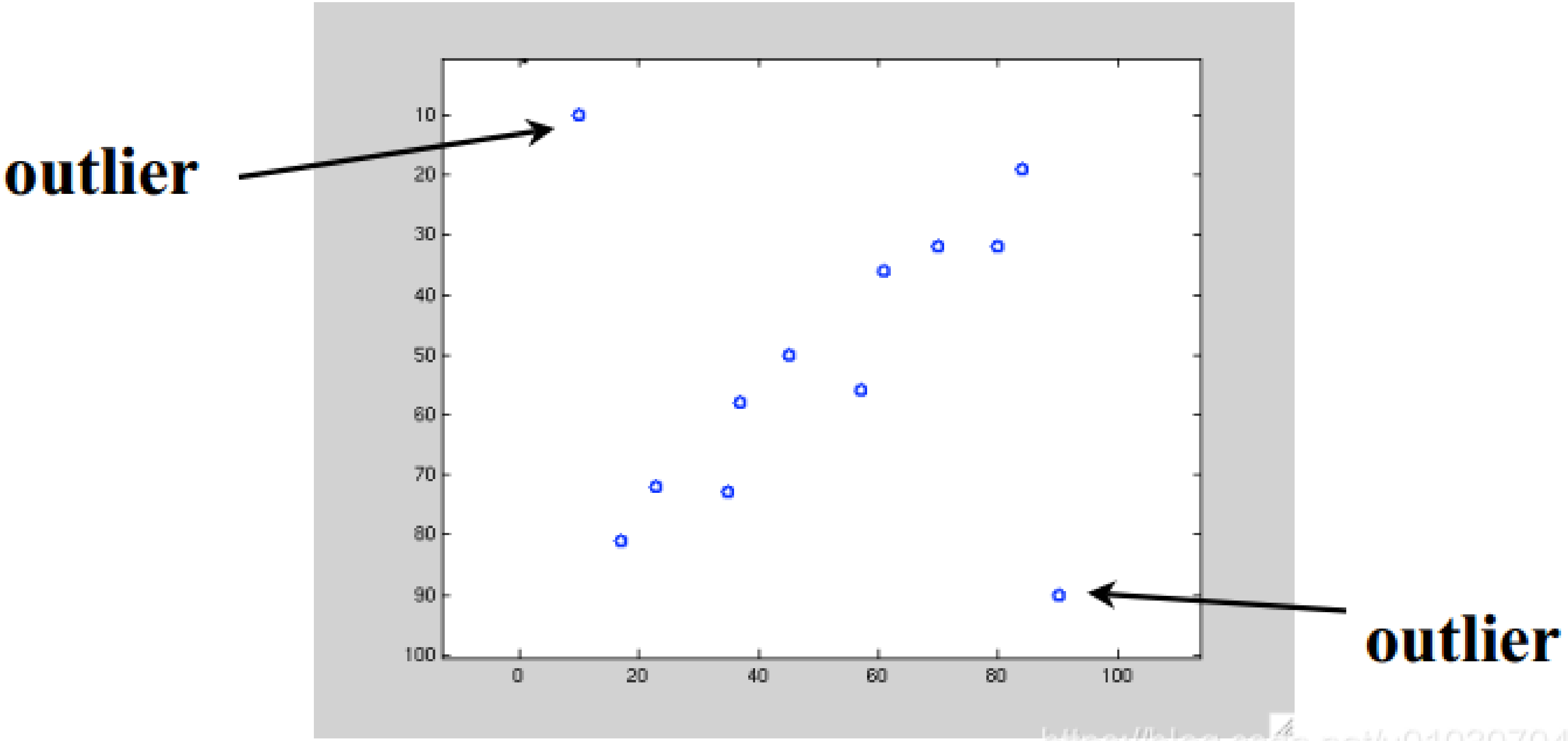
u – 拟合误差
ρ – 经过σ尺度缩放后的拟合误差



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

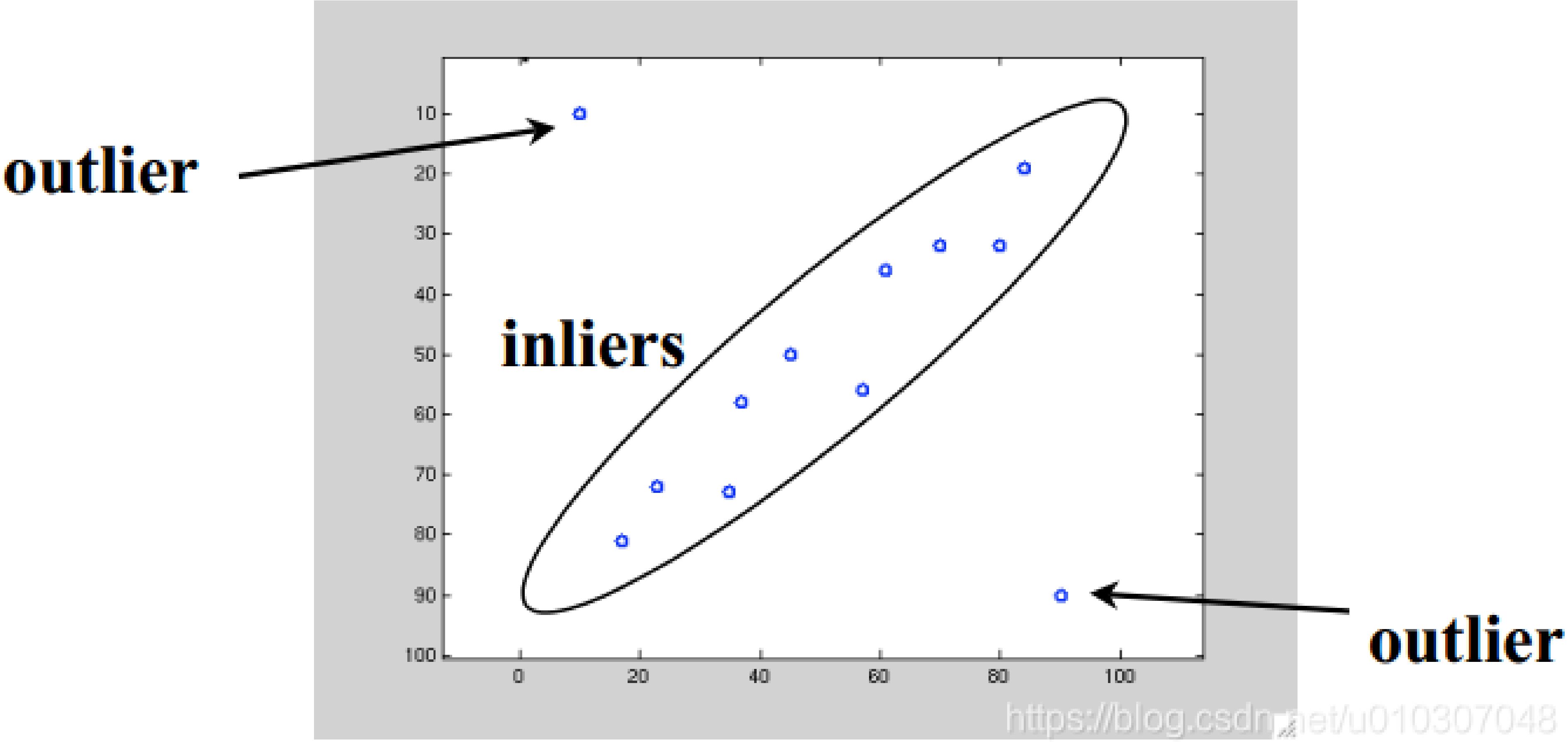
u越大，误差越大，放缩效应越明显

Least squares: Robustness to noise



Slide: S. Lazebnik

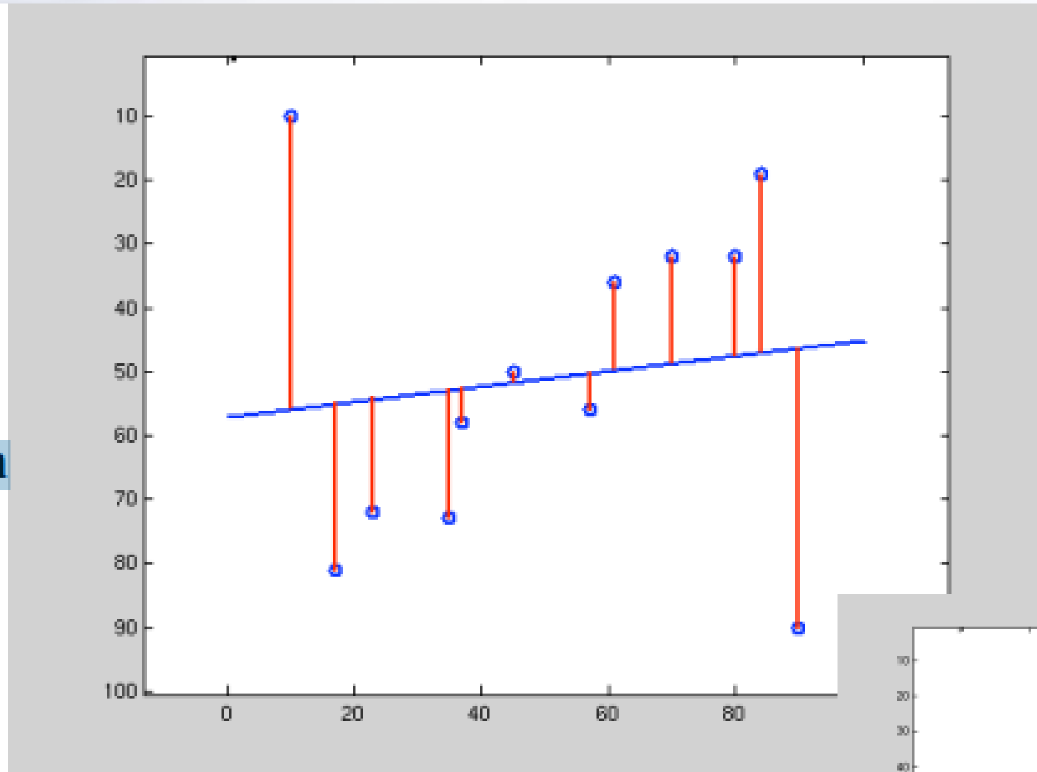
Least squares: Robustness to noise



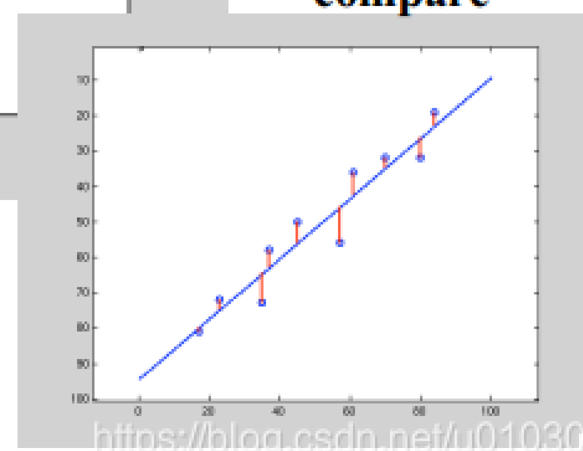
Slide: S. Lazebnik

Least squares: Robustness to noise

Least squares regression with outliers



compare



Solution: Estimation methods that are robust to outliers.

Slide: S. Lazebnik

Overview

- Fitting techniques
  - Least Squares
  - Total Least Squares
  - Robust Fitting
- RANSAC
- Hough Voting
- Alignment as a fitting problem

RANSAC

- Robust fitting can deal with a few outliers – what if we have very many?
- Random sample consensus (RANSAC):
  Very general framework for model fitting in the presence of outliers
- Outline
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find all remaining points that are “close” to the model and reject the rest as outliers
  - Do this many times and choose the best model

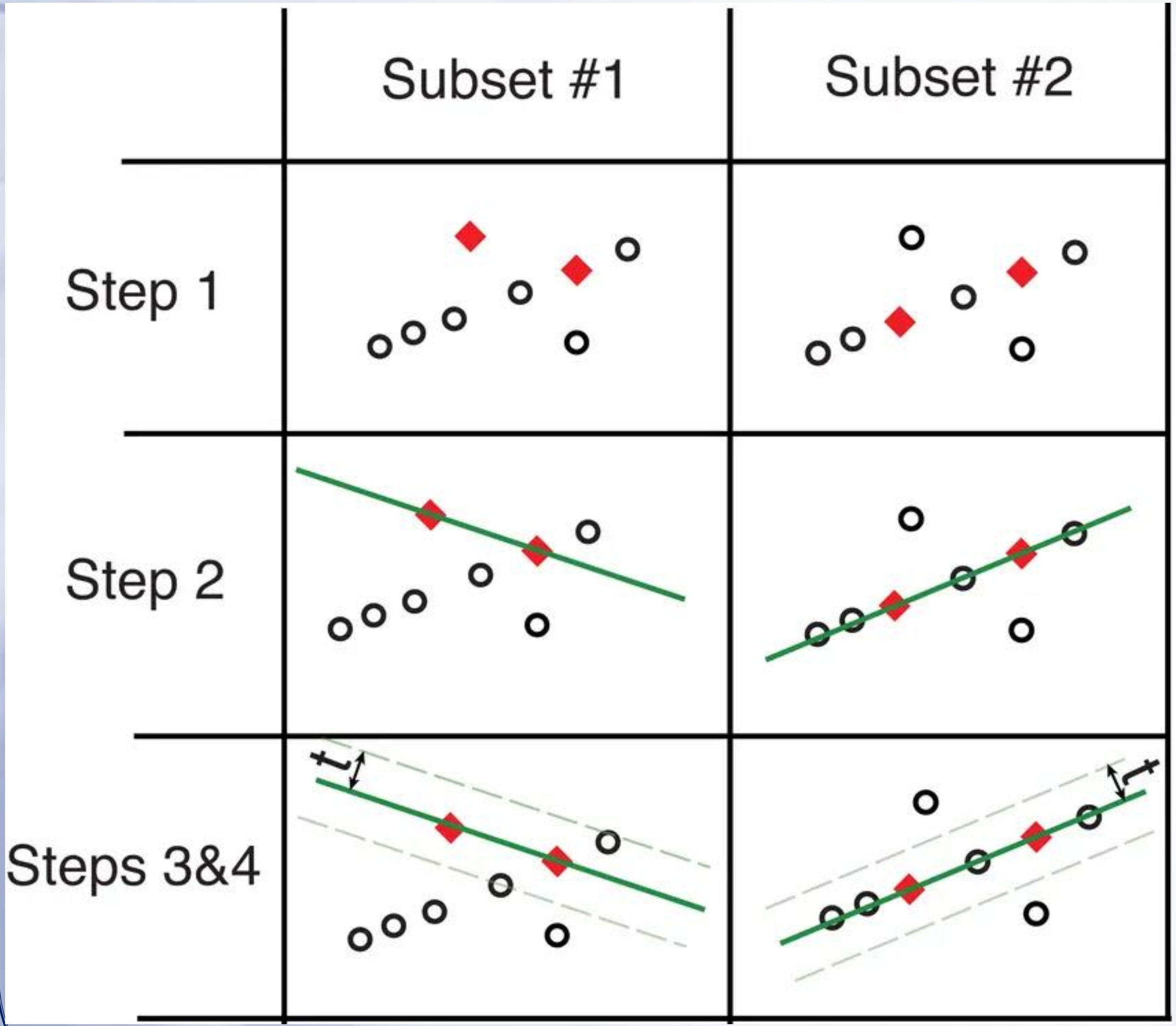
Slide: S. Lazebnik

RANSAC for line fitting

- Repeat ***N*** times:
- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Source: M. Pollefeys

RANSAC for line fitting



1. 首先选择最小集合，例如估计一条直线参数，就需要选择两个点

2. 两点确定一条直线，有了两个点可以写出一个直线方程

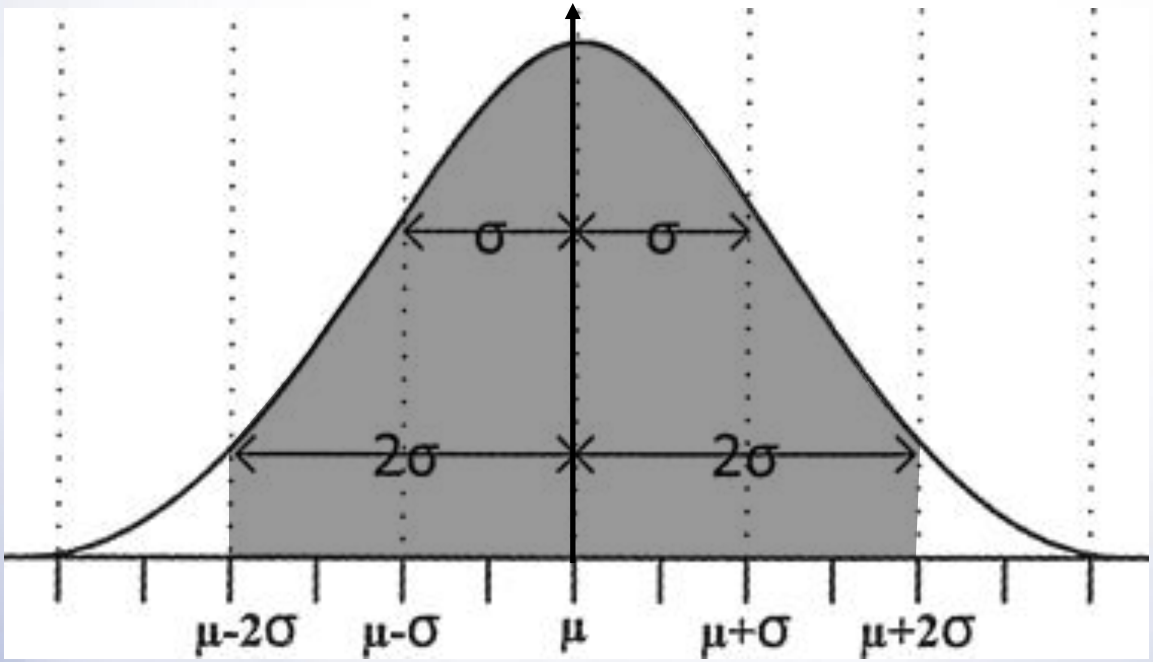
3. 计算剩余点到该直线的距离

4. 根据设定阈值 t 来计算距离小于阈值点数量作为投票点/内点

5. 重复1-4，迭代N次，记录每次迭代选择点、拟合曲线和投票数，投票数/内点数最大所对应的直线模型就是找到的直线

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$



Source: M. Pollefeys

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99)

$$N = \frac{\log(1-z)}{\log(1-p^s)}$$

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
  - 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

Source: M. Pollefeys

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

Overview

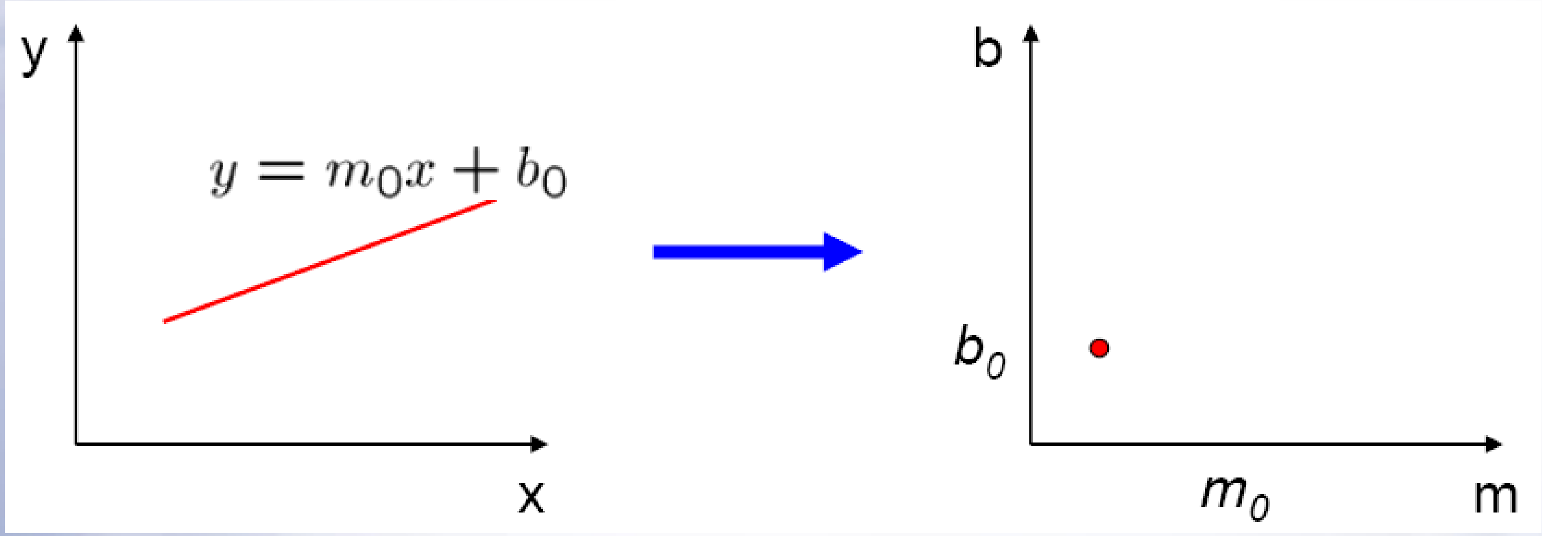
- Fitting techniques
  - Least Squares
  - Total Least Squares
  - Robust Fitting
- RANSAC
- Hough Voting
- Alignment as a fitting problem

Parameter space representation

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

Image space

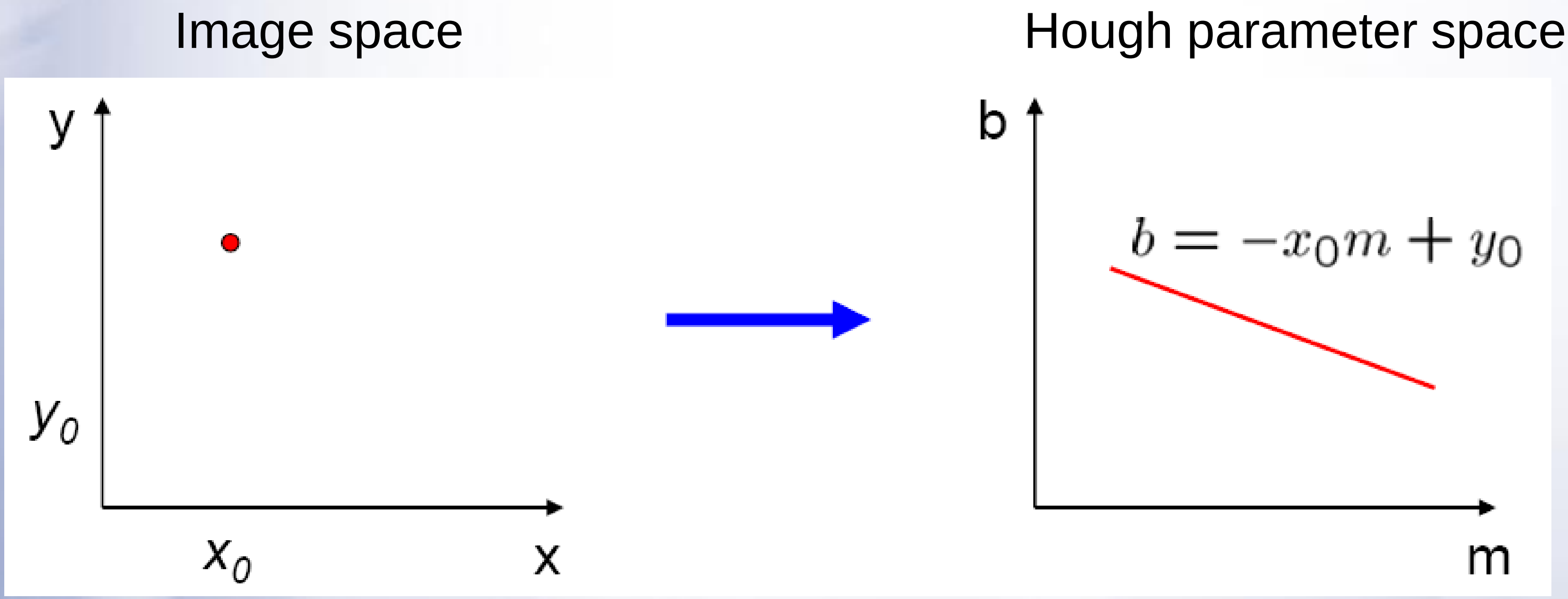
Hough parameter space



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 $b = -x_0 m + y_0$
  - This is a line in Hough space



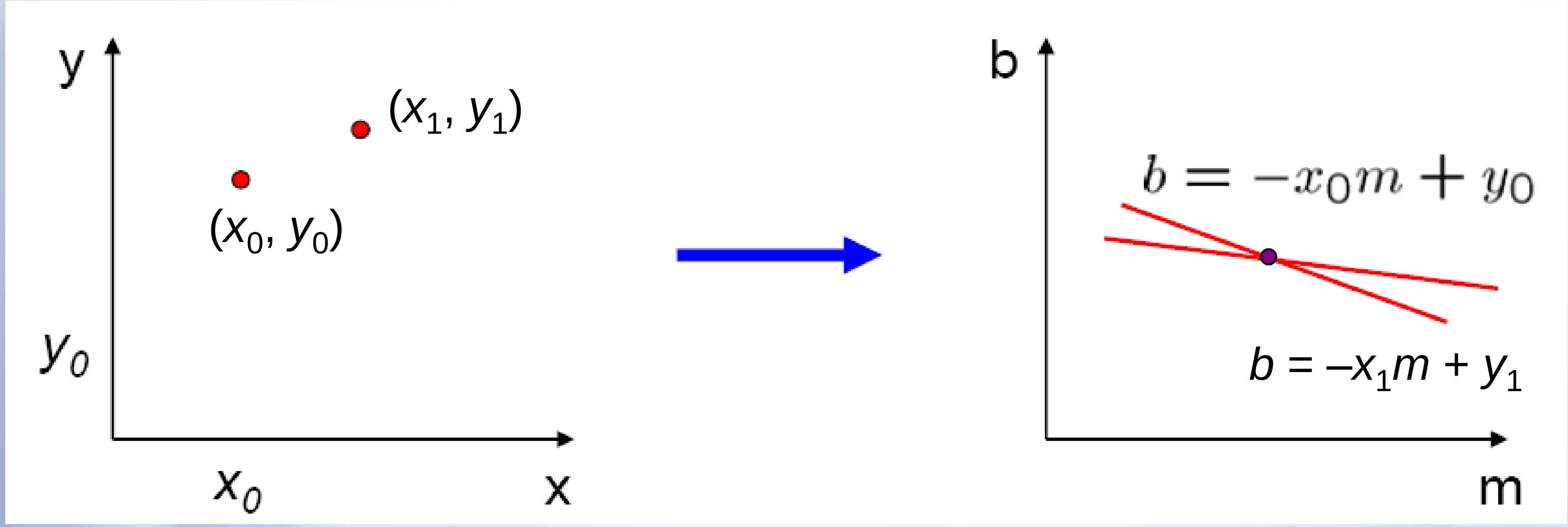
Source: S. Seitz

Parameter space representation

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

Image space

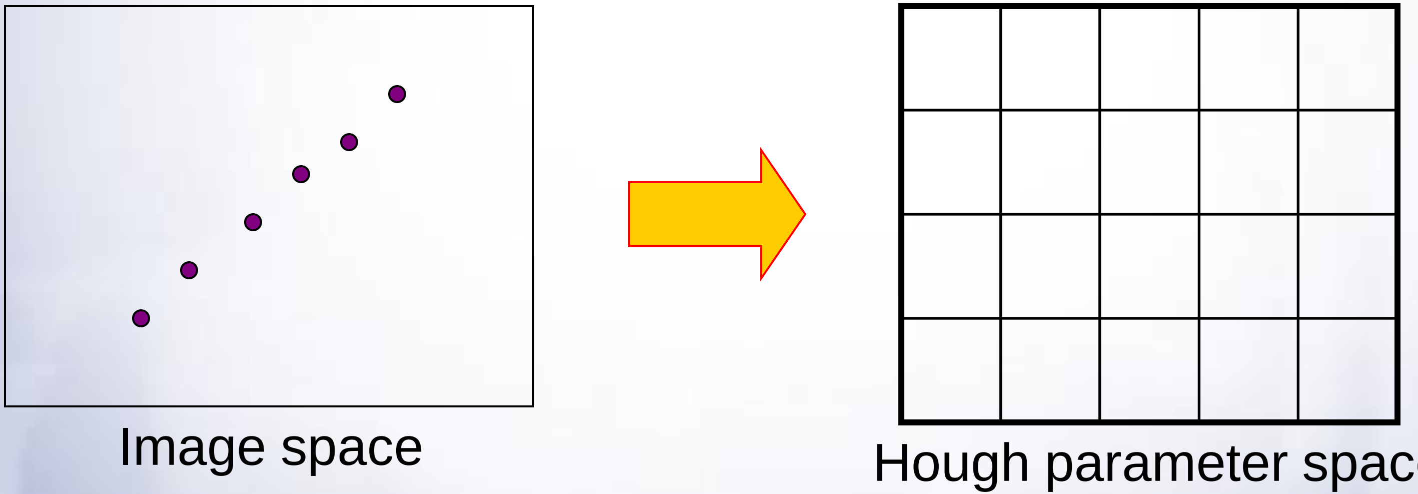
Hough parameter space



Source: S. Seitz

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

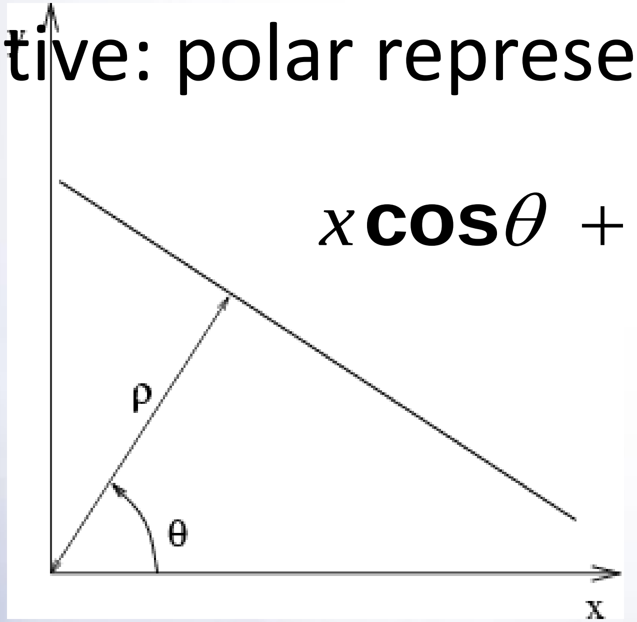


Parameter space representation

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

$$y = \left(-\frac{cos\theta}{sin\theta}\right)x + \left(\frac{\rho}{sin\theta}\right)$$

- Alternative: polar representation

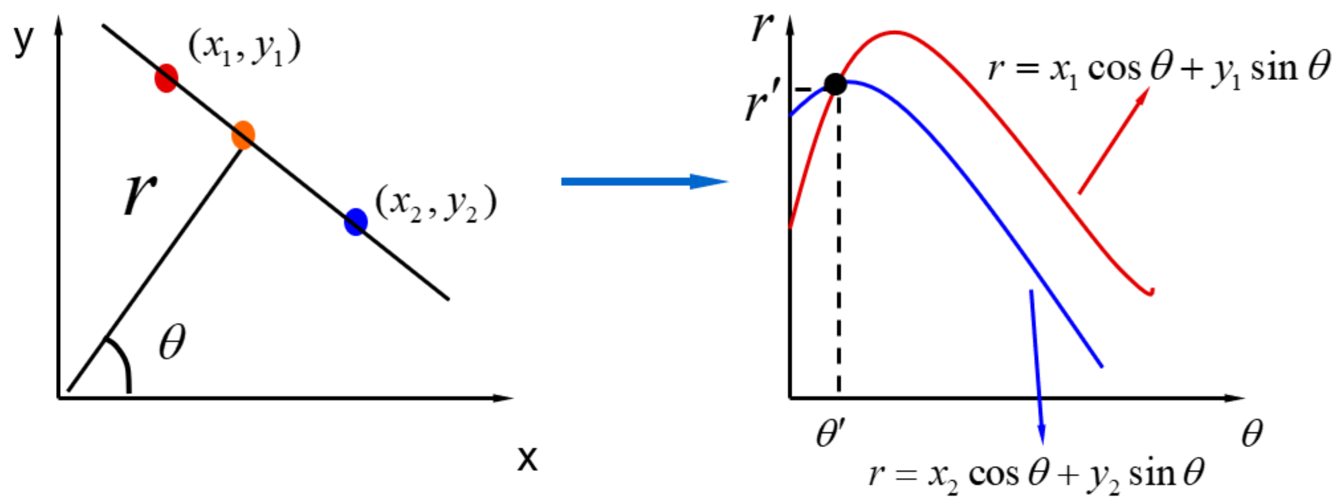


$$x\,\mathbf{cos}\theta + y\,\mathbf{sin}\theta = \rho$$

$$0 \leq \theta \leq 2\pi$$
$$0 \leq \rho \leq \rho_{\max}$$

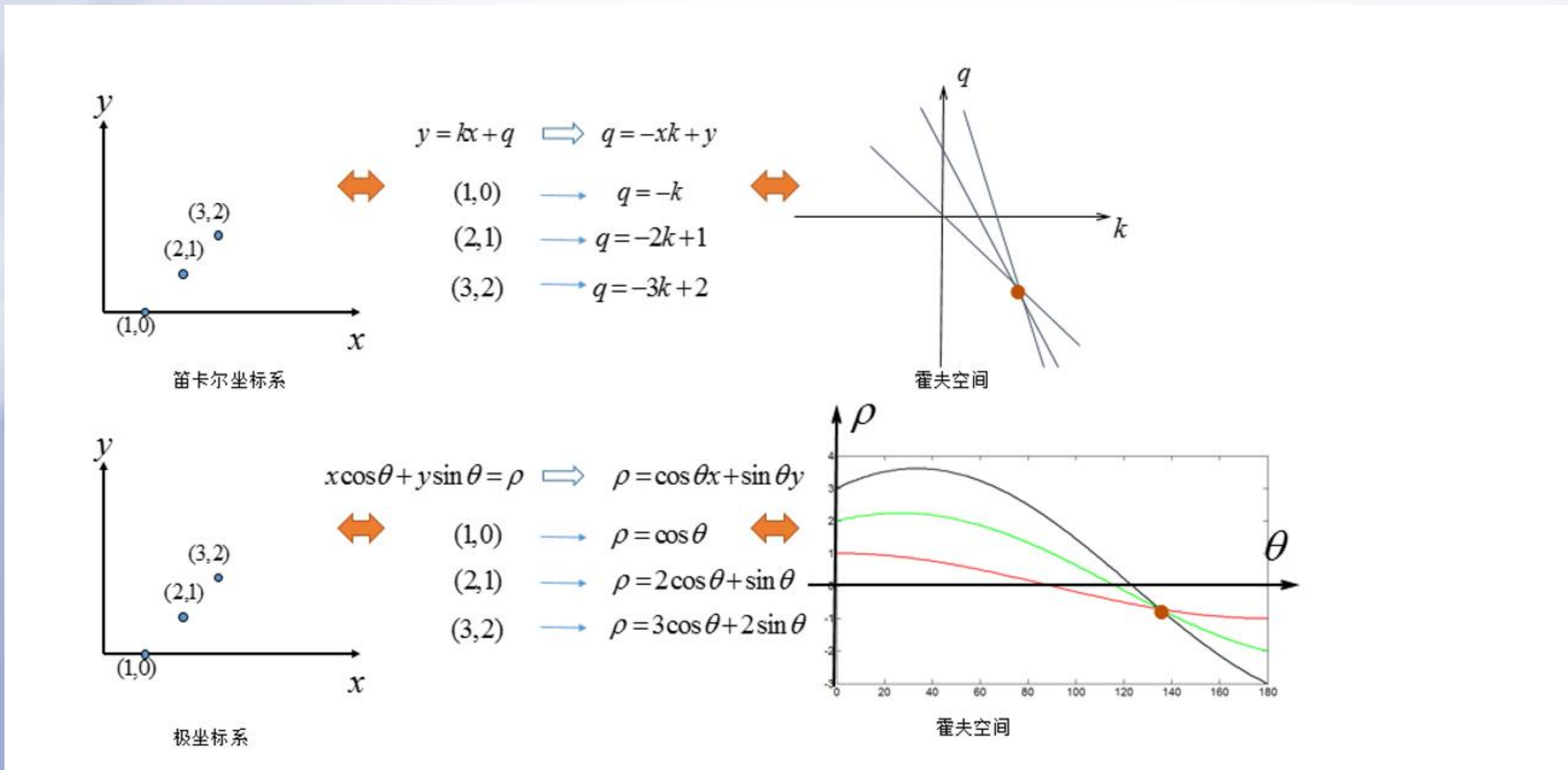
Each point will add a sinusoid in the (θ,ρ) parameter space

Parameter space representation

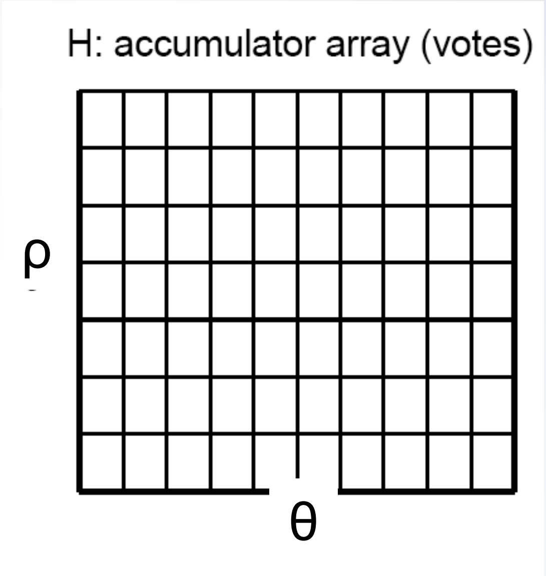


- 经过变换，图像空间中的每个点 (x,y)就被映射为一个(r,***θ***)极坐标空间中的正弦曲线。
- 而图像空间中共线的点所对应的(r,***θ***)空间中正弦曲线相交于一点(r’,***θ***’)。

Parameter space representation



Algorithm outline



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

    For θ = 0 to 180

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

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

    end

  end
- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
  - The detected line in the image is given by $\rho = x \cos \theta + y \sin \theta$

Algorithm outline

霍夫变换检测直线步骤：

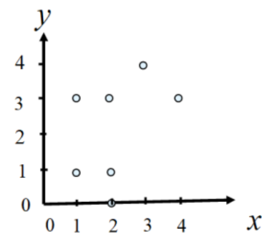


Image Space

步骤：

1.离散化θ。

θ=-45°,0°,45°,90°

2.按点的坐标(x,y)和每个角度θ求r

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

(x,y)	-45°	0°	45°	90°
(2,0)	1.4	2	1.4	0
(1,1)	0	1	1.4	1
(2,1)	0.7	2	2.1	1
(1,3)	-1.4	1	2.8	3
(2,3)	-0.7	2	3.5	3
(4,3)	0.7	4	4.9	3
(3,4)	-0.7	3	4.9	4

- 3 统计(r,θ)出现的次数。

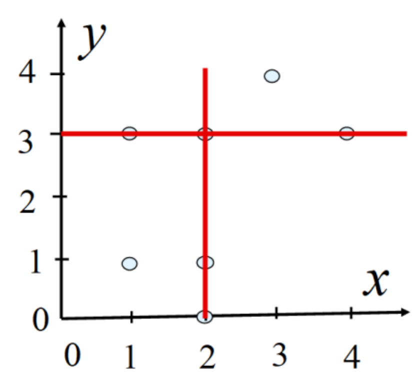
	−1.4	−0.7	0	0.7	1	1.4	2	2.1	2.8	3	3.5	4	4.9
−45°	1	2	1	2		1							
0°					2		3			1		1	
45°						2		1	1		1		2
90°			1		2					3		2	

- 最大次数3出现$(r,\theta)=(2,0^\circ)$和$(r,\theta)=(3,90^\circ)$
- 则相对应的图像空间中的线分别为：

$$2 = x\cos 0 + y\sin 0 \quad 即\ x = 2$$

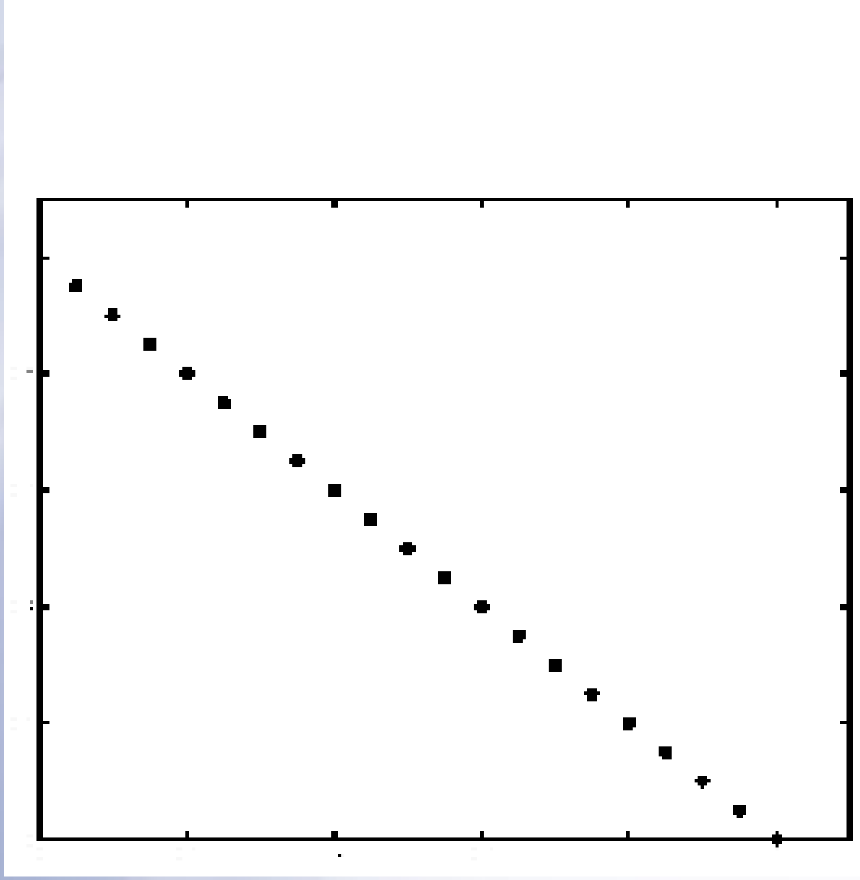
和

$$3 = x\cos 90 + y\sin 90 \quad 即\ y = 3$$

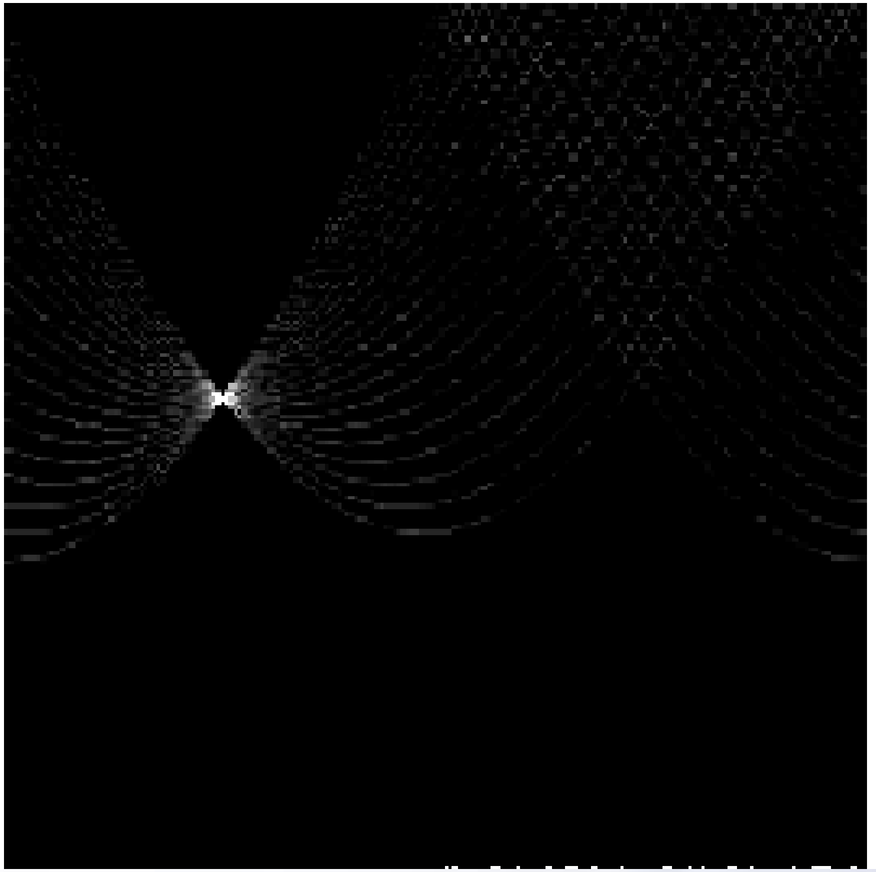


霍夫变换检测到的直线

Basic illustration

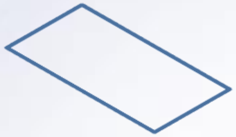


features



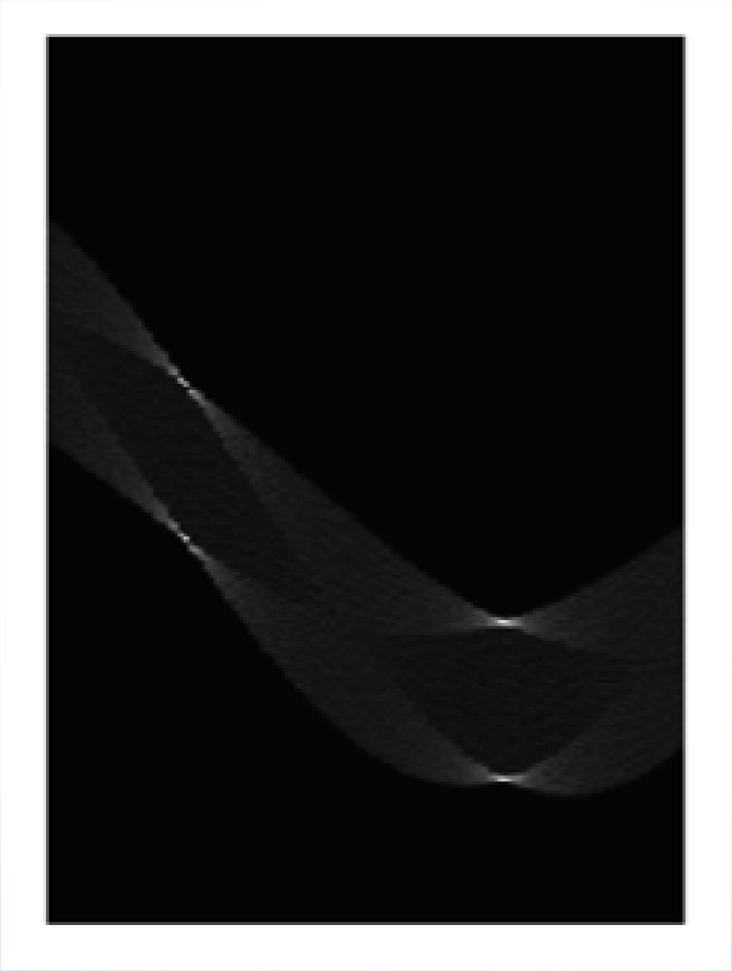
votes

Other shapes

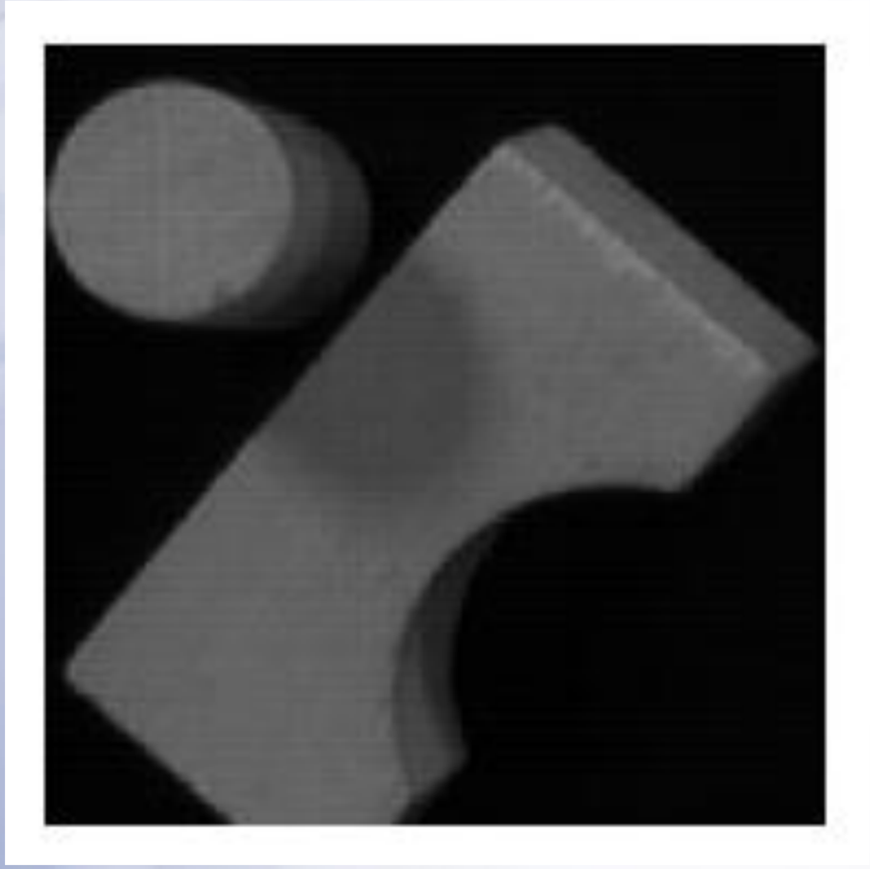


Square

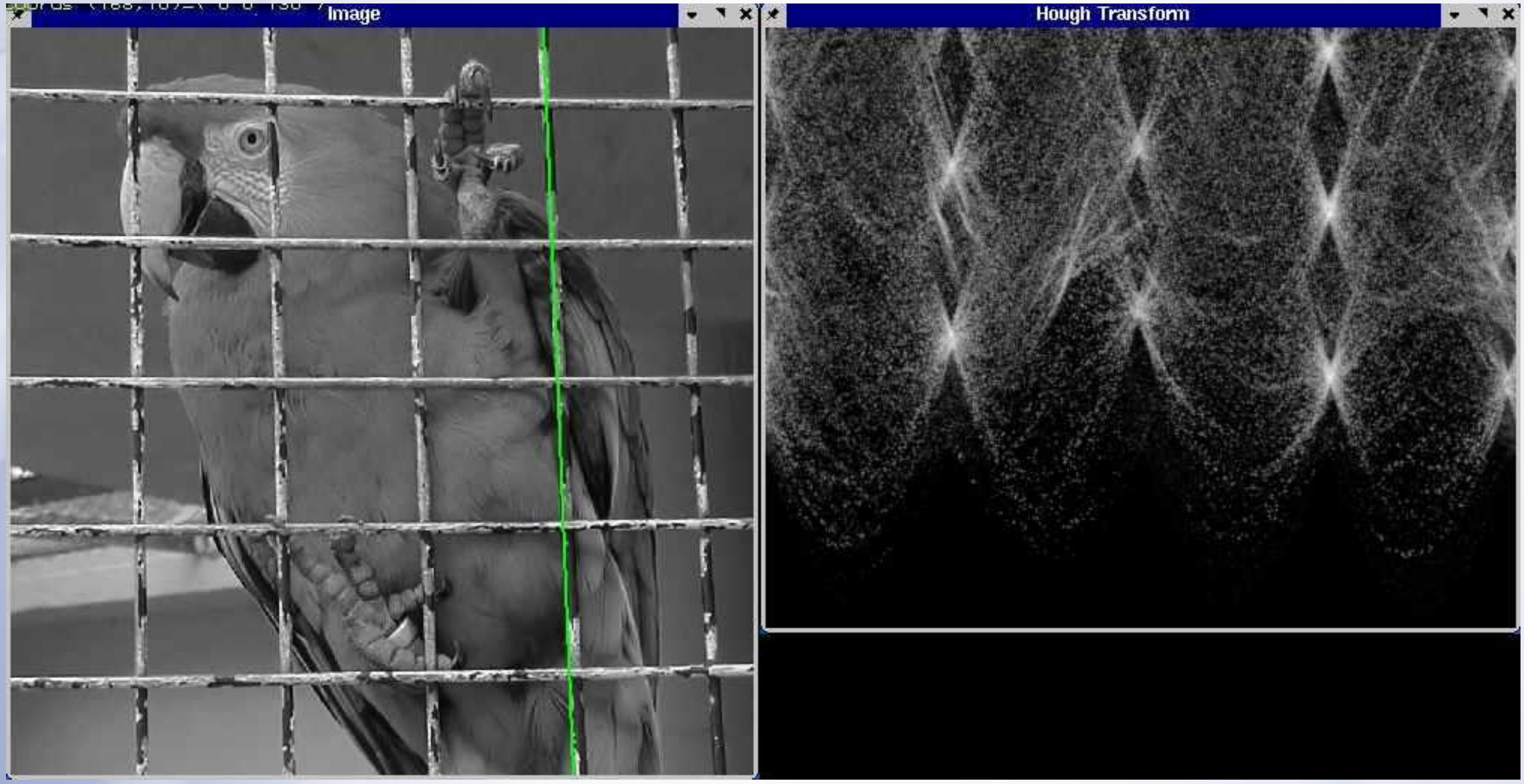
$$y = \left(-\frac{cos\theta}{sin\theta}\right)x + \left(\frac{\rho}{sin\theta}\right)$$



Several lines

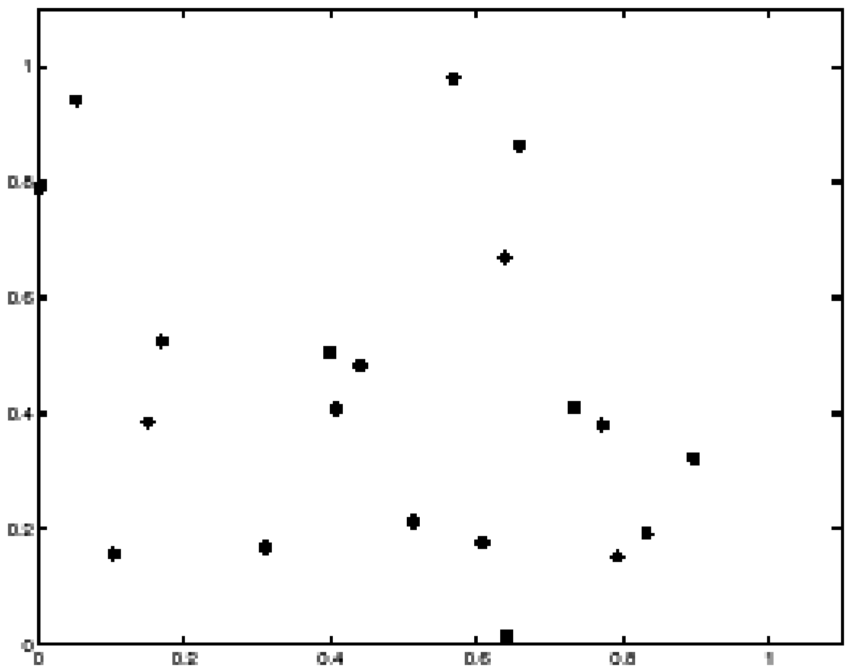


A more complicated image

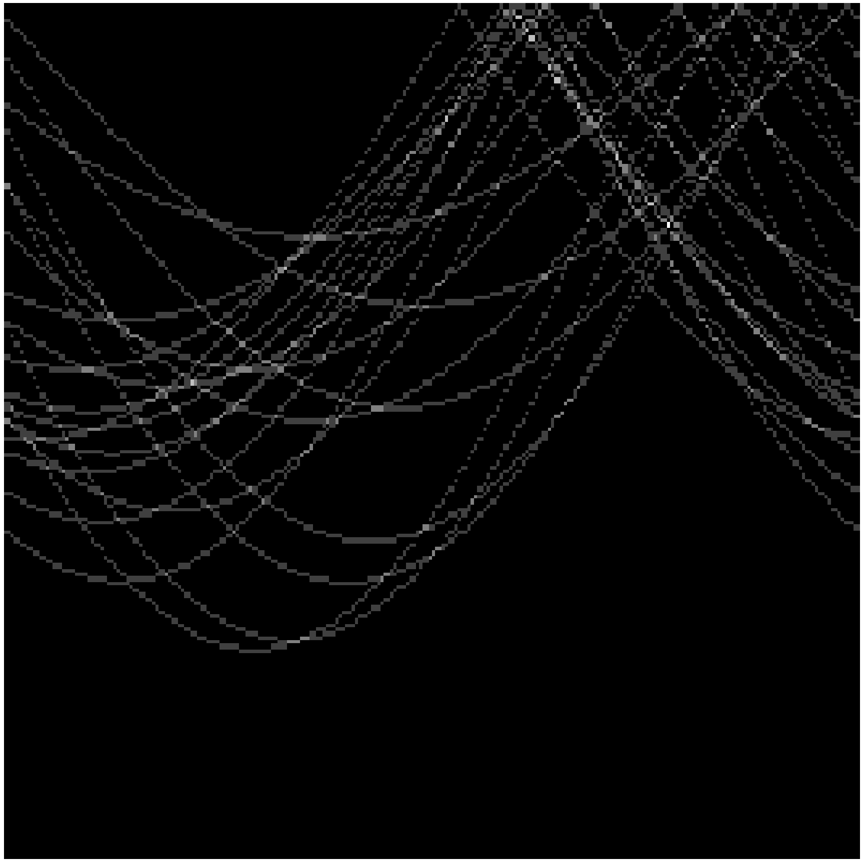


http://ostatic.com/files/images/ss_hough.jpg

Random points



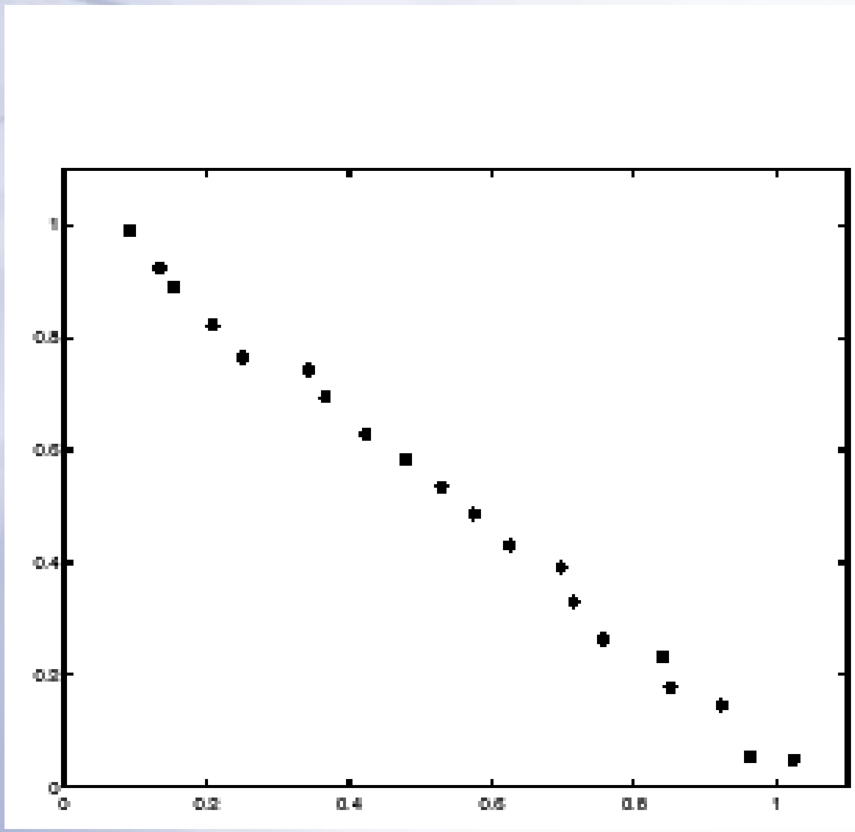
features



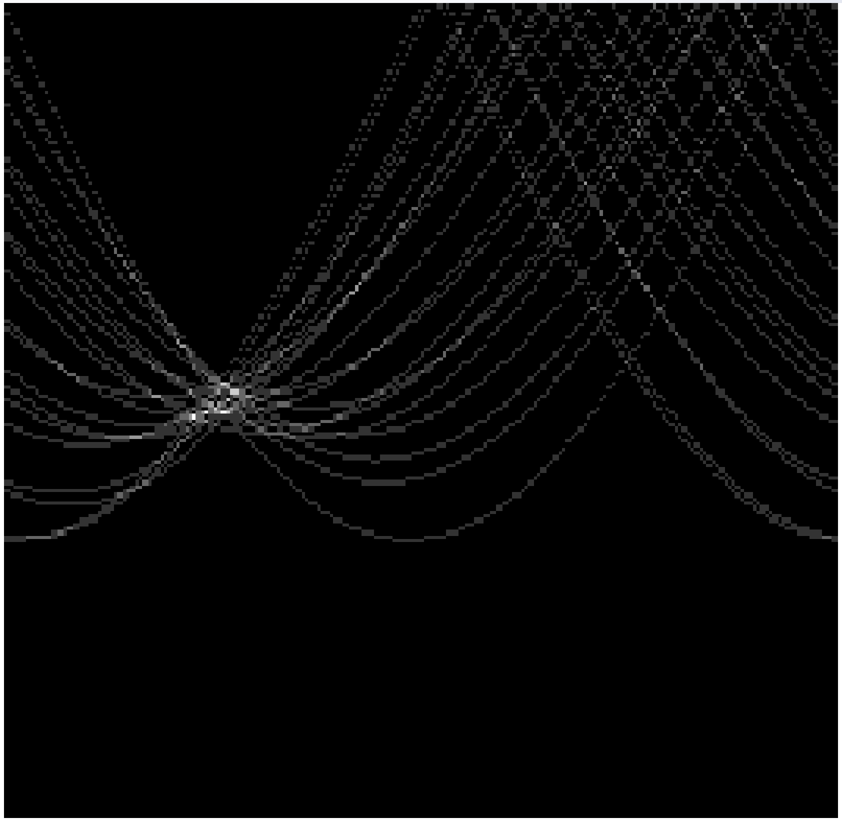
votes

- Uniform noise can lead to spurious peaks in the array

Effect of noise



features



votes

- Peak gets fuzzy and hard to locate

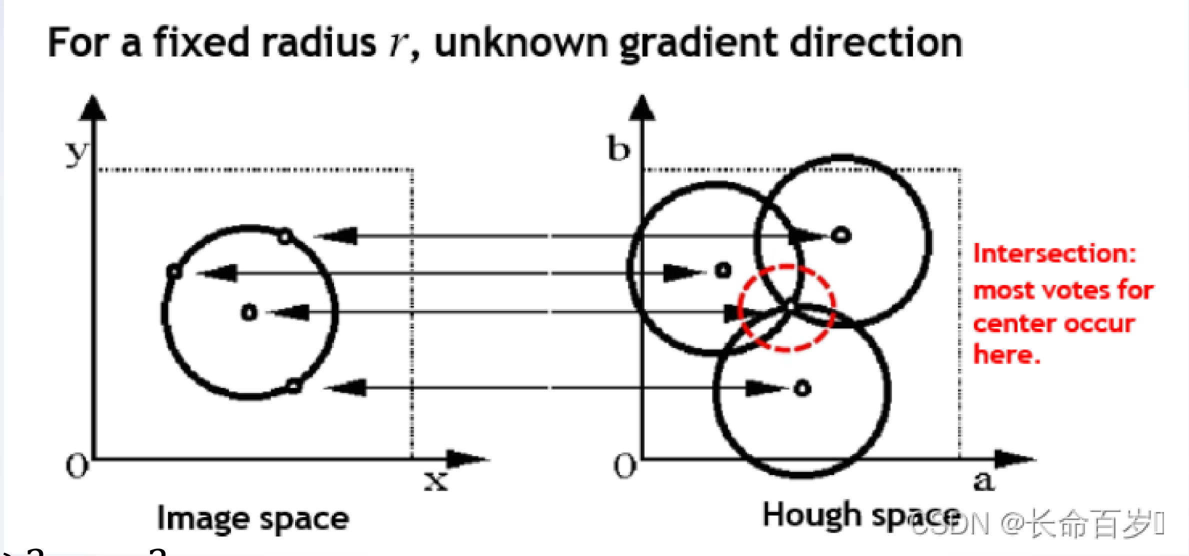
Dealing with noise

- Choose a good grid / discretization
  - Too coarse: large votes obtained when too many different lines correspond to a single bucket
  - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Try to get rid of irrelevant features
  - Take only edge points with significant gradient magnitude

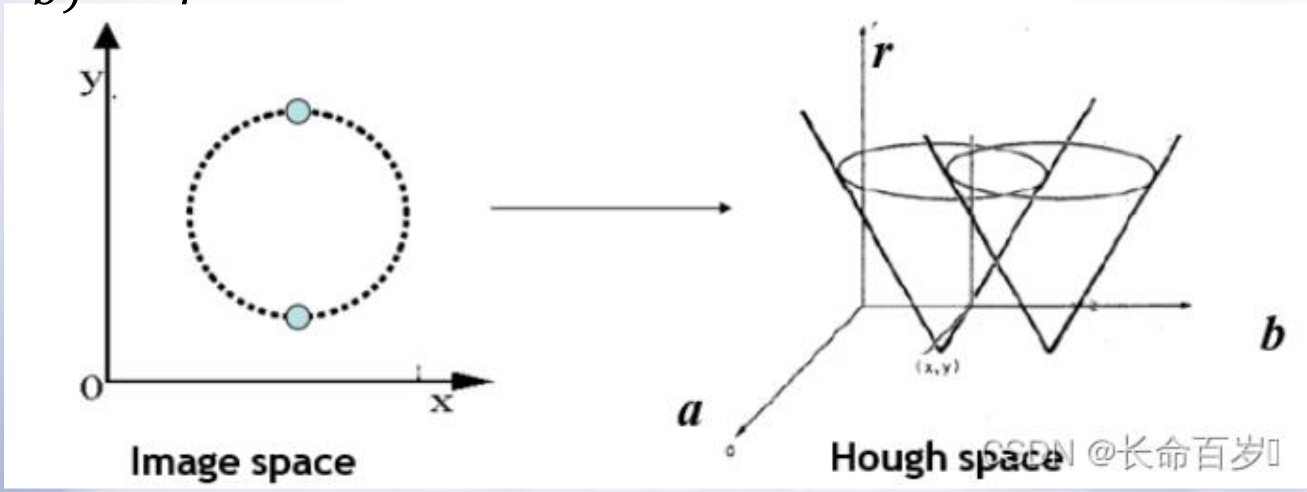
Hough transform for circles

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

Hough transform for circles



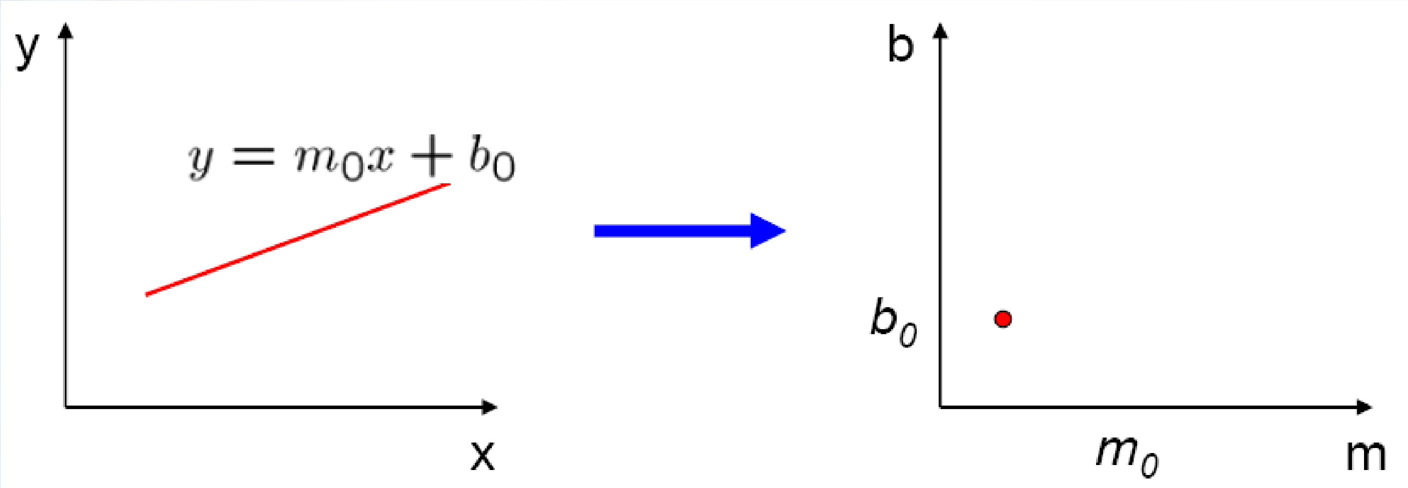
$$(x-a)^2+(y-b)^2=r^2$$

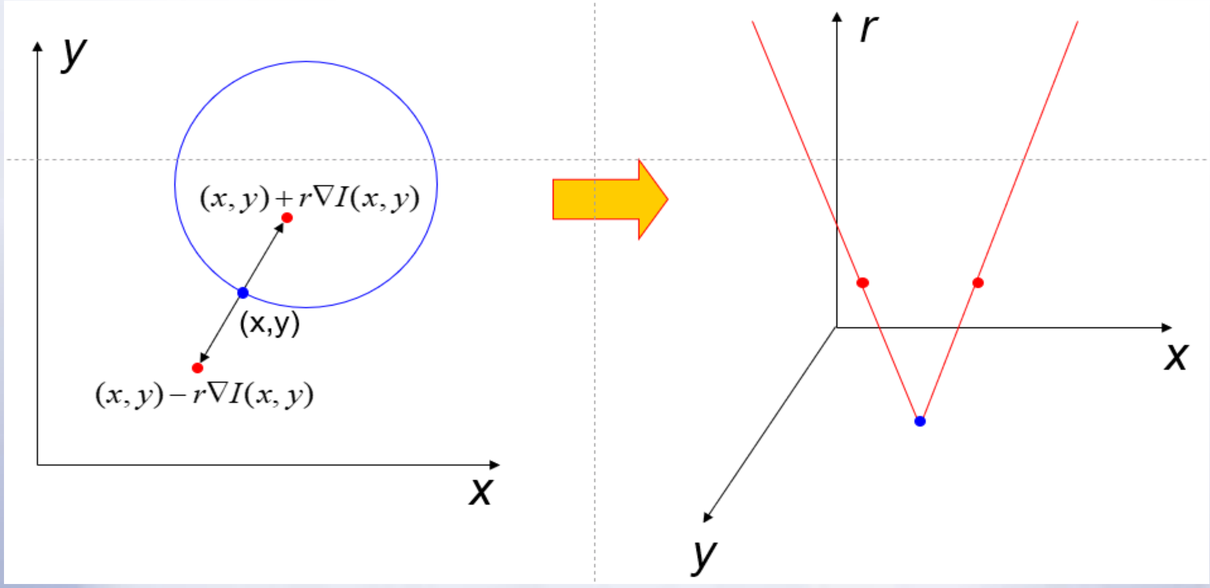


Hough transform for circles

Image space

Hough parameter space





Hough transform for circles

For every edge pixel (x,y) :

  For each possible radius value r:

    For each possible gradient direction θ:
      // or use estimated gradient

    $a = x - r\cos(\theta)$

    $b = y + r\sin(\theta)$

    $H[a,b,r] \mathrel{+}= 1$

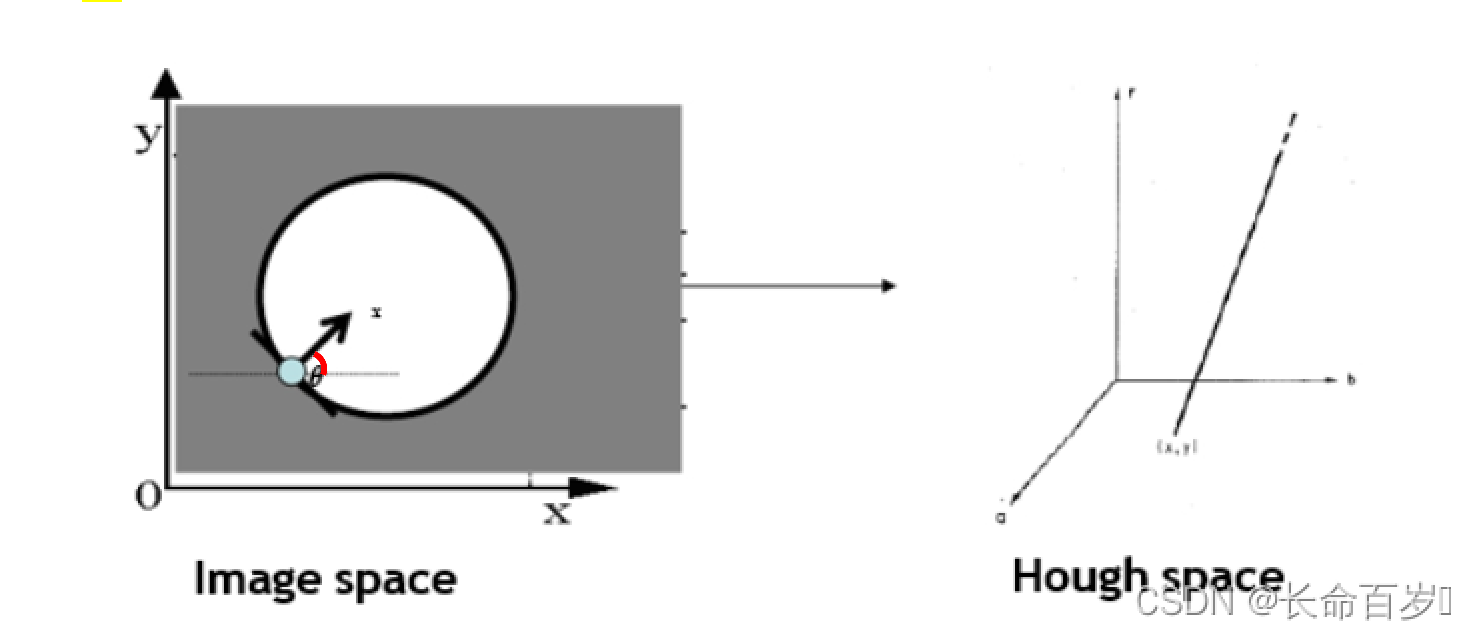
  end

end

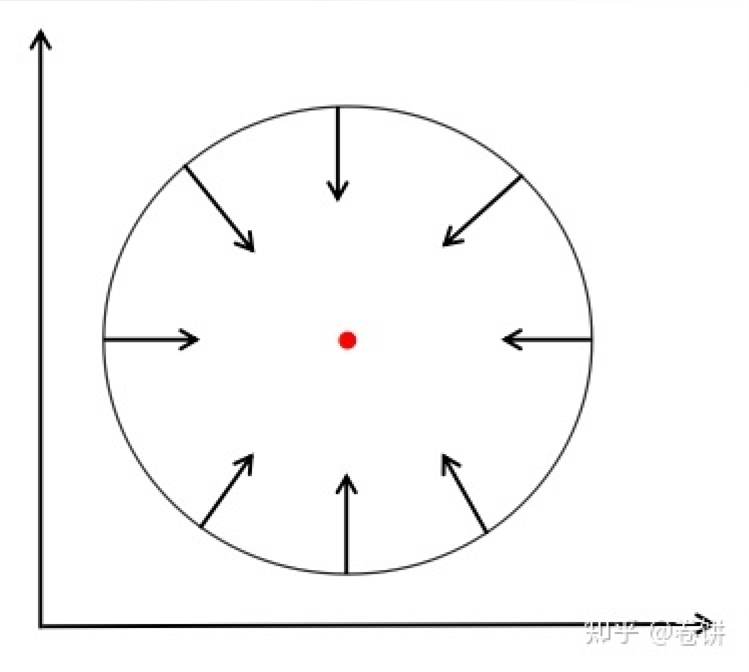
Hough transform for circles

$$(x-a)^2+(y-b)^2=r^2$$

$$x = a + rcos\theta$$
$$y = b + rsin\theta$$



Hough transform for circles



（1）首先对图像应用边缘检测
（2）使用sobel算子计算所有像素的梯度
（3）遍历边缘检测之后的所有非0的像素点，沿着梯度方向画线，每个点有一个累加器，有一个线经过该点，累加器加1，对所有累加器进行排序，根据阈值找到所有可能的圆心
（4）计算边缘图像中所有的非0像素点距离圆心的距离，距离从小到大排序，选取合适的半径
（5）对选取的半径设置累加器，对于满足半径r的累加器+1

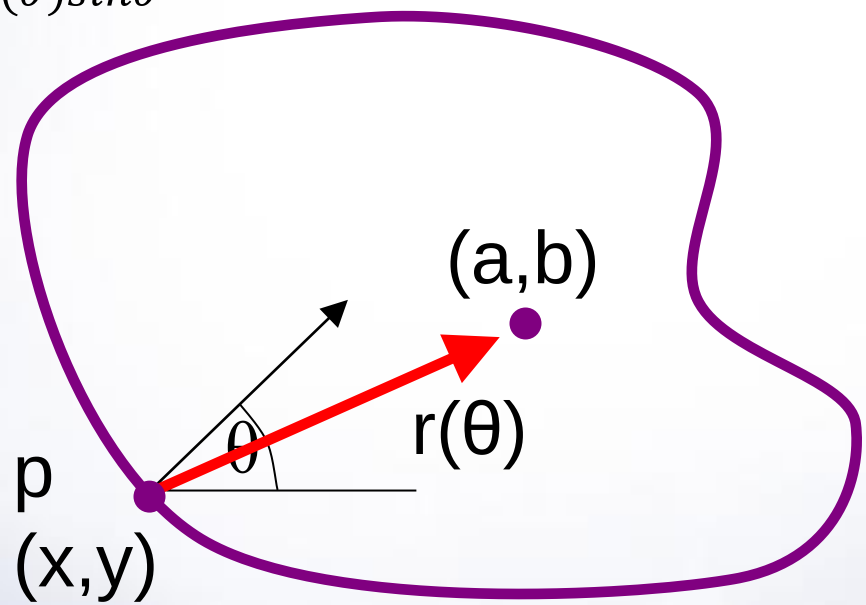
Generalized Hough transform

- We want to find a shape 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

$$(x-a)^2 + (y-b)^2 = r(\theta)^2$$

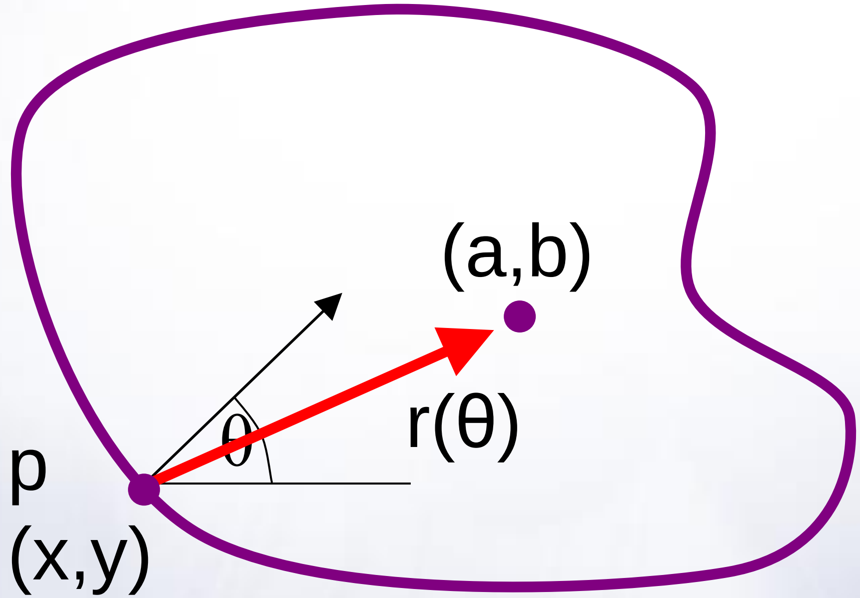
$$x = a + r(\theta)cos\theta$$

$$y = b + r(\theta)sin\theta$$



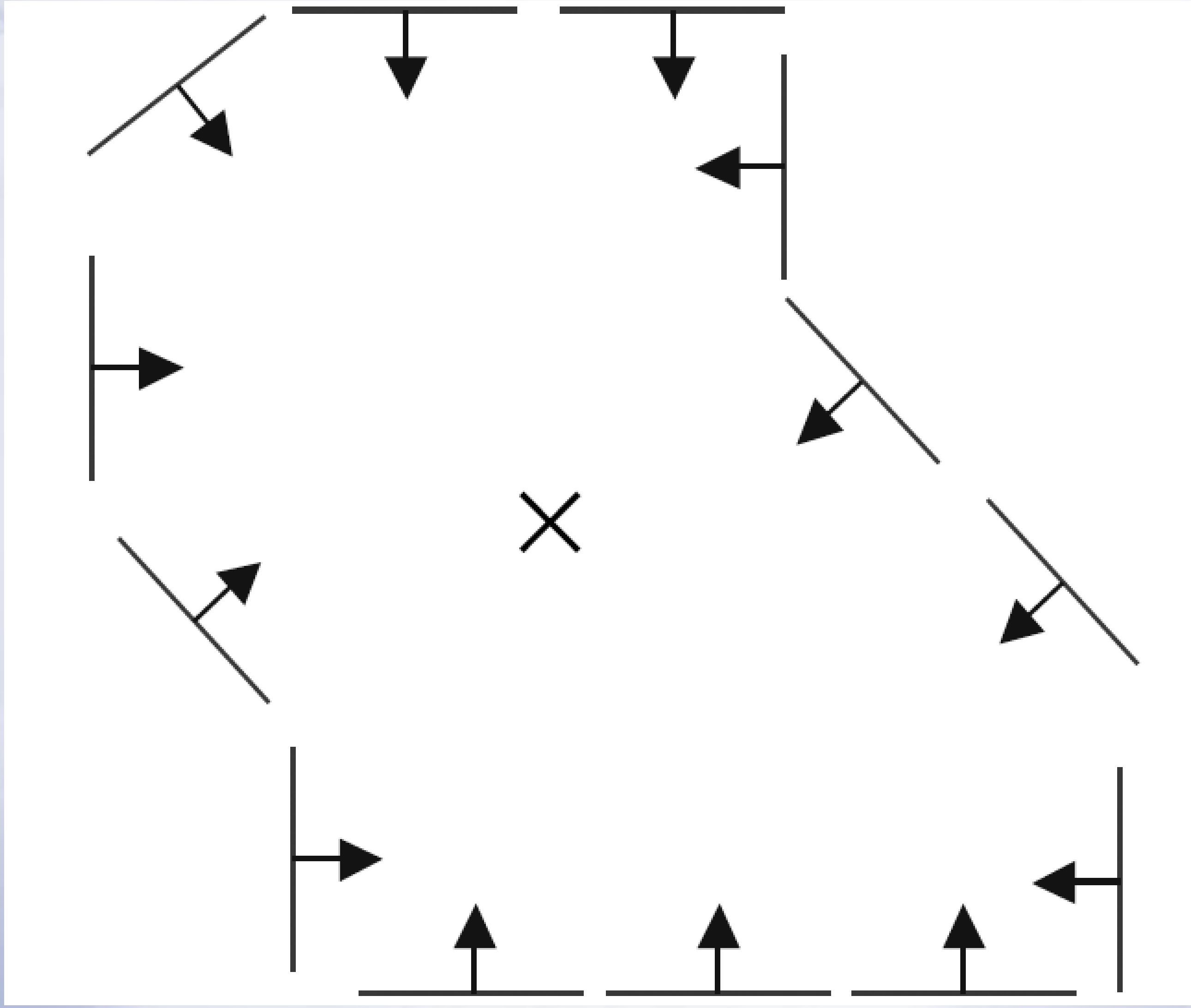
Generalized Hough transform

- For model shape: construct a table indexed by ϑ storing displacement vectors r as function of gradient direction
- Detection: For each edge point *p* with gradient orientation ϑ:
  - Retrieve all *r* indexed with ϑ
  - For each $r(\vartheta)$, put a vote in the Hough space at $p + r(\vartheta)$
- 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



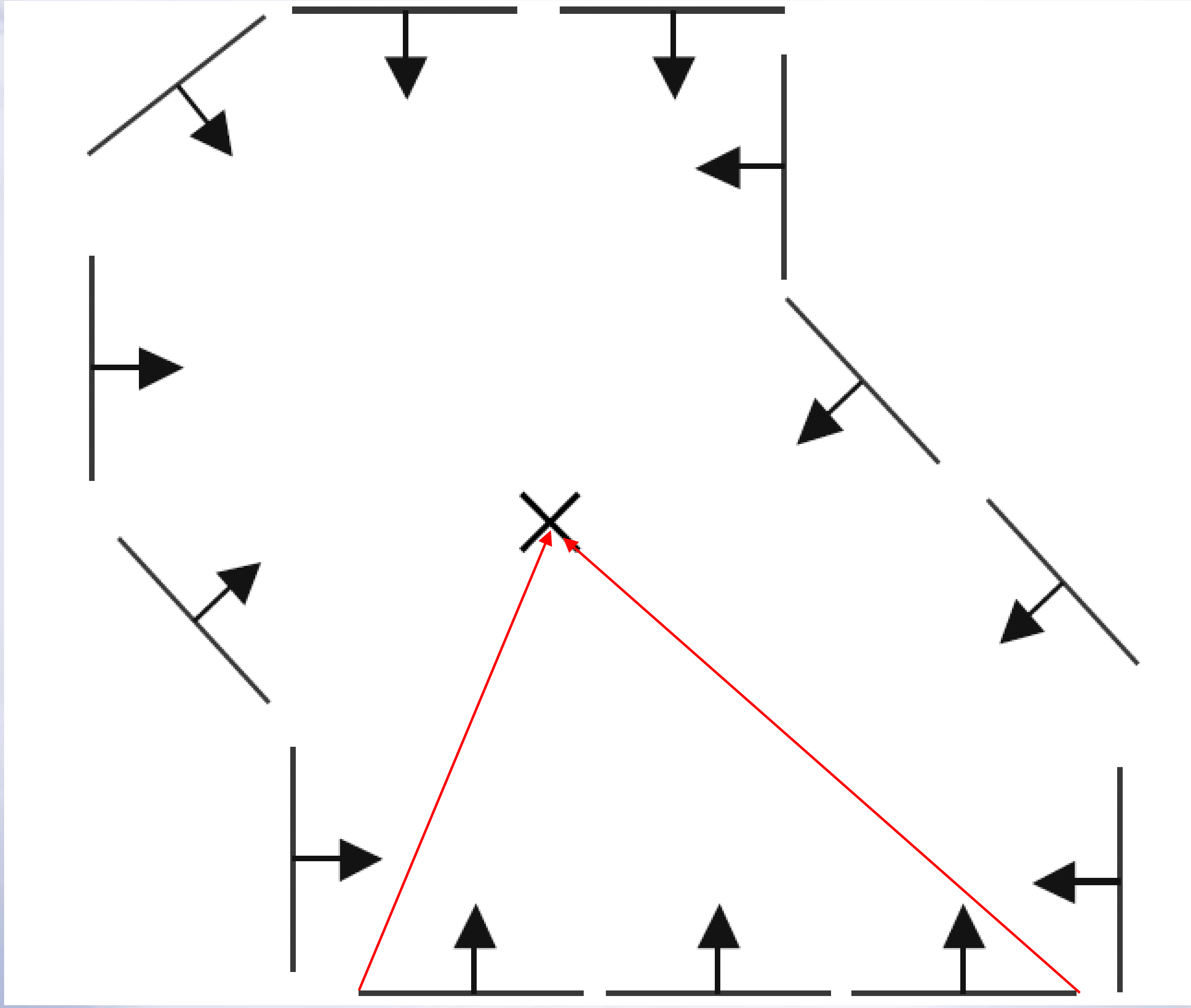
Source: K. Grauman

Example



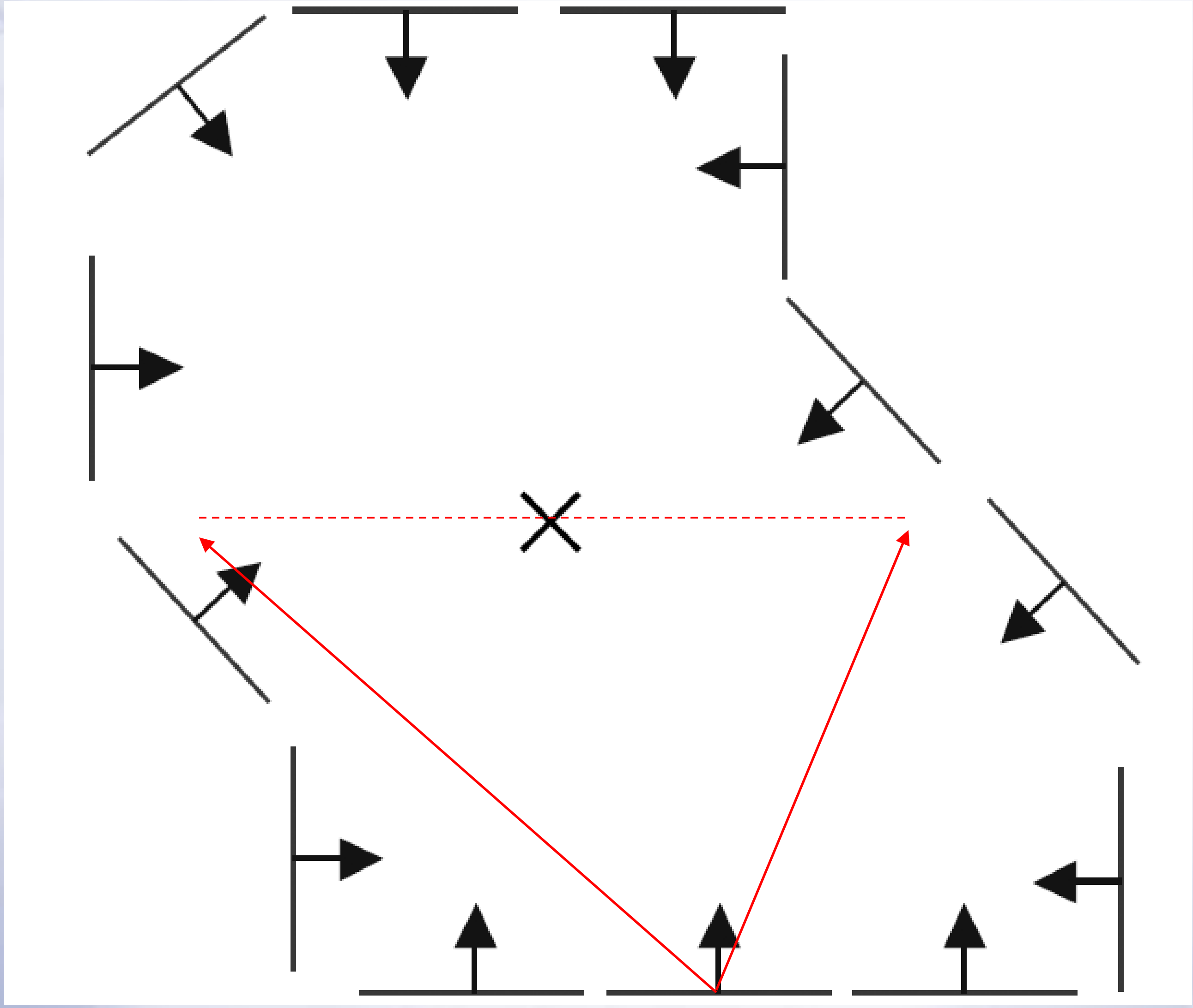
model shape

Example



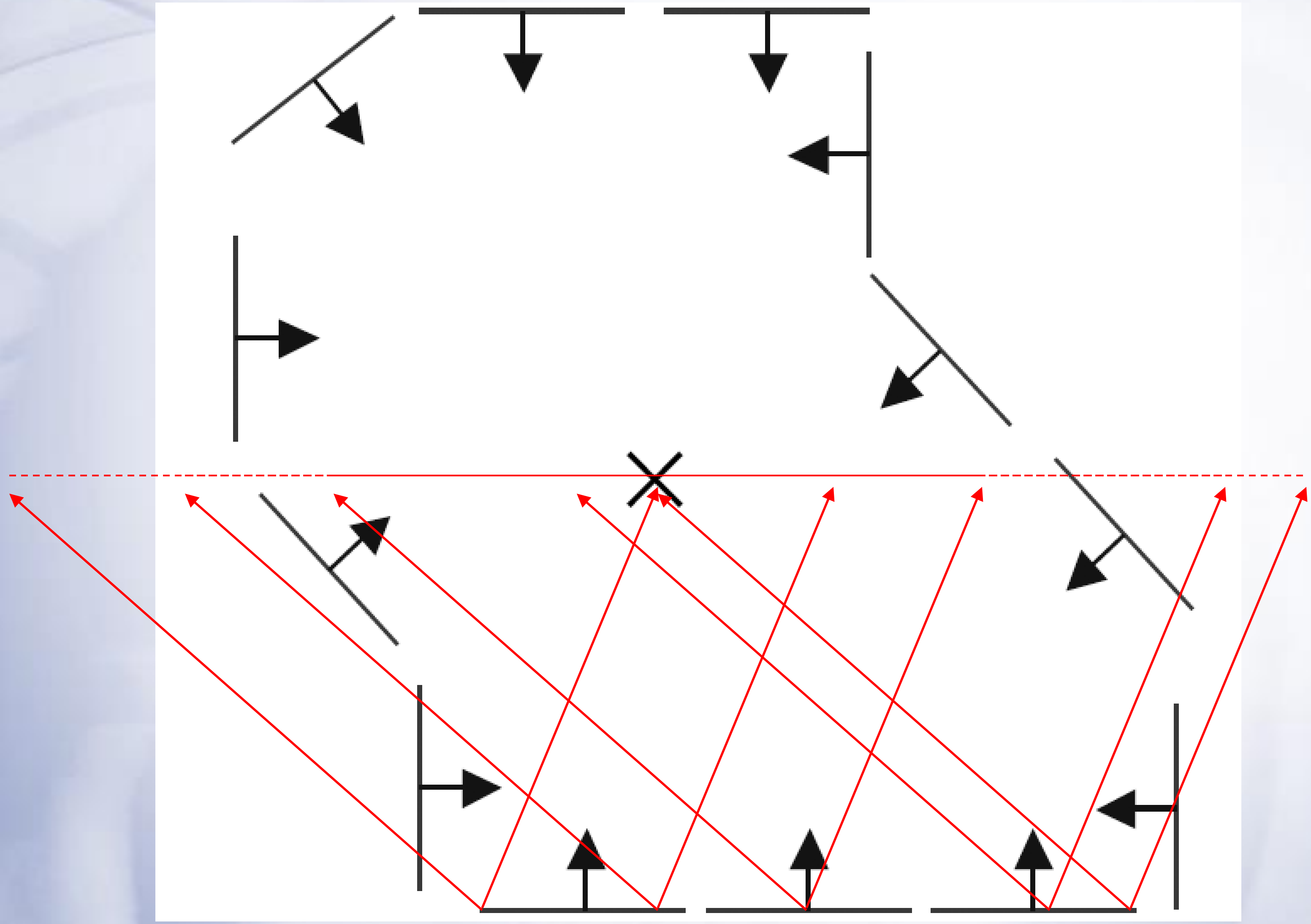
displacement vectors for model points

Example



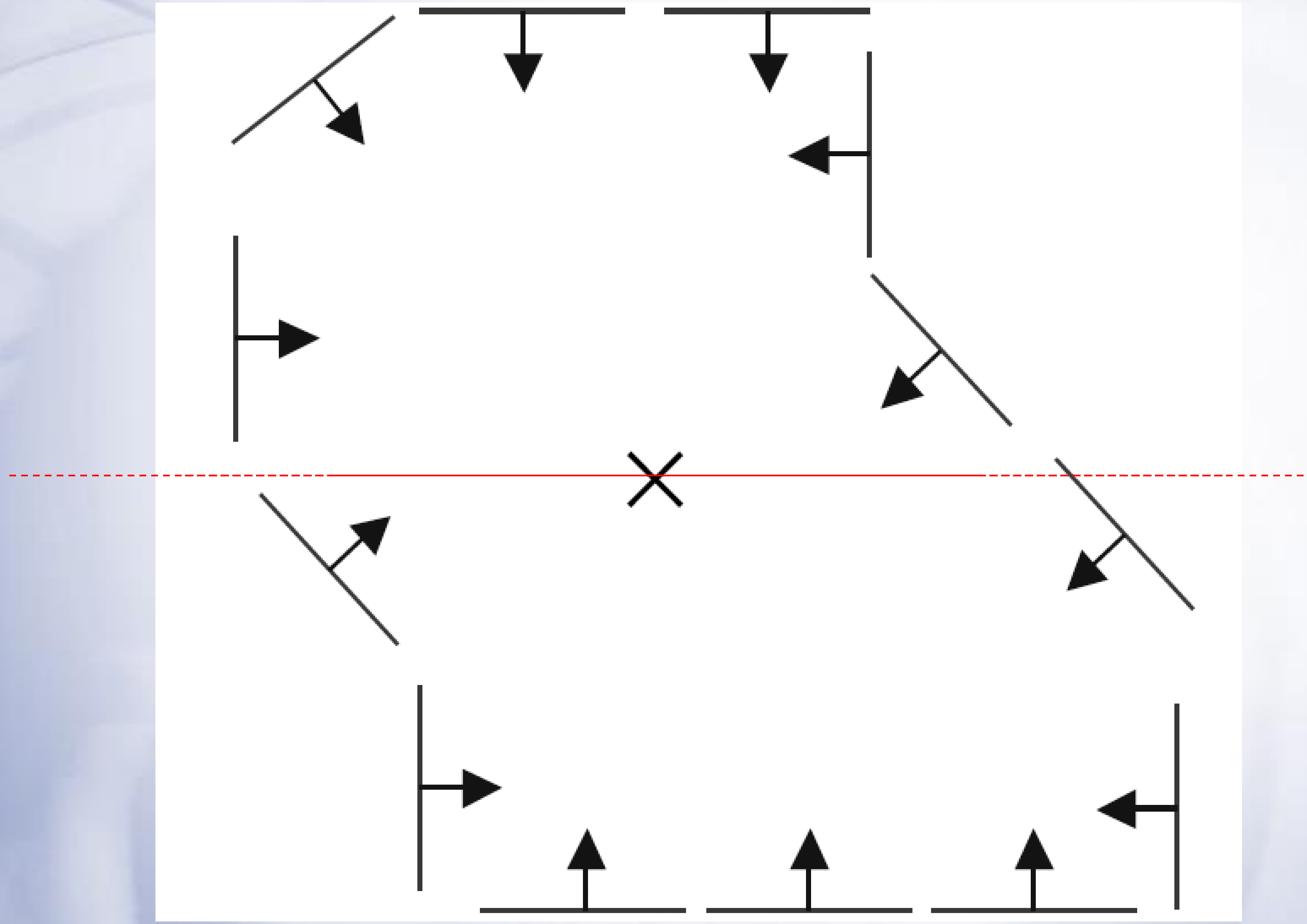
range of voting locations for test point

Example



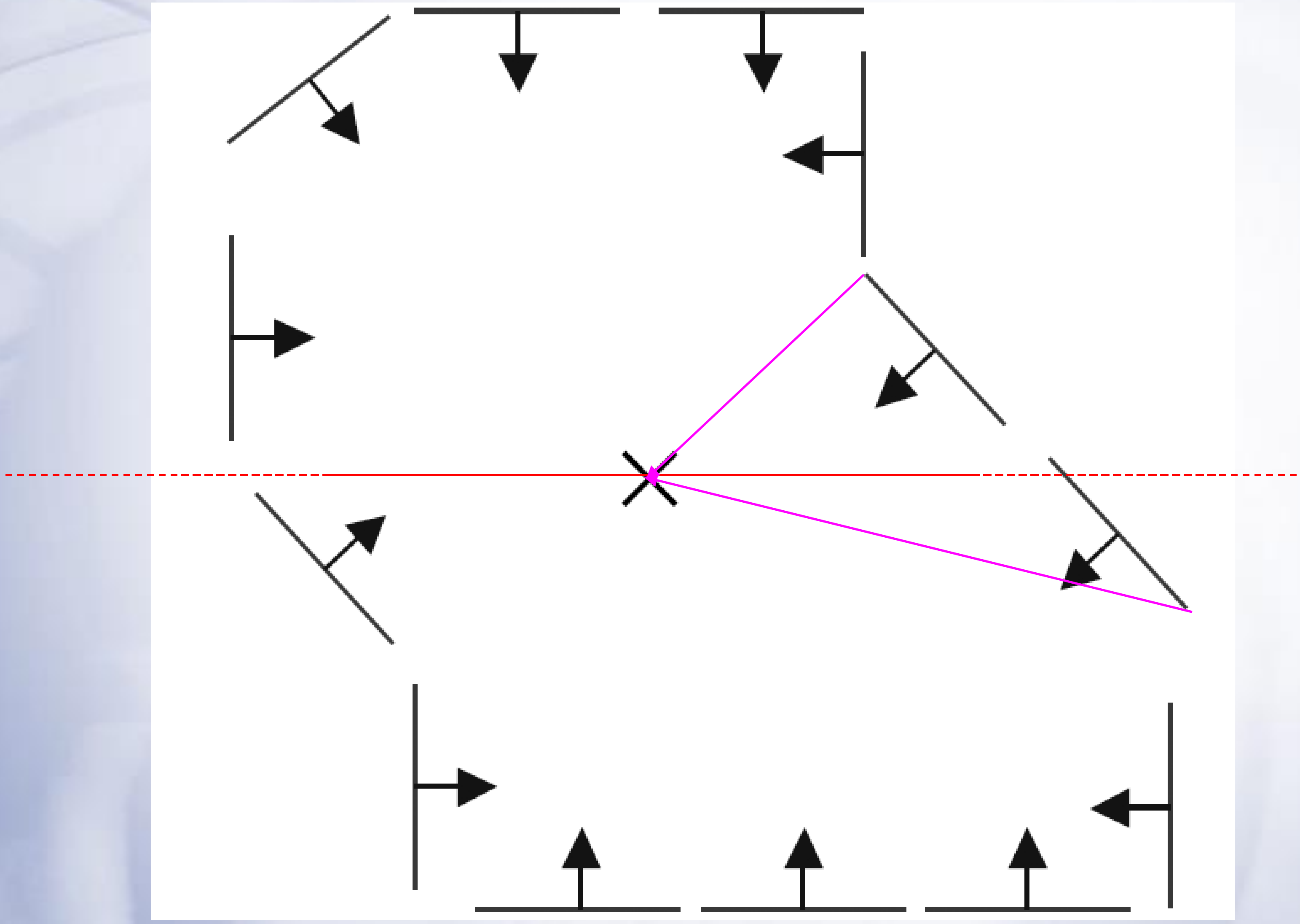
range of voting locations for test point

Example



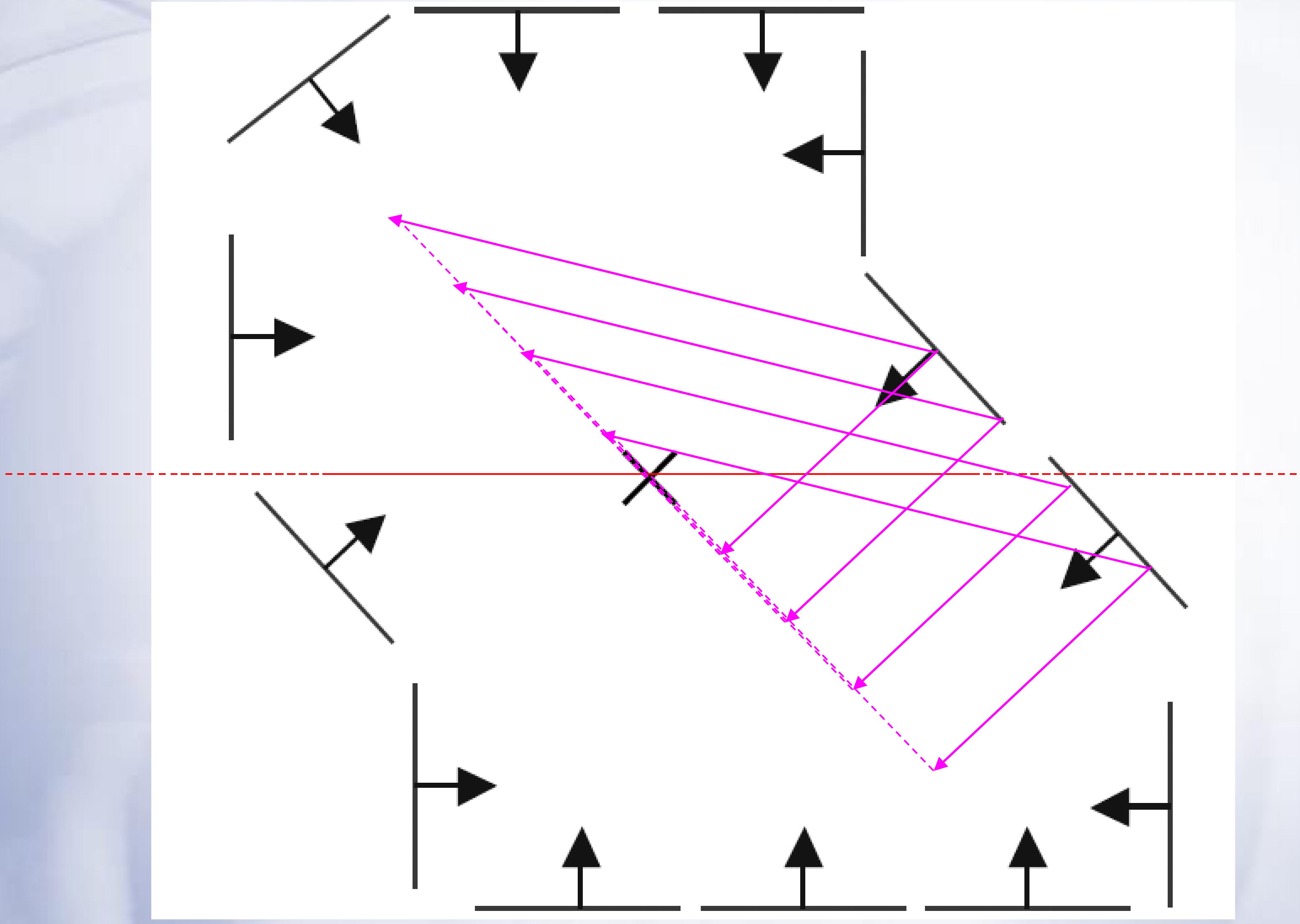
votes for points with $\theta = \uparrow$

Example



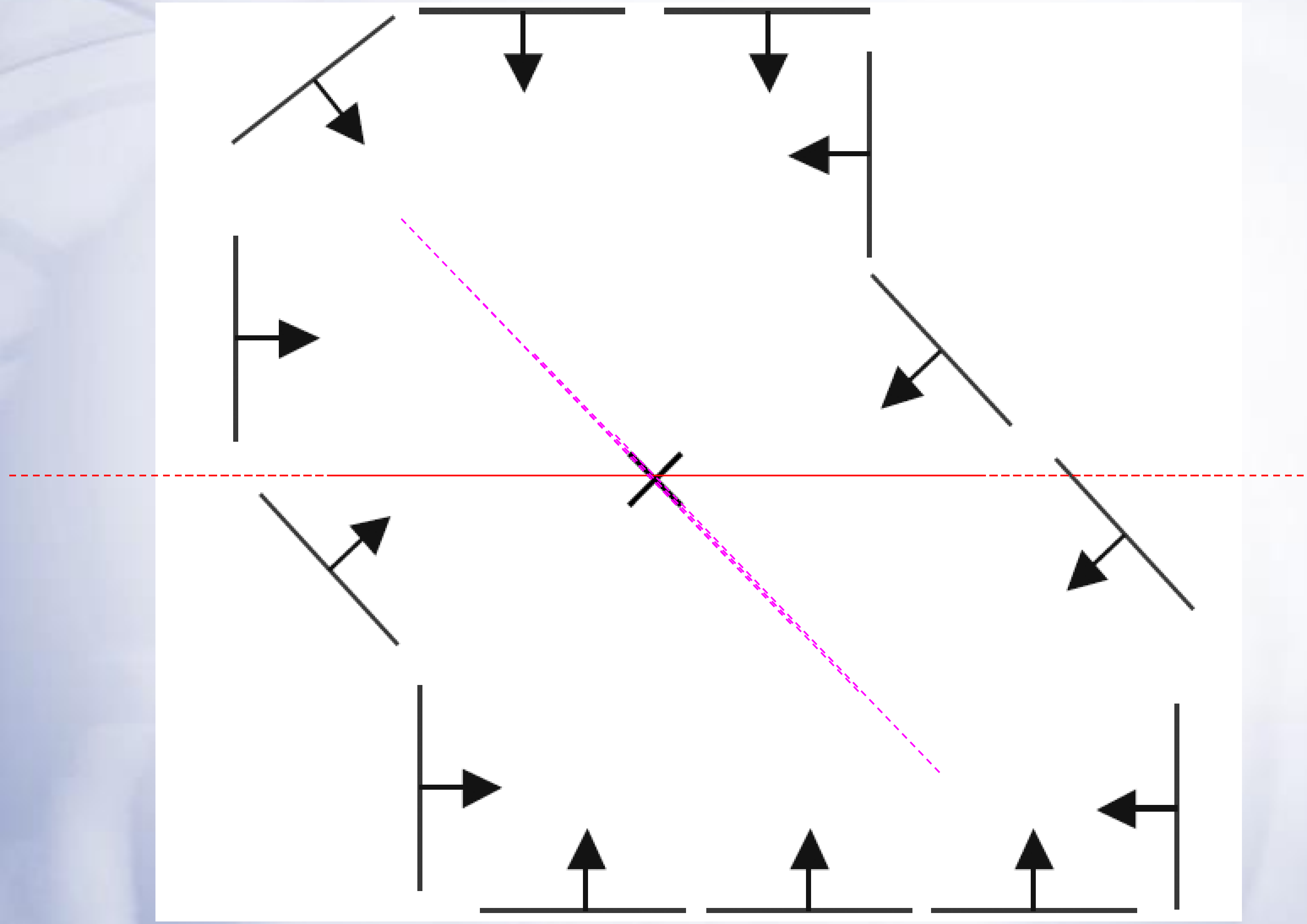
displacement vectors for model points

Example



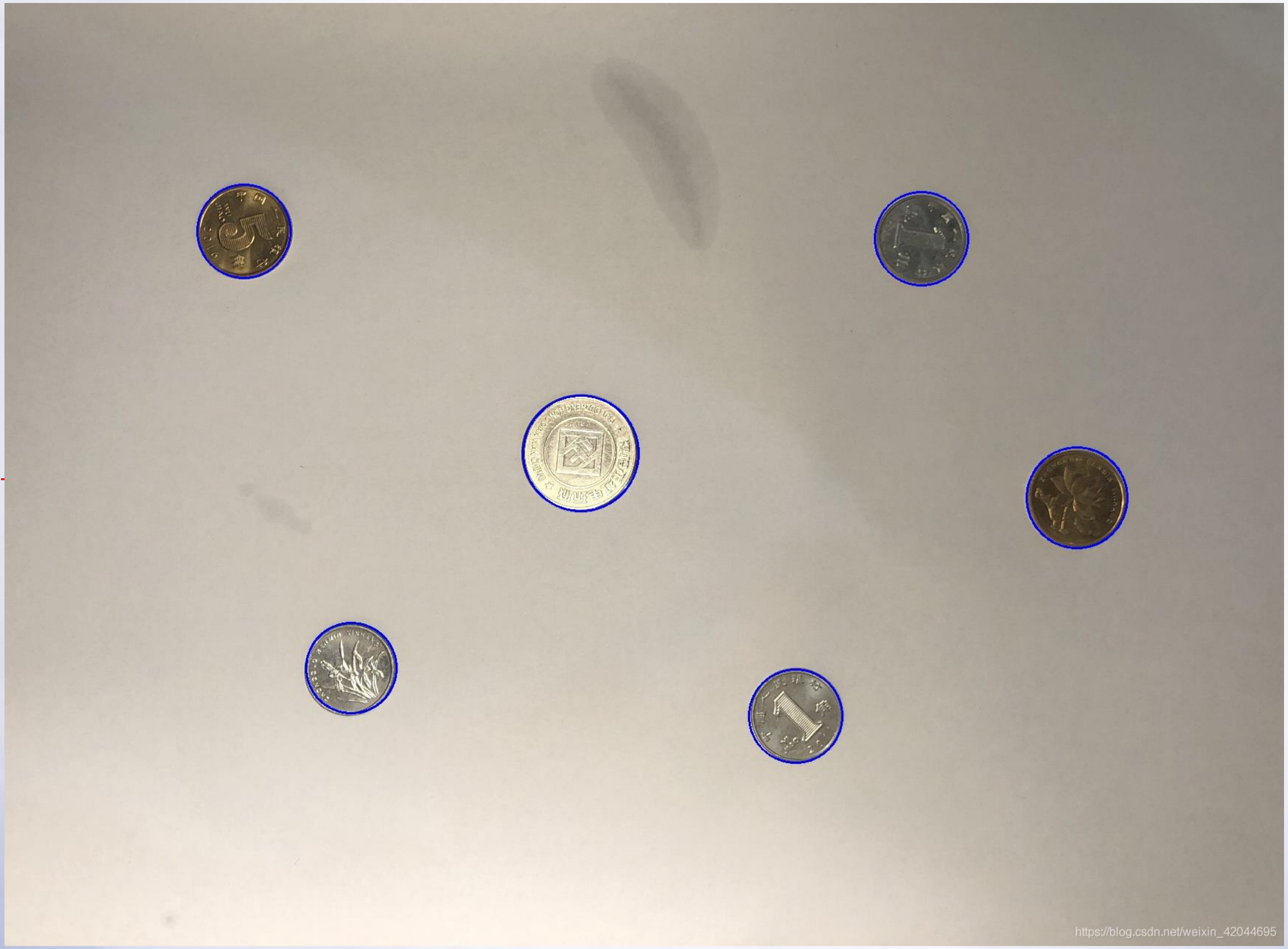
range of voting locations for test point

Example



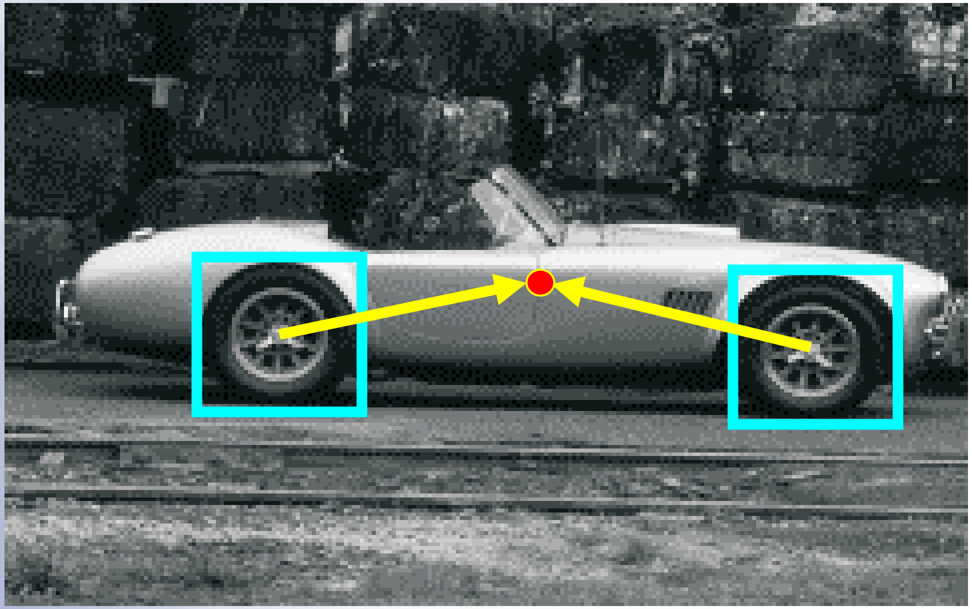
votes for points with θ = ↙

Example

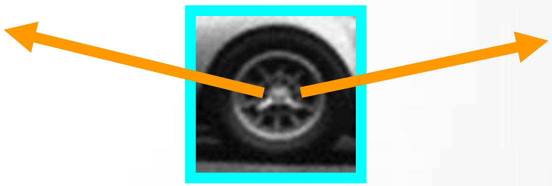


Application in recognition

- Instead of indexing displacements by gradient orientation, index 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

Application in recognition

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

test image

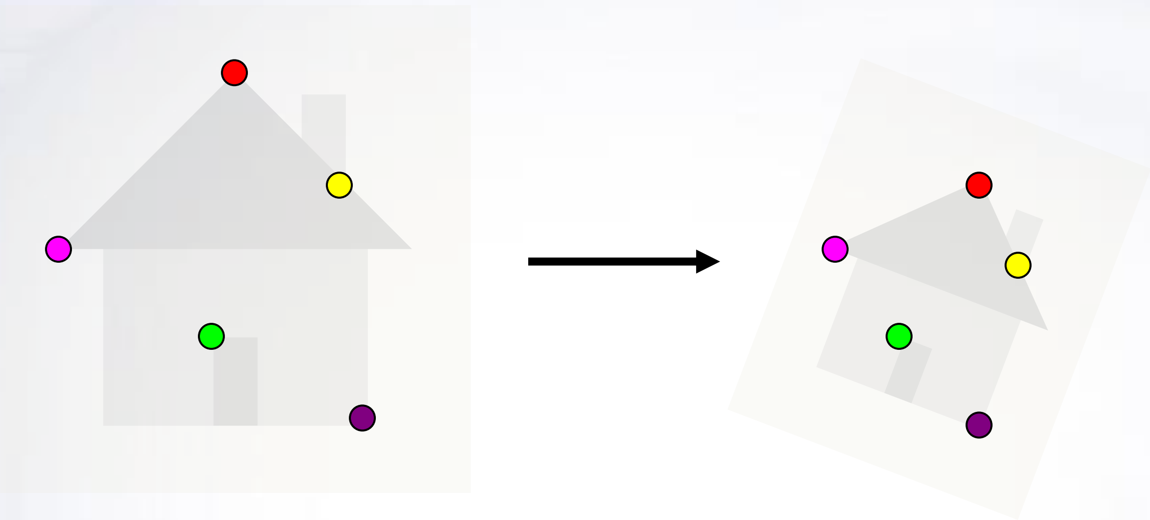
B. Leibe, A. Leonardis, and B. Schiele, [Combined Object Categorization and Segmentation with an Implicit Shape Model](), ECCV Workshop on Statistical Learning in Computer Vision 2004

Overview

- Fitting techniques
  - Least Squares
  - Total Least Squares
  - Robust Fitting
- RANSAC
- Hough Voting

- Alignment as a fitting problem

Image alignment

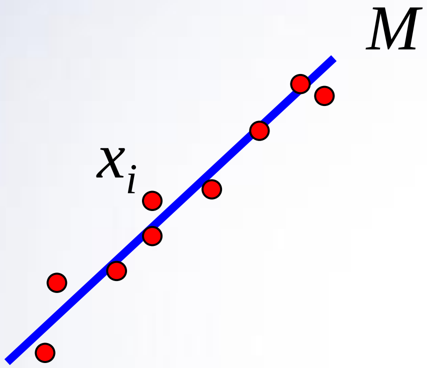


- Two broad approaches:
  - Direct (pixel-based) alignment
    - Search for alignment where most pixels agree
  - Feature-based alignment
    - Search for alignment where *extracted features* agree
    - Can be verified using pixel-based alignment

Source: S. Lazebnik

Alignment as fitting

- Previously: fitting a model to features in one image



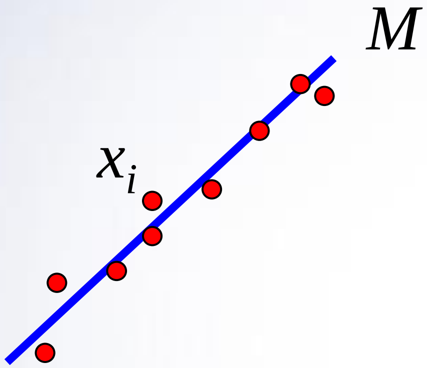
Find model *M* that minimizes

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

Source: S. Lazebnik

Alignment as fitting

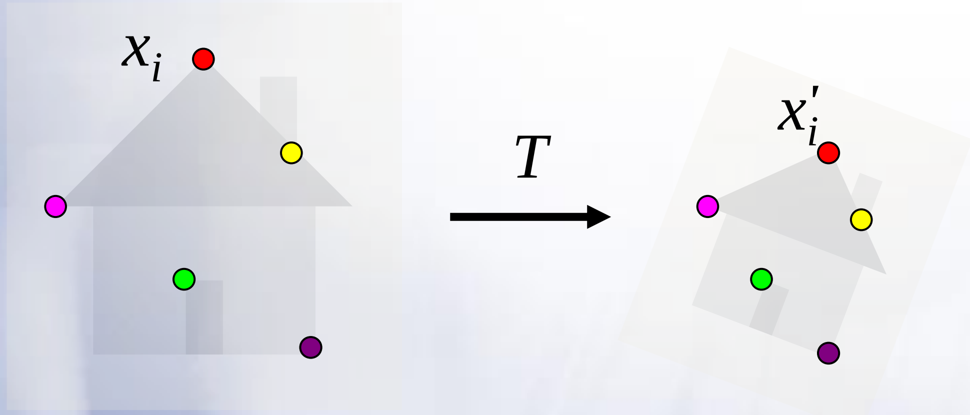
- Previously: fitting a model to features in one image



Find model *M* that minimizes

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

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



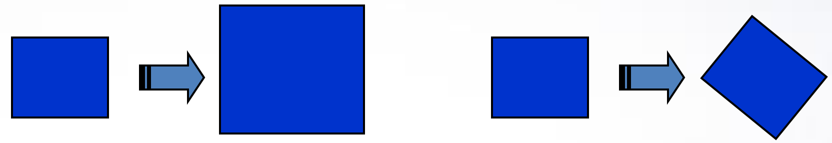
Find transformation *T* that minimizes

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

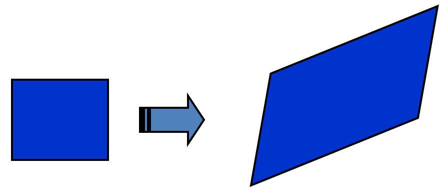
Source: S. Lazebnik

2D transformation models

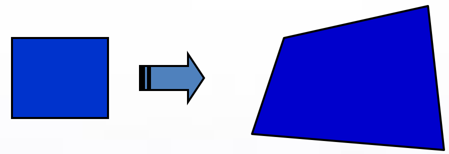
- Similarity (translation, scale, rotation)



- Affine



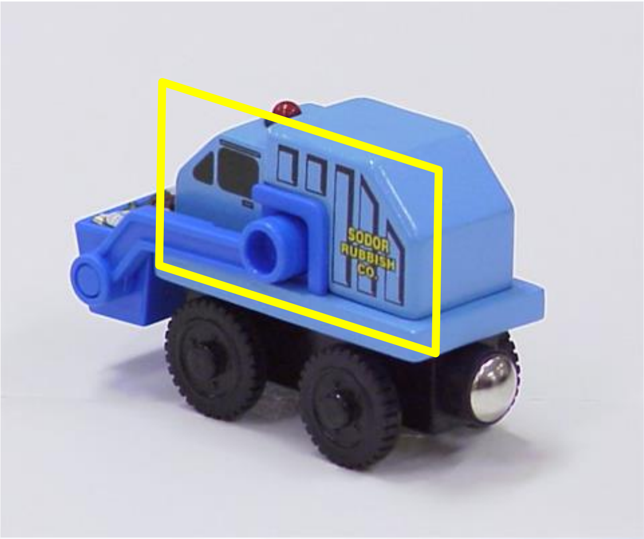
- Projective (homography)



Source: S. Lazebnik

Let’s start with affine transformations

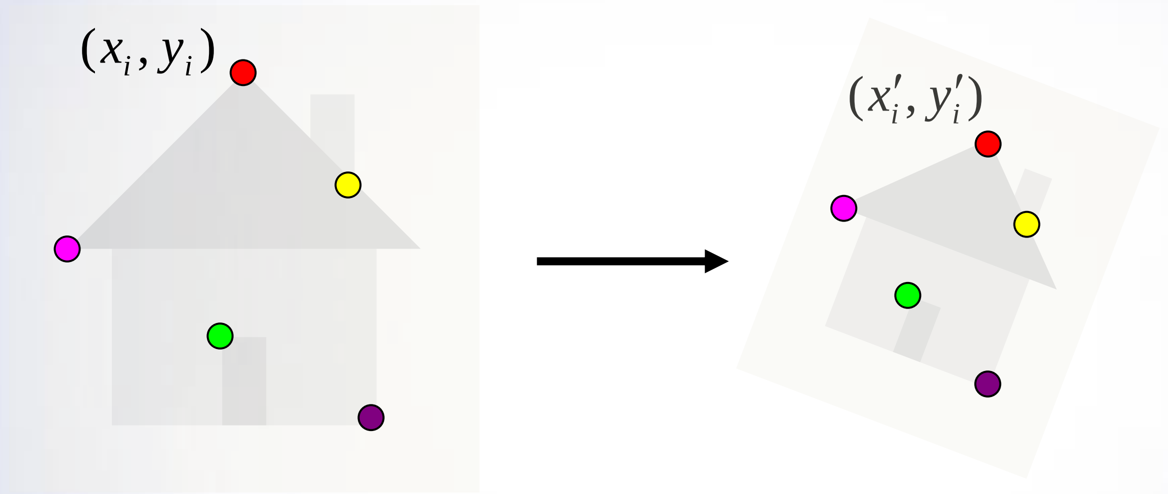
- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used to initialize fitting for more complex models



Source: S. Lazebnik

Fitting an affine transformation

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



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

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

Source: S. Lazebnik

Fitting an affine transformation

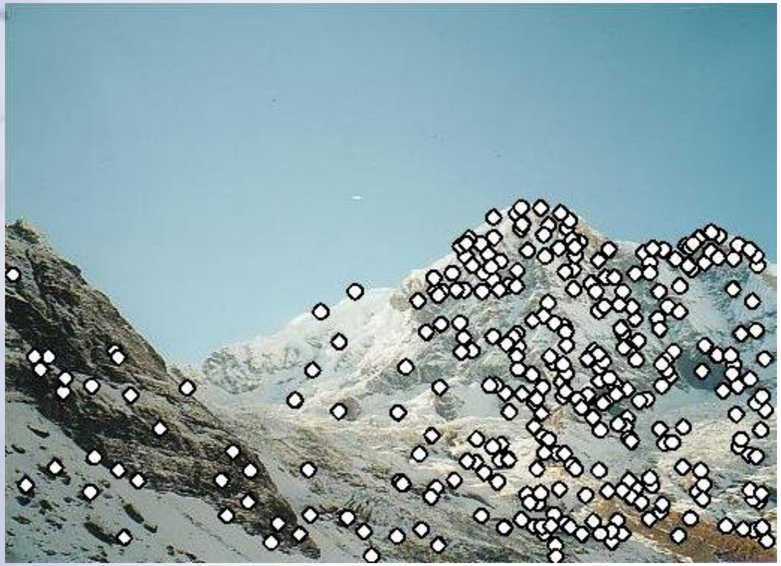
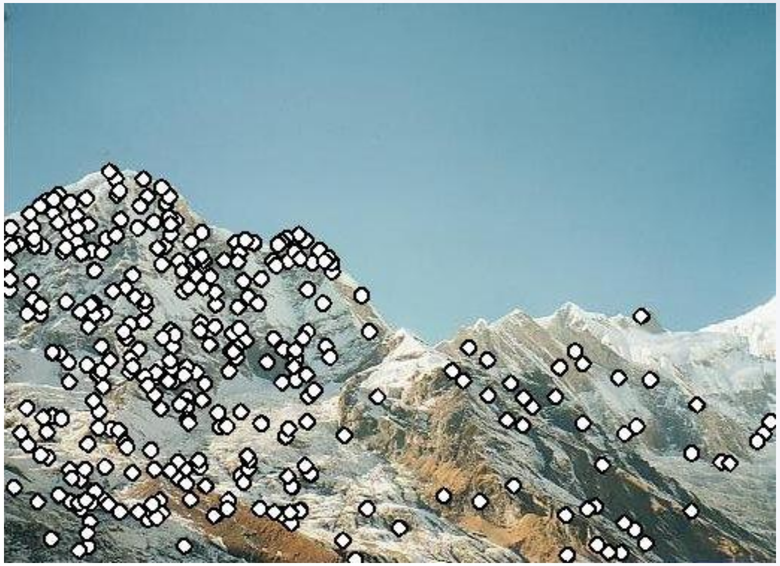
$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

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

Source: S. Lazebnik

Feature-based alignment outline

Feature-based alignment outline

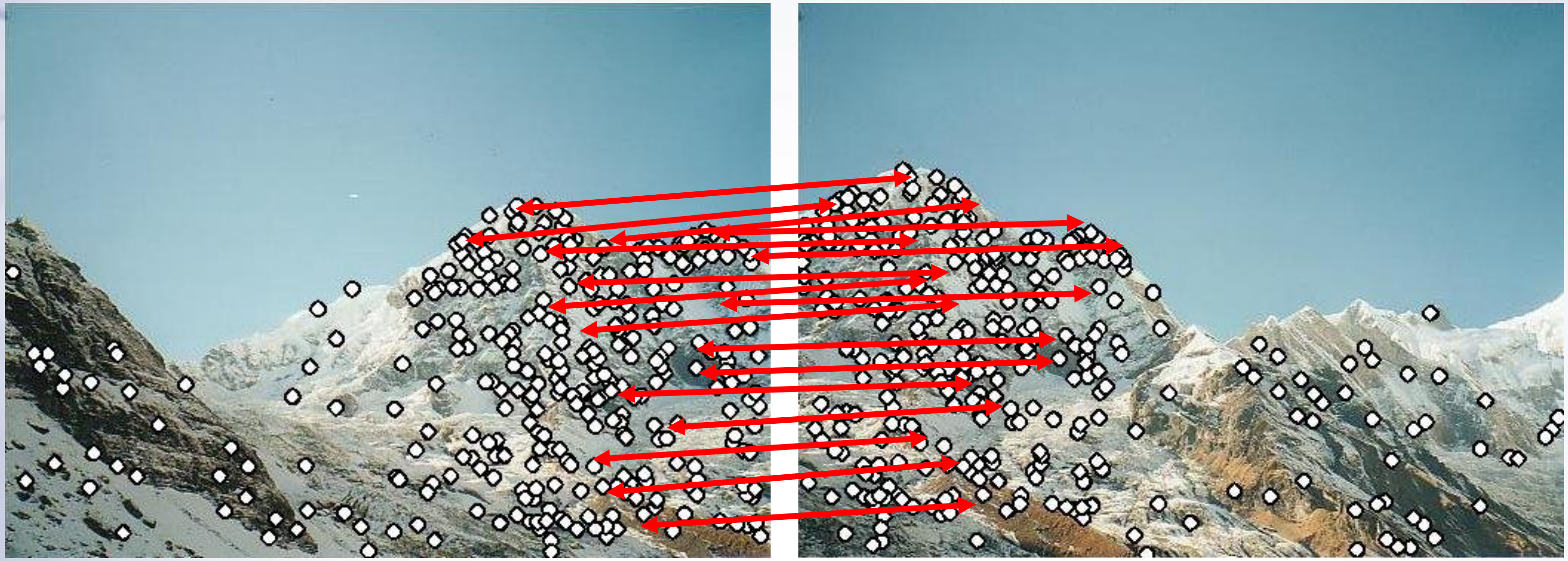
 

- Extract features

Feature-based alignment outline

- Extract features
- Compute *putative matches*

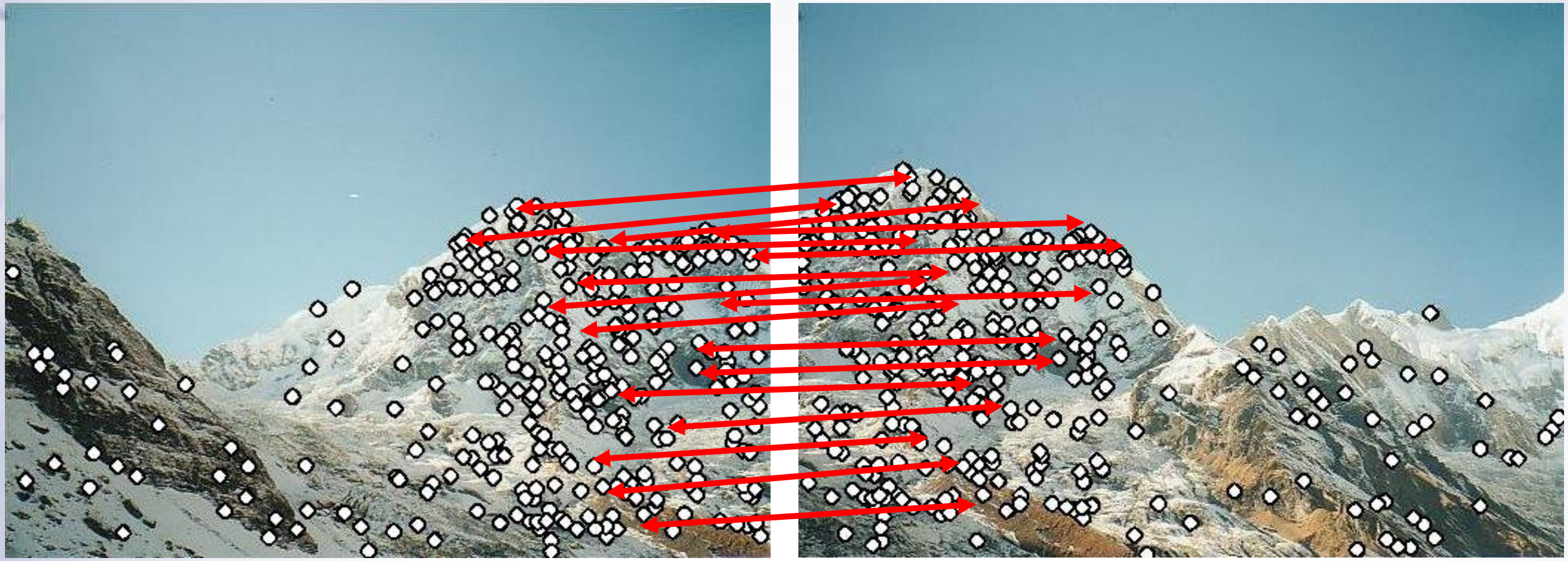
估计单应性矩阵



- 在特征匹配中，我们最终要得到一个3*3的单应性矩阵。通常令h33=1来归一化矩阵，因此单应性矩阵有8个自由度h11-h32，求这八个未知数，至少要包含四个匹配点对。

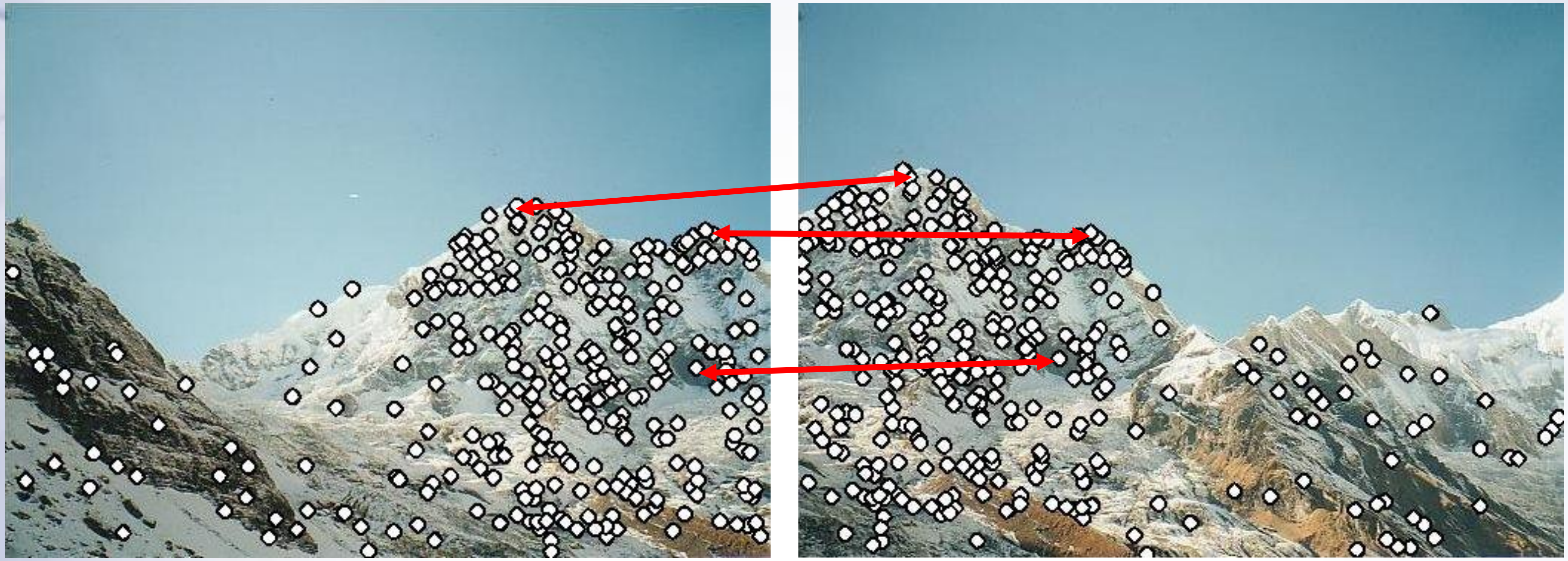
$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

估计单应性矩阵



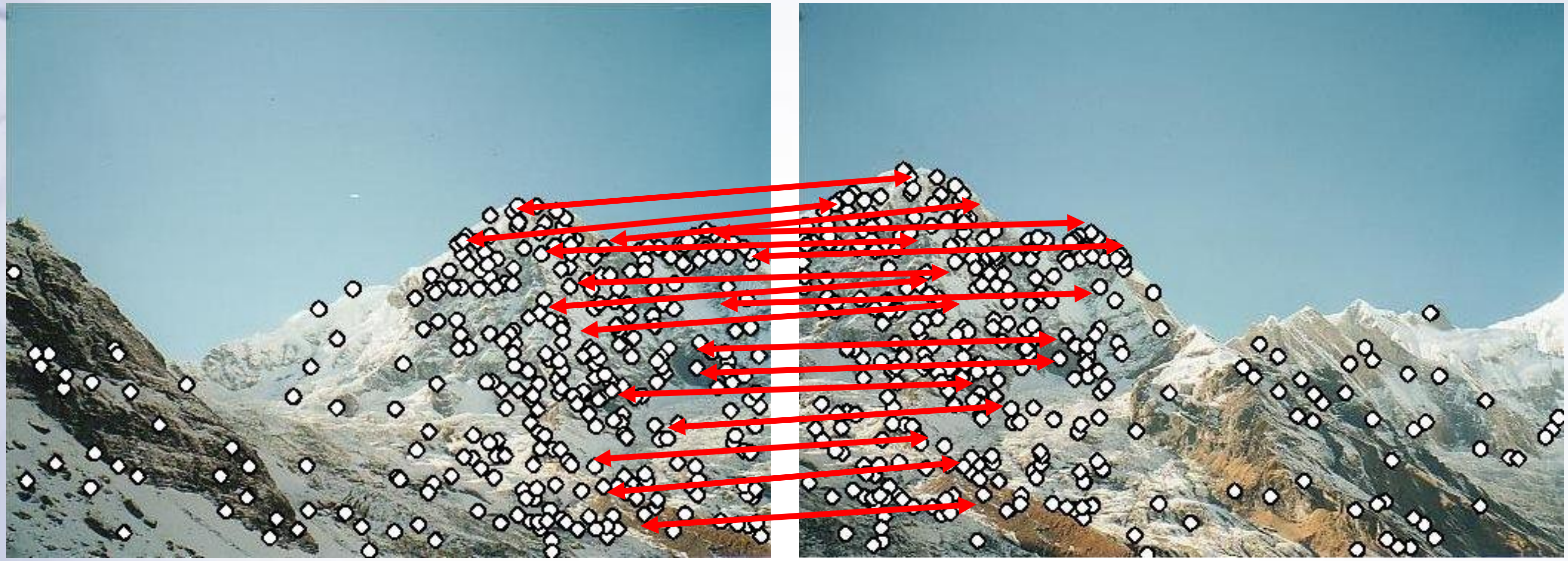
- 1、首先在得到的匹配点中，随机选择4个匹配点对（不共线），其他匹配点为外点。
- 2、根据4对内点计算单应性矩阵。
- 3、根据此矩阵来测试其他匹配点(计算的是其他匹配点与该模型的投影误差），并设置阈值，若小于为新内点，若大于则为外点，也就是误匹配对，因此通过计算出的单应性矩阵，就能实现一次误匹配点的剔除。
- 4、将所有的内点统计进行内点更新，在此基础上再次进行步骤3，迭代M次，最终得到含有内点最多的模型，此时模型为最优模型，也就是我们最终所需要的单应性矩阵。

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*)

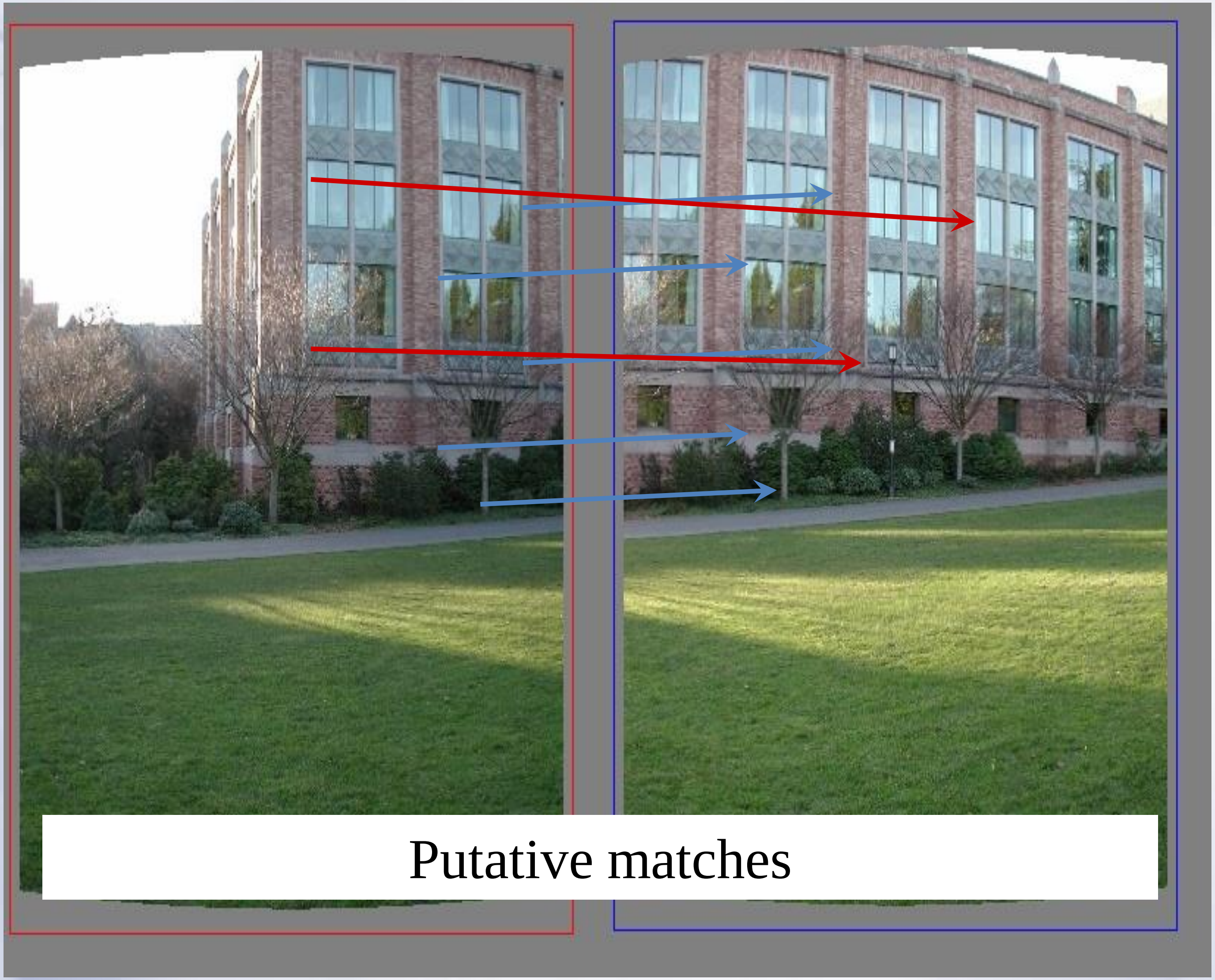
Feature-based alignment outline

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

Dealing with outliers

- The set of putative matches contains a very high percentage of outliers
- Geometric fitting strategies:
  - RANSAC
  - Hough transform

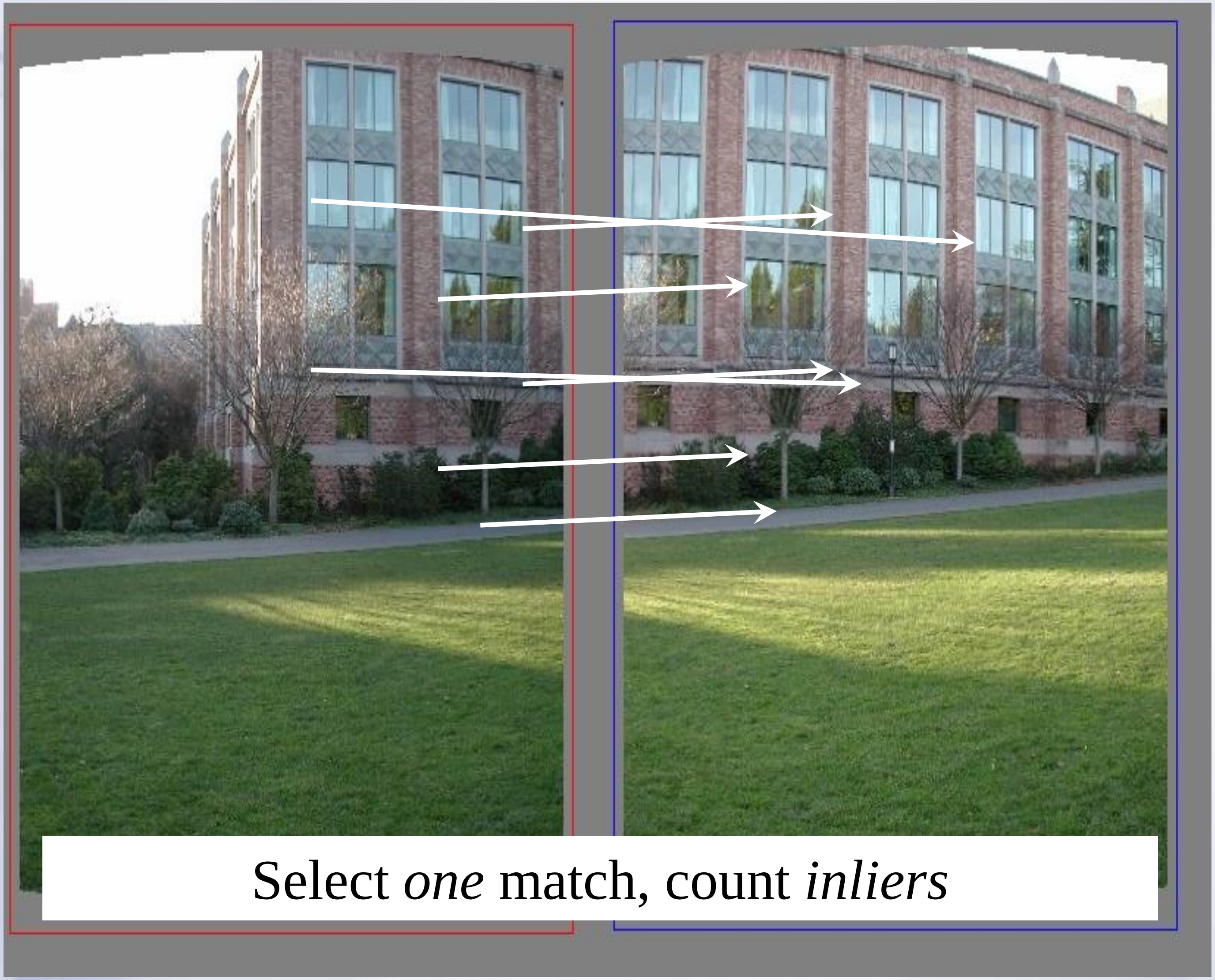
RANSAC example: Translation



Putative matches

Source: A. Efros

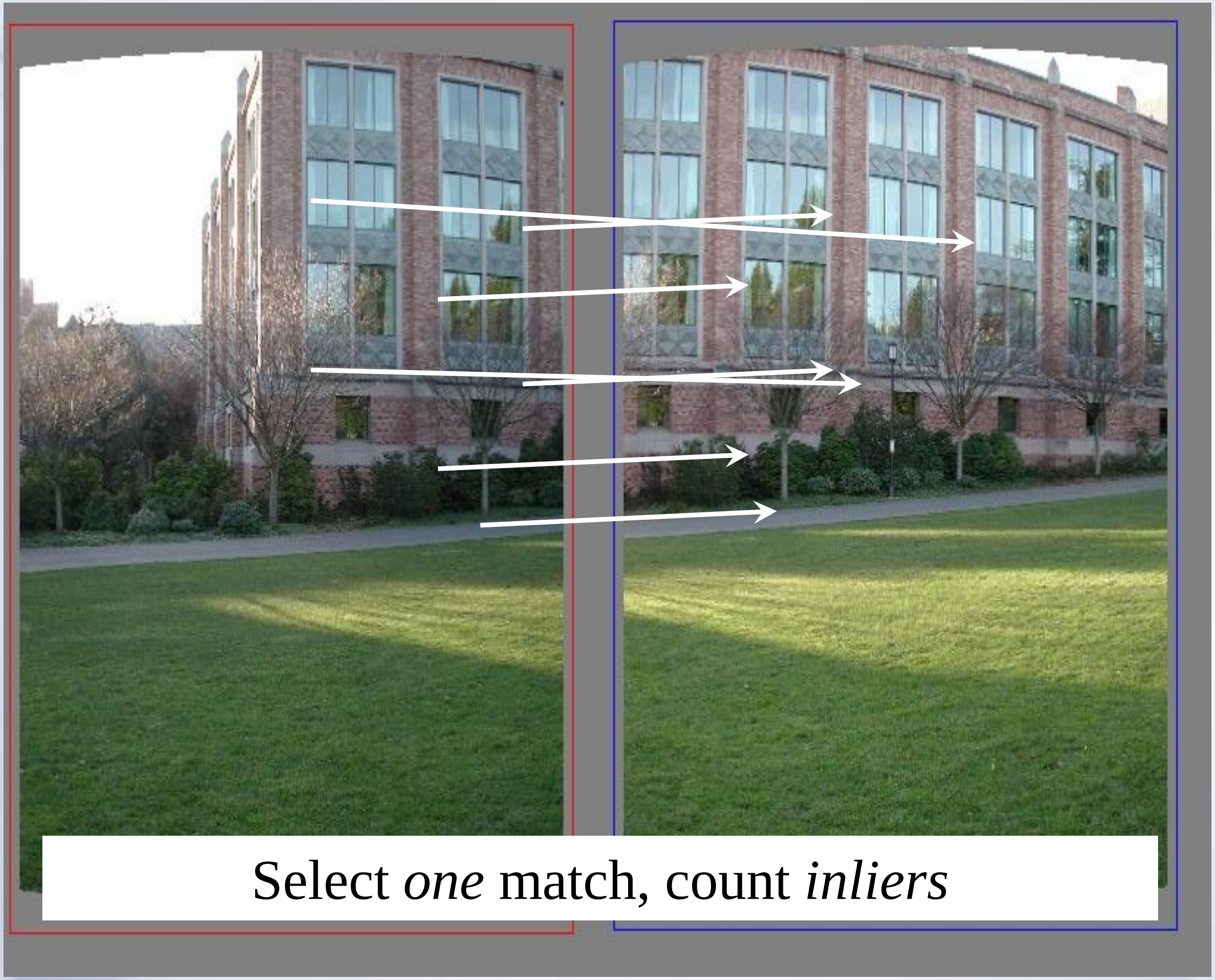
RANSAC example: Translation



Select *one* match, count *inliers*

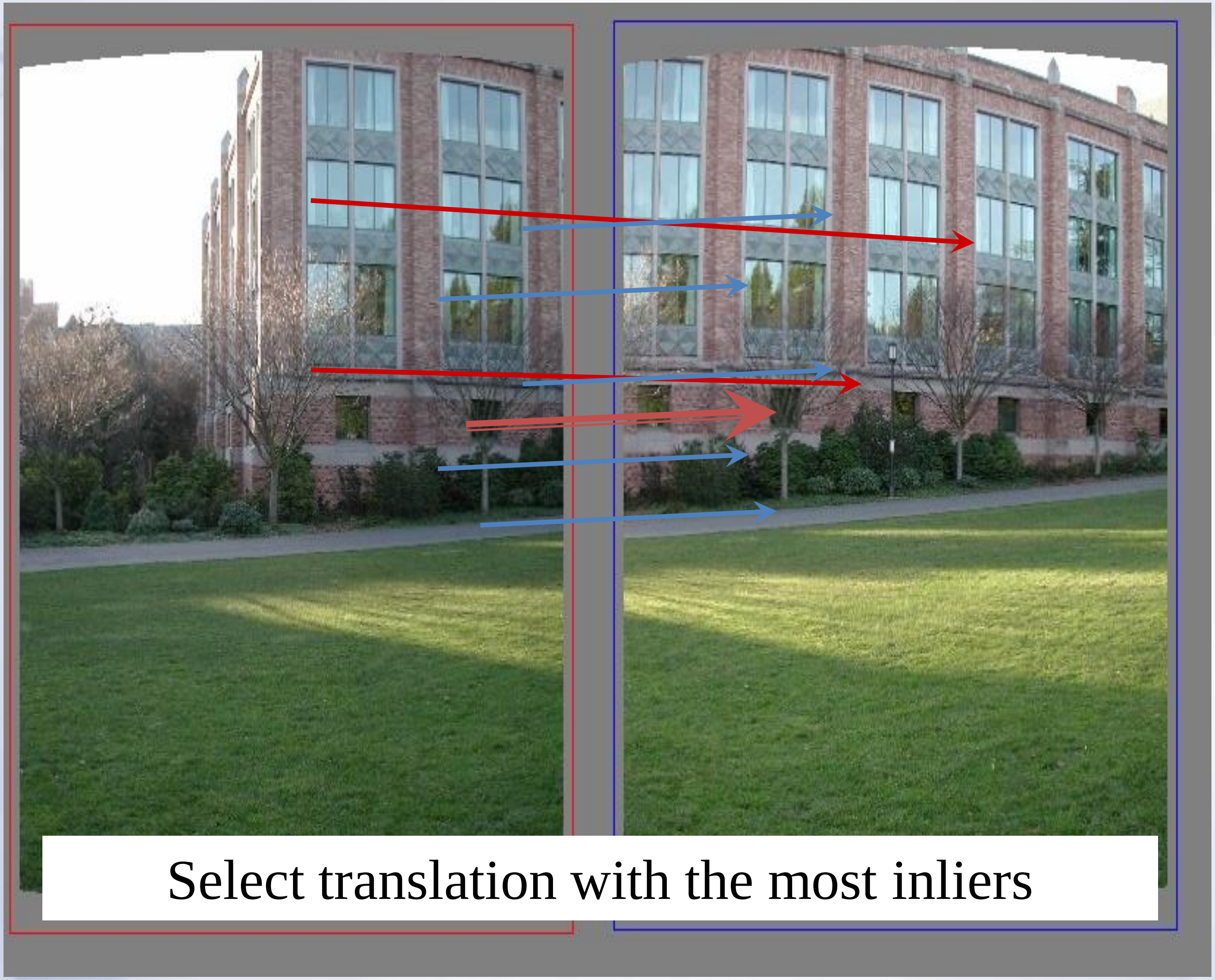
Source: A. Efros

RANSAC example: Translation



Source: A. Efros

RANSAC example: Translation



Source: A. Efros

作业：

1. 编程实现Ransec算法
2. 编程实现基于Hough变换的直线检测算法