Canney Edge Detector 边缘检测，使用卷积运算，
只能返回一堆像素点，意义不大，比如一堆点，形成了一个圆
我们更希望得到这个圆的方程
则需要做 Fitting

Fitting

(Least squares & RANSAC)

最小二乘　　随机采样一致性

Lu Peng

School of Computer Science,

Beijing University of Posts and Telecommunications

0

Machine Vision Technology							
Semantic information				**Metric 3D information**			
Pixels	Segments	Images	Videos	Camera		Multi-view Geometry	
Convolutions Edges & Fitting Local features Texture	Segmentation Clustering	Recognition Detection	Motion Tracking	Camera Model	Camera Calibration	Epipolar Geometry	SFM
10	4	4	2	2	2	2	2

1

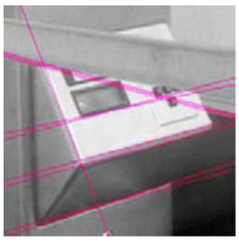
Fitting

- We've learned how to detect edges. 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

Source: S. Lazebnik

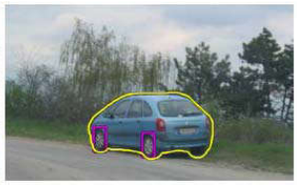
Fitting

- 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

Source：S. Lazebnik

Fitting: Overview

- If we know which points belong to the line, how do we find the "optimal" line parameters?
  - Least squares
- What if there are outliers?
  - Robust fitting, RANSAC
- What if there are many lines?
  - Voting methods: RANSAC, Hough transform
- What if we're not even sure it's a line?
  - Model selection

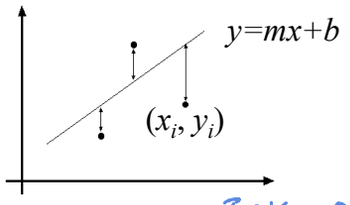
Source：S. Lazebnik

Least squares line fitting

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

Line equation: $y_i = m x_i + b$

Find (m, b) to minimize



Loss 函数

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

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

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

这里 $Y^T XB = (XB)^T Y$ 结果都是系数

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

求导（梯度）

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

(一阶导) ← 此题为，求极值点时的B

$$X^T XB = X^T Y$$

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

Source：S. Lazebnik

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

这里手动从一下子得出B

计算机一般用梯度下降来寻找B

Problem with “vertical” least squares

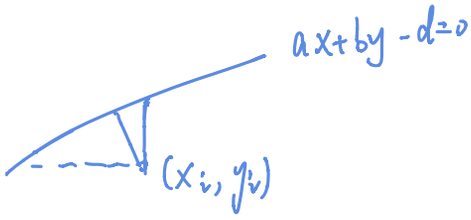
- Not rotation-invariant
- Fails completely for vertical lines

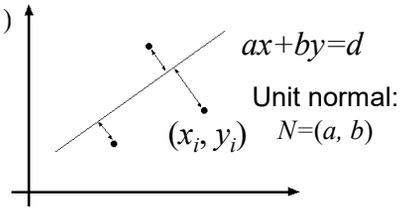
Source：S. Lazebnik

Total least squares

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

$$|ax_i + by_i - d|$$





Source：S. Lazebnik

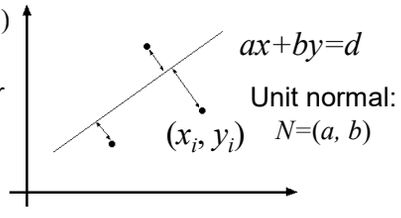
Total least squares

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

$$|ax_i + by_i - d|$$

Find (a, b, d) to minimize the sum of squared perpendicular distances

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



Source：S. Lazebnik

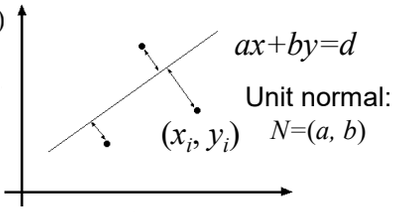
Total least squares

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

$$|ax_i + by_i - d|$$

Find (a, b, d) to minimize the sum of squared perpendicular distances

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



$$\frac{\partial E}{\partial d} = \sum_{i=1}^{n} -2(ax_i + by_i - d) = 0$$

$$d = \frac{a}{n}\sum_{i=1}^{n} x_i + \frac{b}{n}\sum_{i=1}^{n} y_i = a\bar{x} + b\bar{y}$$

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

$U_{n\times 2} \cdot N_{2\times 1}$

这里求 N，即 (a, b)

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

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

Source: S. Lazebnik

Total least squares

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

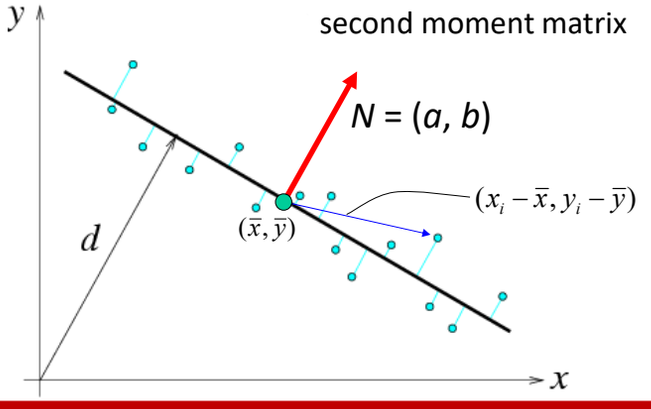
second moment matrix

Source: S. Lazebnik

Total least squares

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

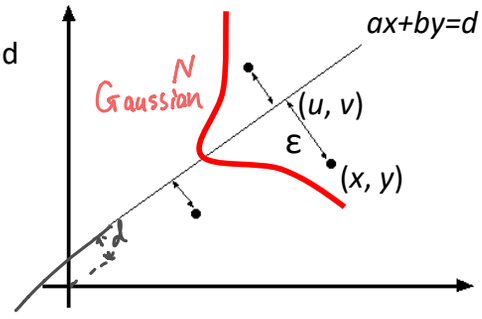
second moment matrix



Source: S. Lazebnik

Least squares as likelihood maximization

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



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

point on the line

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

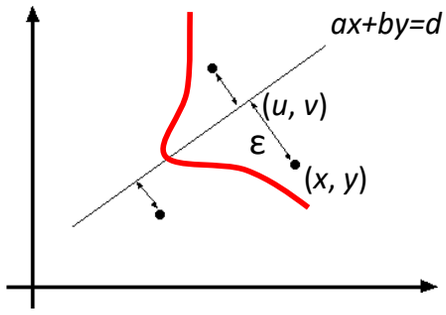
normal direction

Source: S. Lazebnik

Least squares as likelihood maximization

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

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



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

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

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

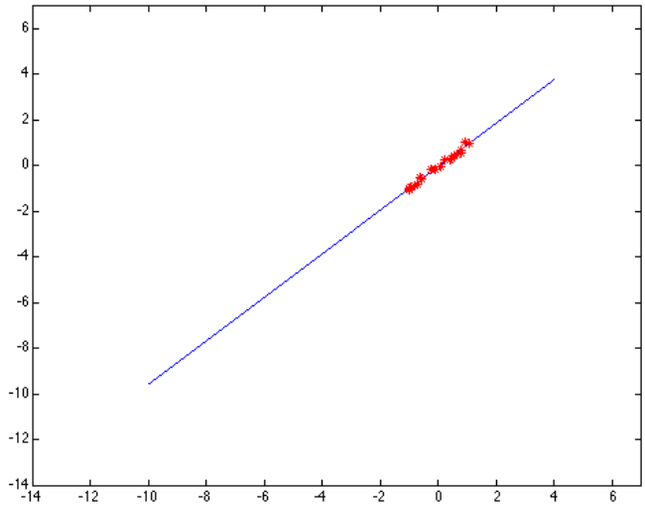
Source: S. Lazebnik

14

这里我是站在概率的角度理解，用 least squares 找边的问题
因为真实值周围出现的 noise 偏离度是有概率 P 的，P~N
越接近真实值，那这个点在正态分布中的位置，越靠中间
用这些点的数据去找到一组正态分布参数 (μ, σ)，则中间位置与法向量的垂线，就是我们
要拟合的直线，即
真实边

Least squares: Robustness to noise

Least squares fit to the red points:

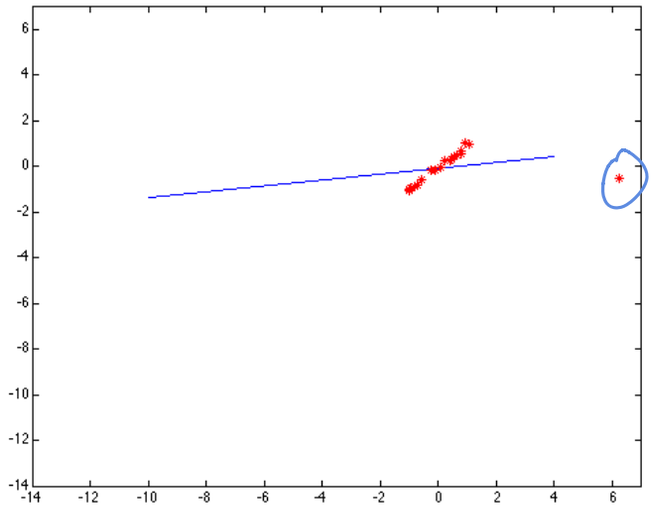


Source: S. Lazebnik

15

Least squares: Robustness to noise

Least squares fit with an outlier:



这个点偏离度大 导致least square fit不好 应该降低它的贡献度

Problem: squared error heavily penalizes outliers

Source：S. Lazebnik

Robust estimators

- General approach: find model parameters θ that minimize

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

点到直线的距离

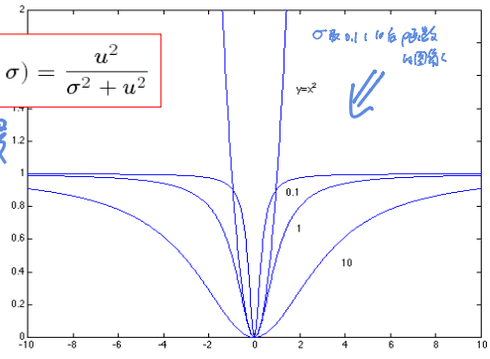
用 $\rho(u;\sigma)$ 函数来返回 u 的贡献度

$r_i(x_i, \theta)$– residual of ith point w.r.t. model parameters θ

ρ – robust function with scale parameter σ

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

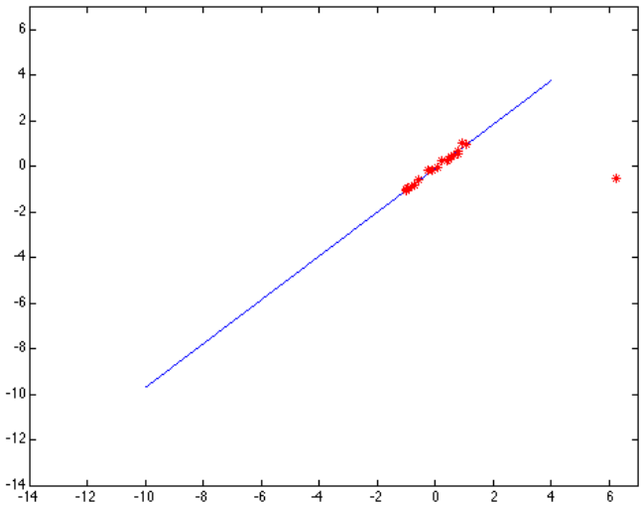
σ↑ ρ上升越慢



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

Source：S. Lazebnik

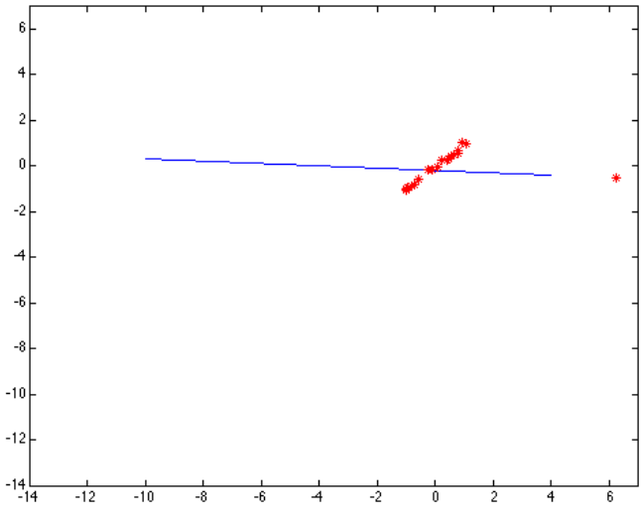
Choosing the scale: Just right



The effect of the outlier is minimized

Source：S. Lazebnik

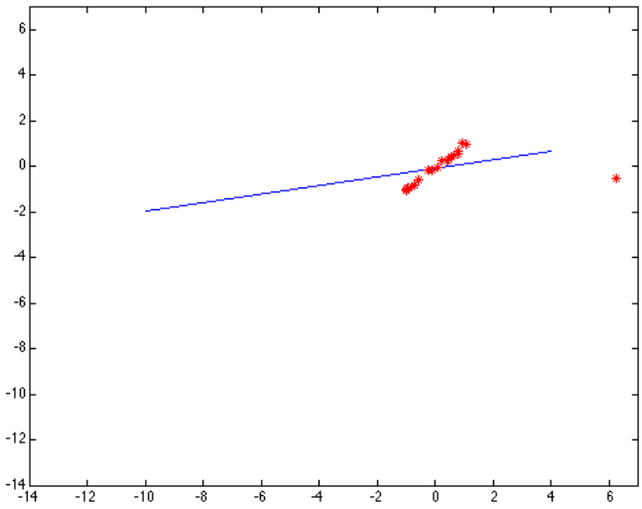
Choosing the scale: Too small



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

Source：S. Lazebnik

Choosing the scale: Too large



Behaves much the same as least squares

Source：S. Lazebnik

Robust estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Least squares solution can be used for initialization
- Adaptive choice of scale: approx. 1.5 times median residual (F&P, Sec. 15.5.1)

Source：S. Lazebnik

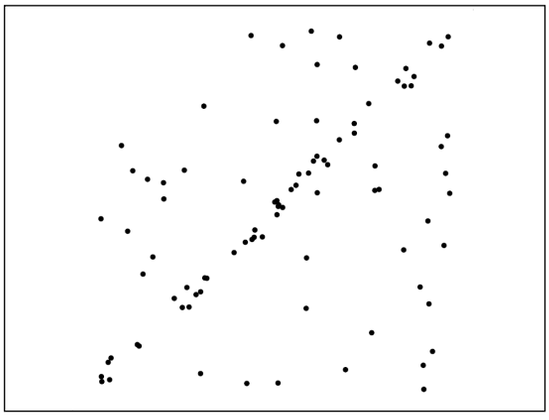
RANSAC

- Robust fitting can deal with a few outliers – what if we have very many? Robust 当 noise 很多时，不适用
- 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.

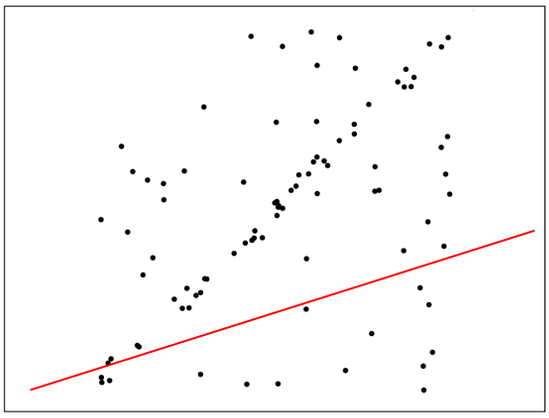
Source：S. Lazebnik

RANSAC for line fitting example



Source: R. Raguram

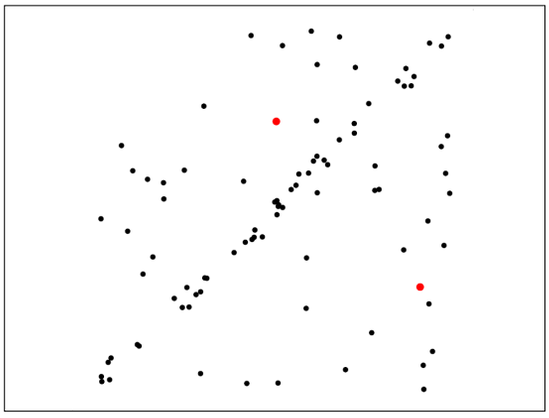
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

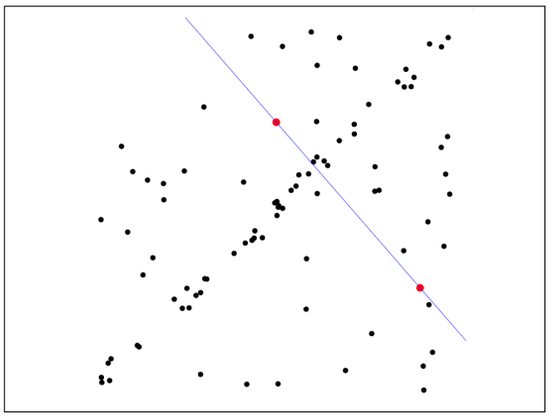
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

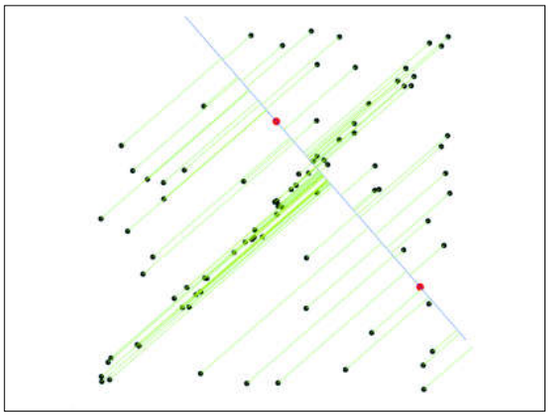
RANSAC for line fitting example



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

Source: R. Raguram

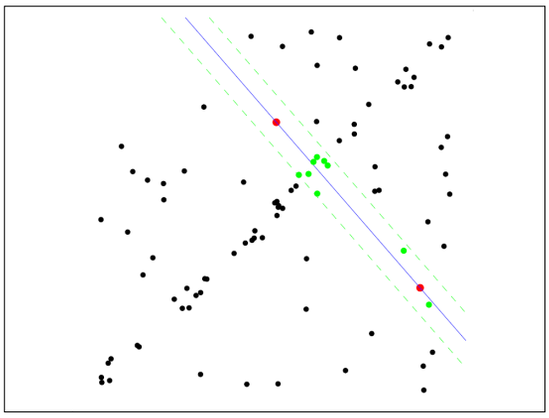
RANSAC for line fitting example



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

Source: R. Raguram

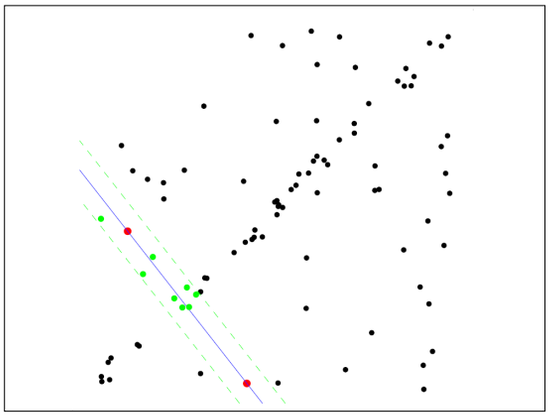
RANSAC for line fitting example



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

Source: R. Raguram

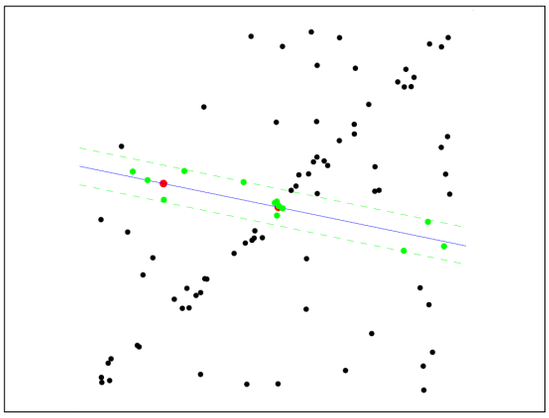
RANSAC for line fitting example



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

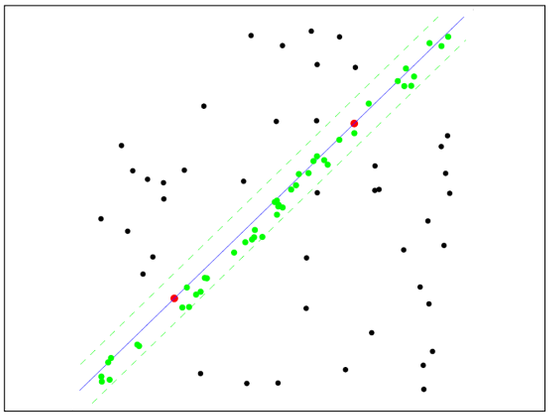


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

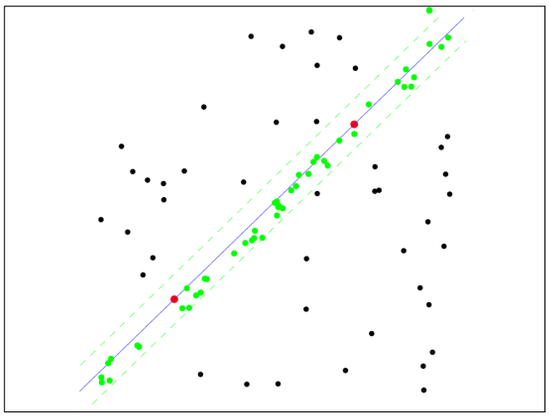
Uncontaminated sample



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: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

Source: M. Pollefeys

Choosing the parameters

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

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

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

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

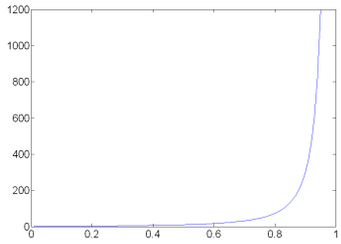
Source: M. Pollefeys

Choosing the parameters

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

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

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



Source: M. Pollefeys

Choosing the parameters

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

Source: M. Pollefeys

Adaptively determining the number of samples

- 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*=∞, *sample_count* =0
  - While *N* >*sample_count*
    - Choose a sample and count the number of inliers
    - Set e = 1 – (number of inliers)/(total number of points)
    - Recompute *N* from *e:*

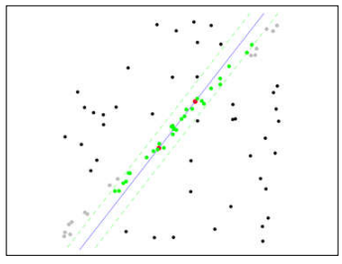
$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

    - Increment the *sample_count* by 1

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

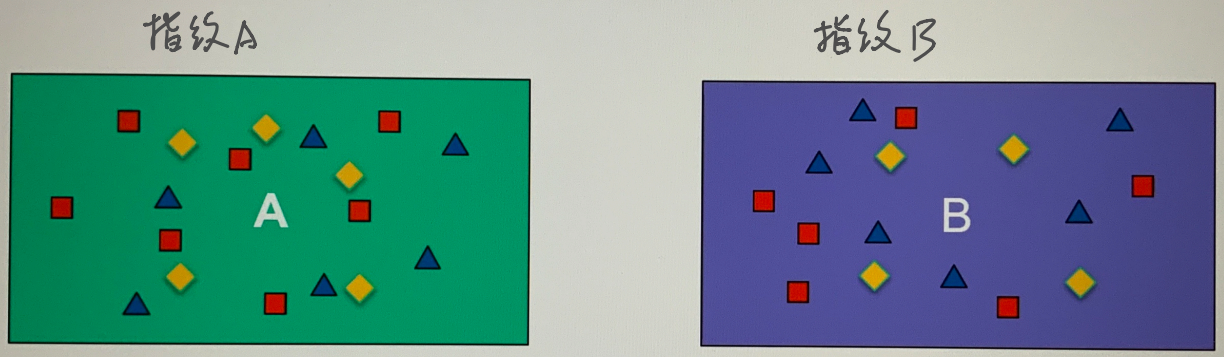


最后还要使用 least square 从这组评分最高的点中来重新拟合

Source：S. Lazebnik

视频里这边，PPT上没有

RANSAC 的一种应用场景



形状表示指纹的某种细节类型（eg. 方形：分岔 菱形：圆弧 …）

现在想求A与B相似度

B可能经过旋转

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix}$$

仿射矩阵T

思路：

A若与B是同一个指纹，一定经过某个仿射矩阵变换

因为仿射矩阵T由abcdef 6个未知数组成，所以需要3个点即3对 x,y 来确定T

随机找3个点，算出T，再用A中其他点用这个T去变换，得到1个结果，看B中有没有这个结果

对这个T进行投票