EBU7240

Computer Vision

- Fitting: Least squares, RANSAC, Hough Transform -

Semester 1, 2021

Changjae Oh

Raguram ‘11

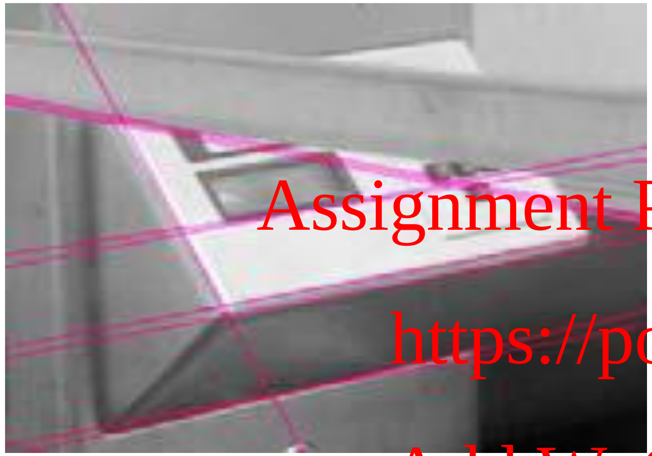
Fitting

- We’ve learned how to detect edges, cor ners, blobs. **Now what?**

- We would like to form a higher-level, m ore compact representation of the feat ures in the image by grouping multiple features according to a simple model

Fitting

- **Choose a parametric model to represent a set of features**



simple model: lines

simple model: circles

complicated model: car

Credit: K. Grauman

Fitting: Challenges

Case study: Line detection

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

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

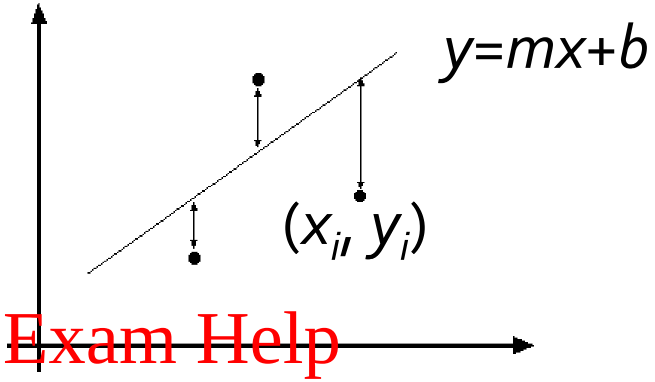
- **What if we're not even sure it's a line?**
  - Model selection (not covered)

Methods

- **Least squares**
- RANSAC
- Hough Transform

Least squares line fitting

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



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

$$E = \|Y - XB\|^2 \quad \text{where} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

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

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

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

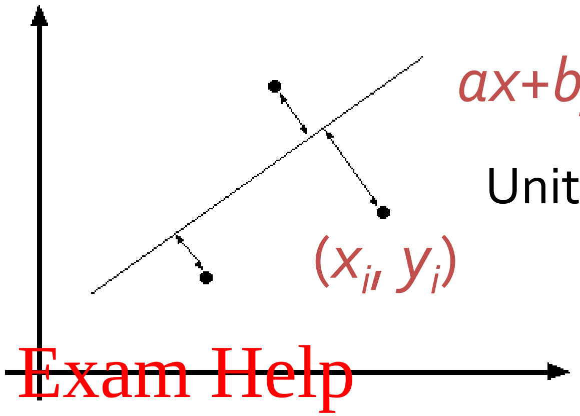
Normal equations: least squares solution to $XB=Y$

Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines

Total least squares

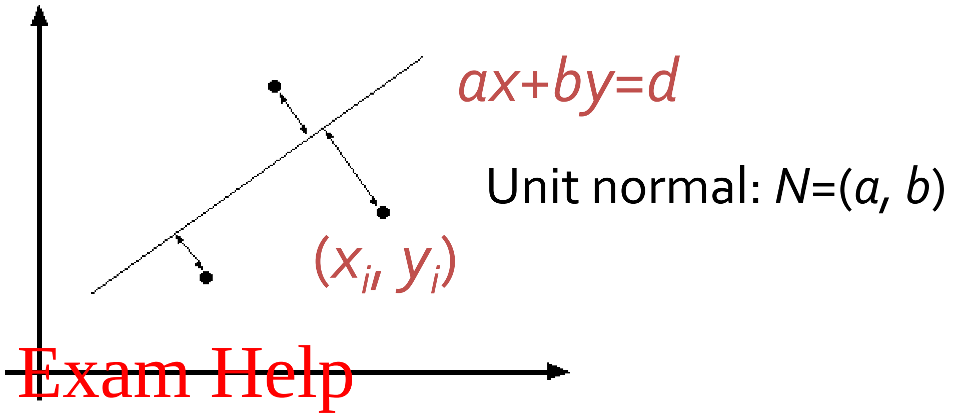
- Distance between point (x_i, y_i) and line $ax+by=d$ $(a^2+b^2=1)$: $|ax_i + by_i - d|$



Total least squares

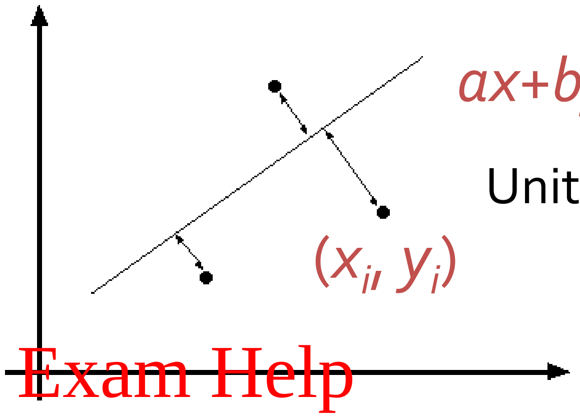
- Distance between point (x_i, y_i) and line $ax+by=d$ $(a^2+b^2=1)$: $|ax_i + by_i - d|$
- Find (a, b, d) to minimize the sum of squared *perpendicular* distances

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



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



Unit normal: $N=(a, b)$

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

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

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

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

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

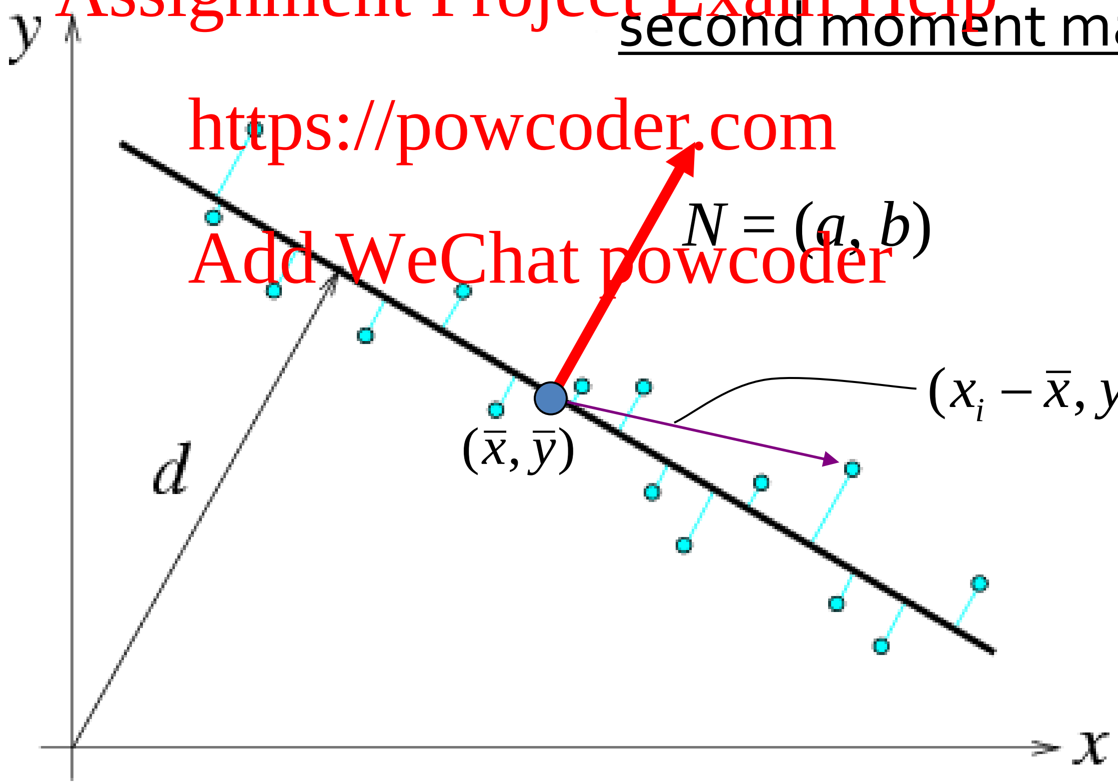
Total least squares

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

Total least squares

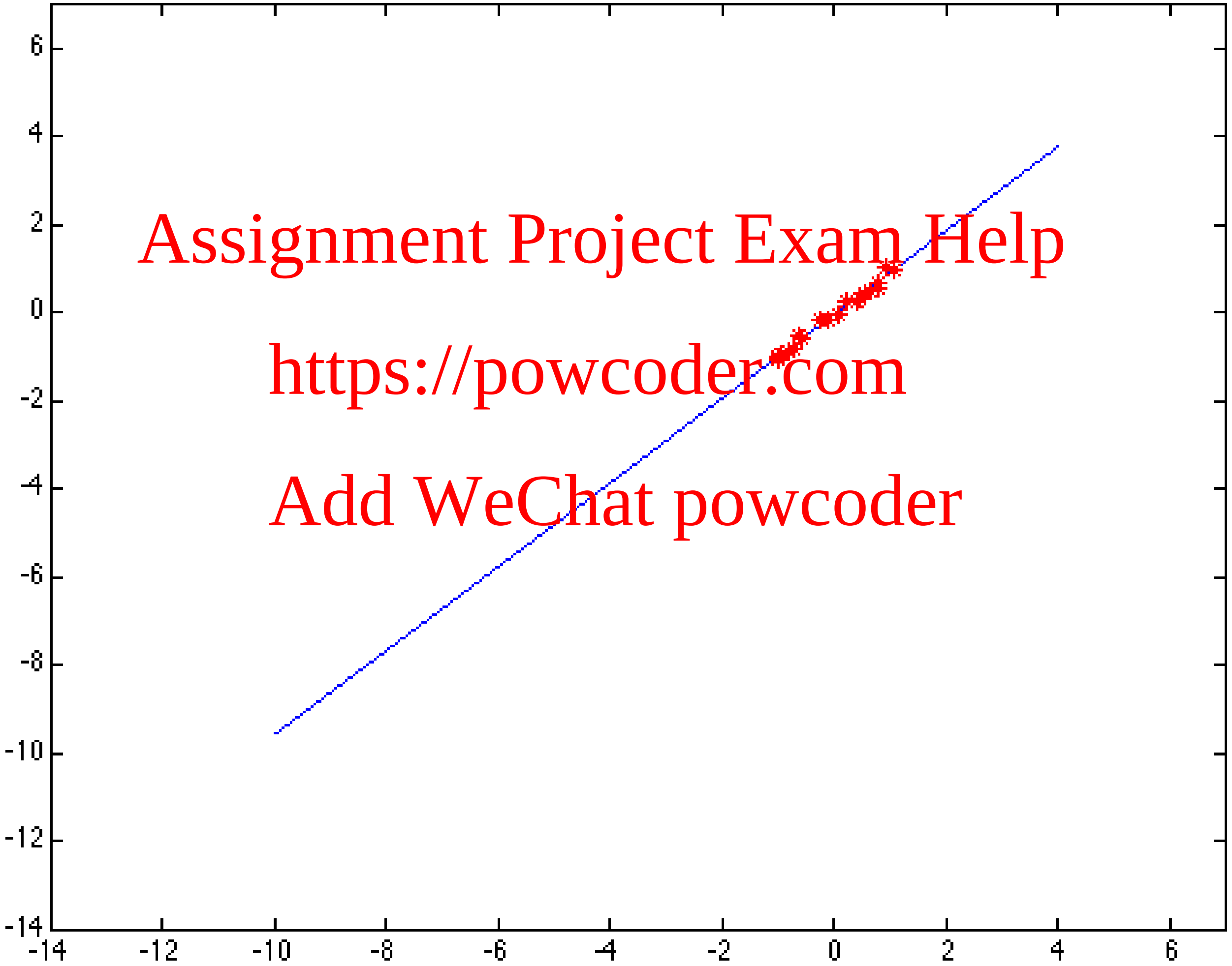
$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \qquad U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i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

second moment matrix



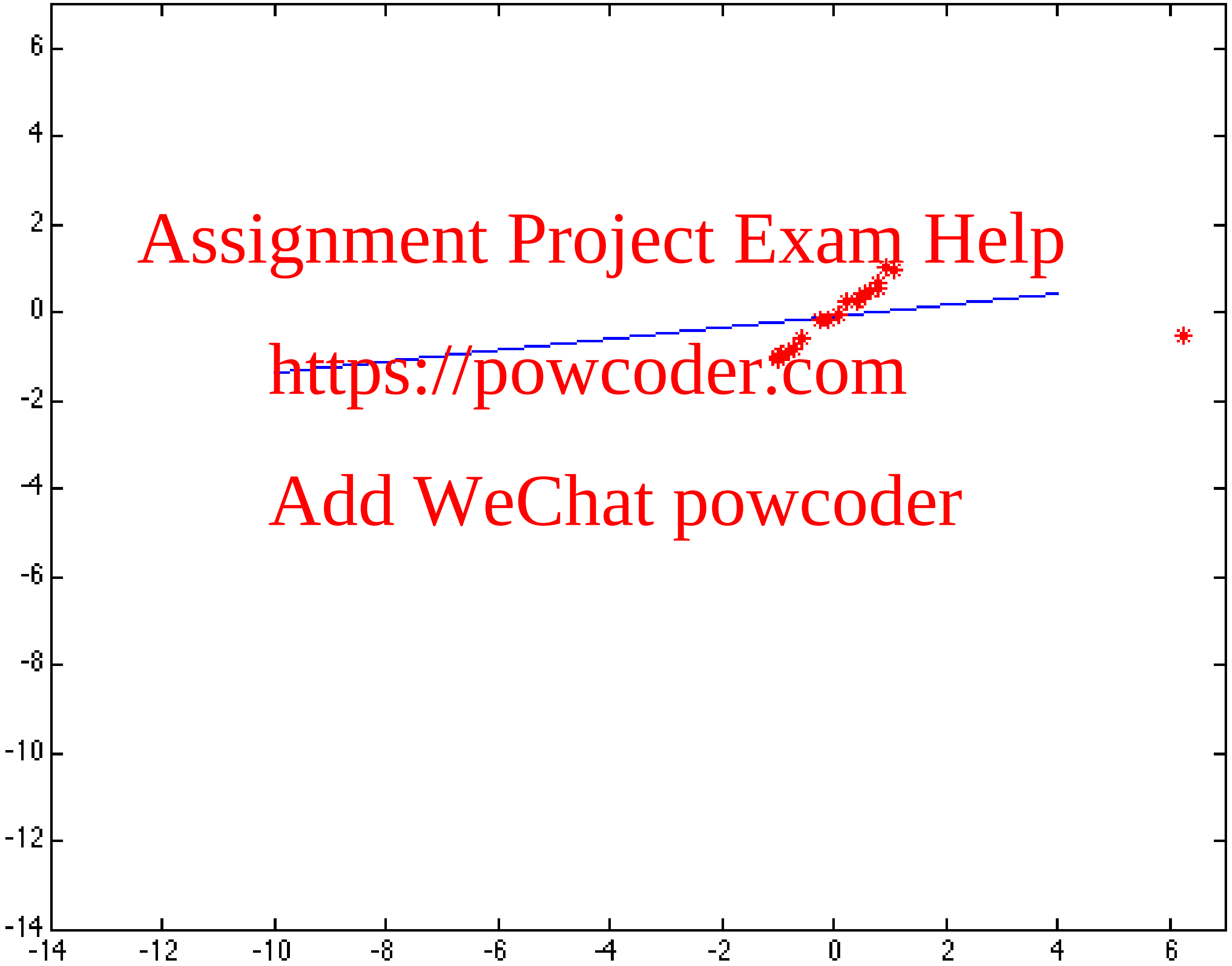
Least squares: Robustness to noise

- **Least squares fit to the red points:**



Least squares: Robustness to noise

- **Least squares fit with an outlier:**



Problem: squared error heavily penalizes outliers

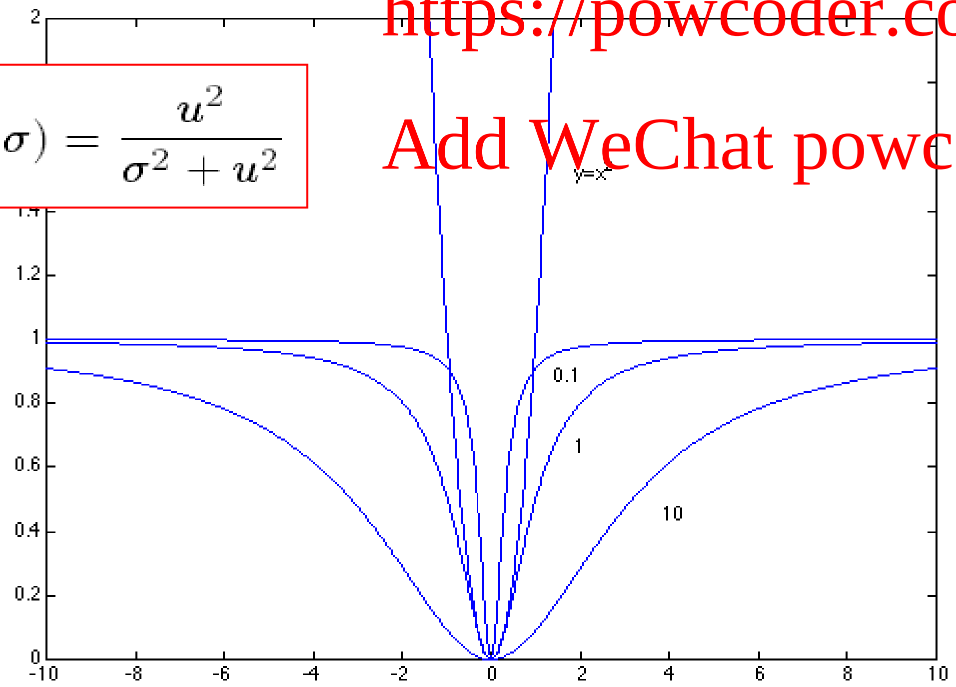
Robust estimators

- **General approach: find model parameters θ that minimize**

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

- $r_i(x_i, \theta)$: residual of *i*-th point w.r.t. model parameters θ
- $\rho(.)$: robust function with scale parameter σ

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



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

Slide from Lazebnik

Robust estimators

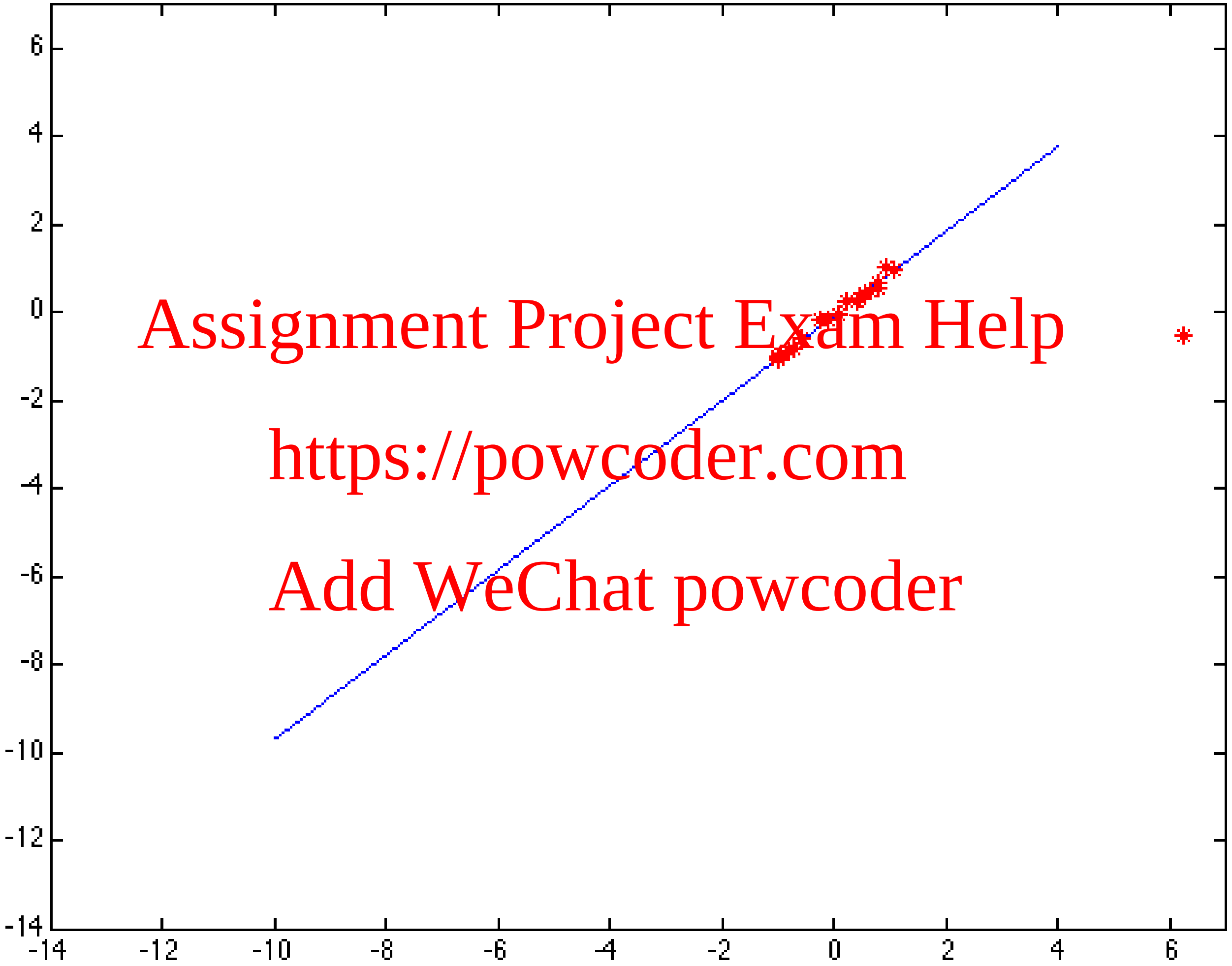
- **General approach: find model parameters θ that minimize**

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

- $r_i(x_i, \theta)$: residual of i-th point w.r.t. model parameters θ
- $\rho(.)$: robust function with scale parameter σ

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Least squares solution can be used for initialization
- Scale of robust function should be chosen carefully

Choosing the scale: Just right



The effect of the outlier is minimized

Methods

- Least squares
- RANSAC
- Hough Transform

RANSAC

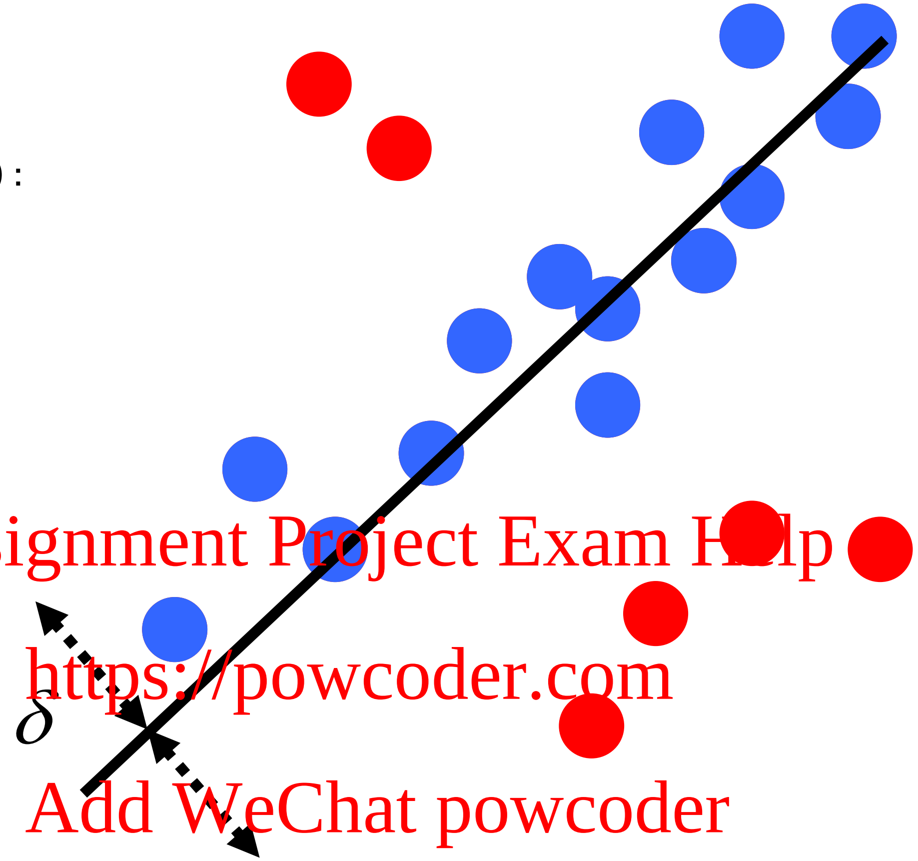
- **Robust fitting can deal with a few outliers – what if we have very many?**
- **Random sample consensus (RANSAC):**
  - Very general framework for model fitting in the presence of outliers
- **Outline**
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find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

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

(RANdom SAmple Consensus) :
Learning technique to estimate parameters of a model by random sampling of observed data

Fischler & Bolles in '81.



$$\pi : \boldsymbol{I} \rightarrow \{\boldsymbol{P}, \boldsymbol{O}\} \qquad \min_{\pi} |\boldsymbol{O}|$$

Model parameters

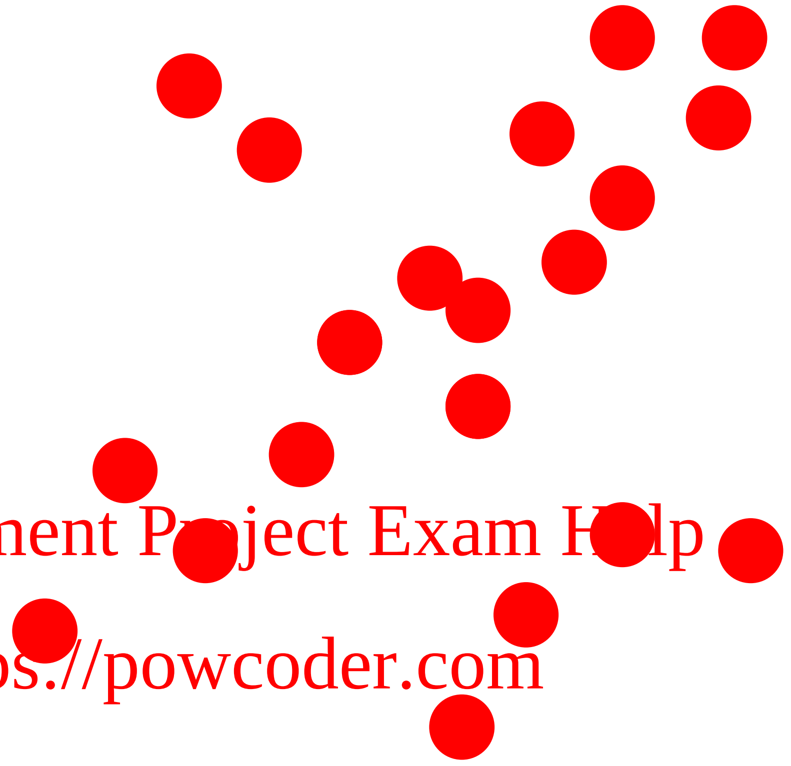
such that:

$$\boldsymbol{f}(\boldsymbol{P}, \beta) < \delta \qquad \boldsymbol{f}(\boldsymbol{P}, \beta) = \left\| \beta - \left(\boldsymbol{P}^{T} \boldsymbol{P}\right)^{-1} \boldsymbol{P}^{T} \right\|$$

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



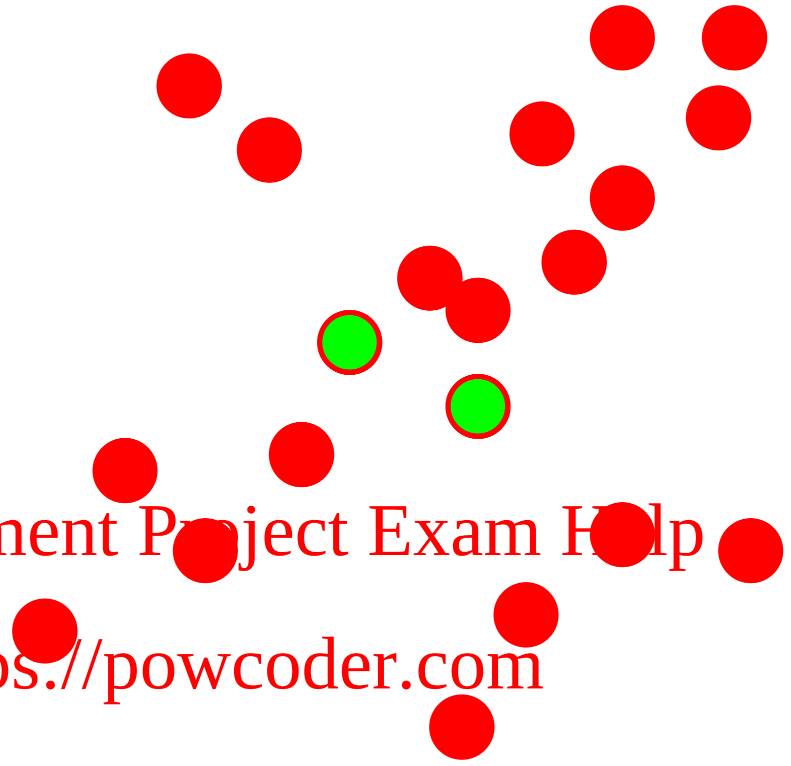
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



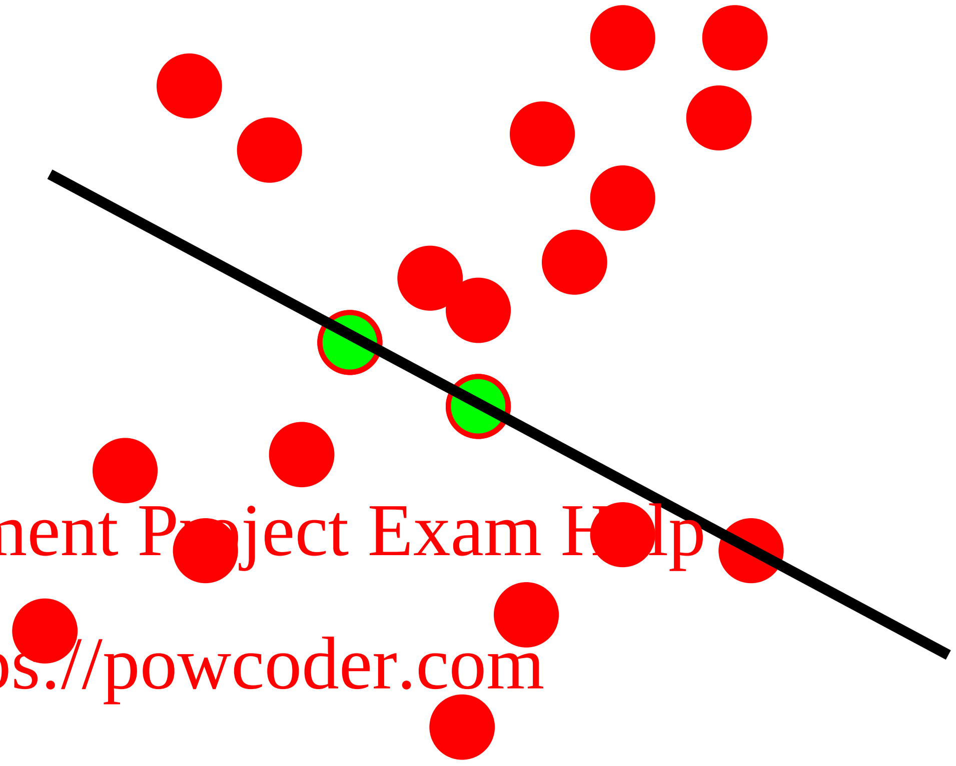
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



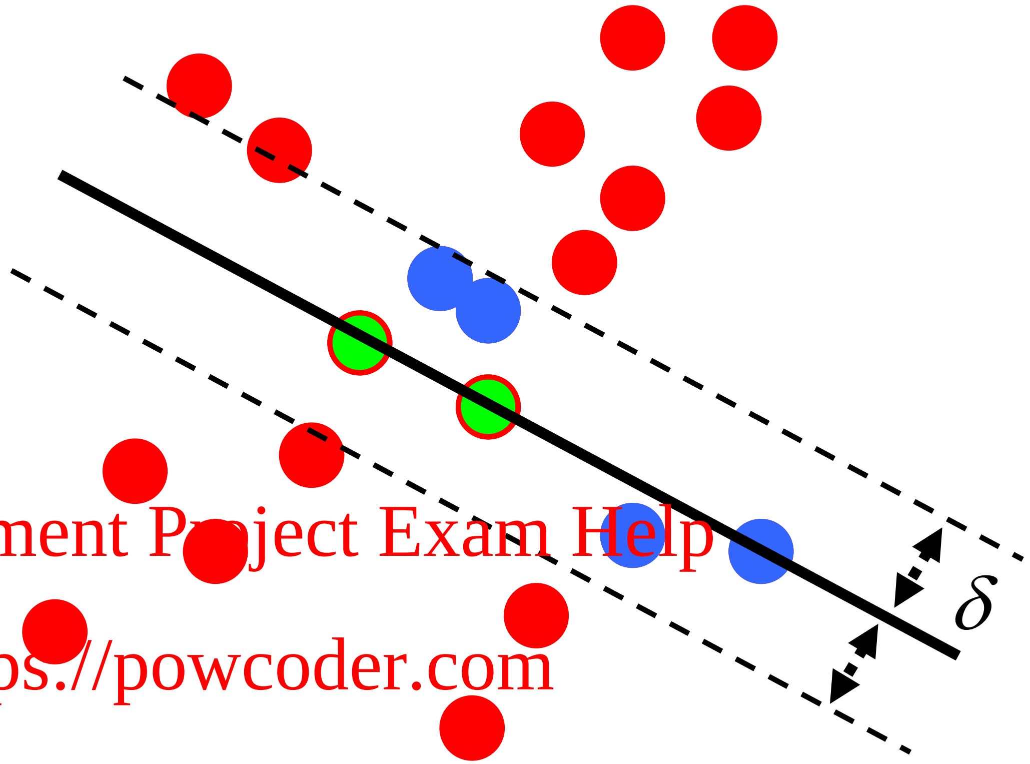
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



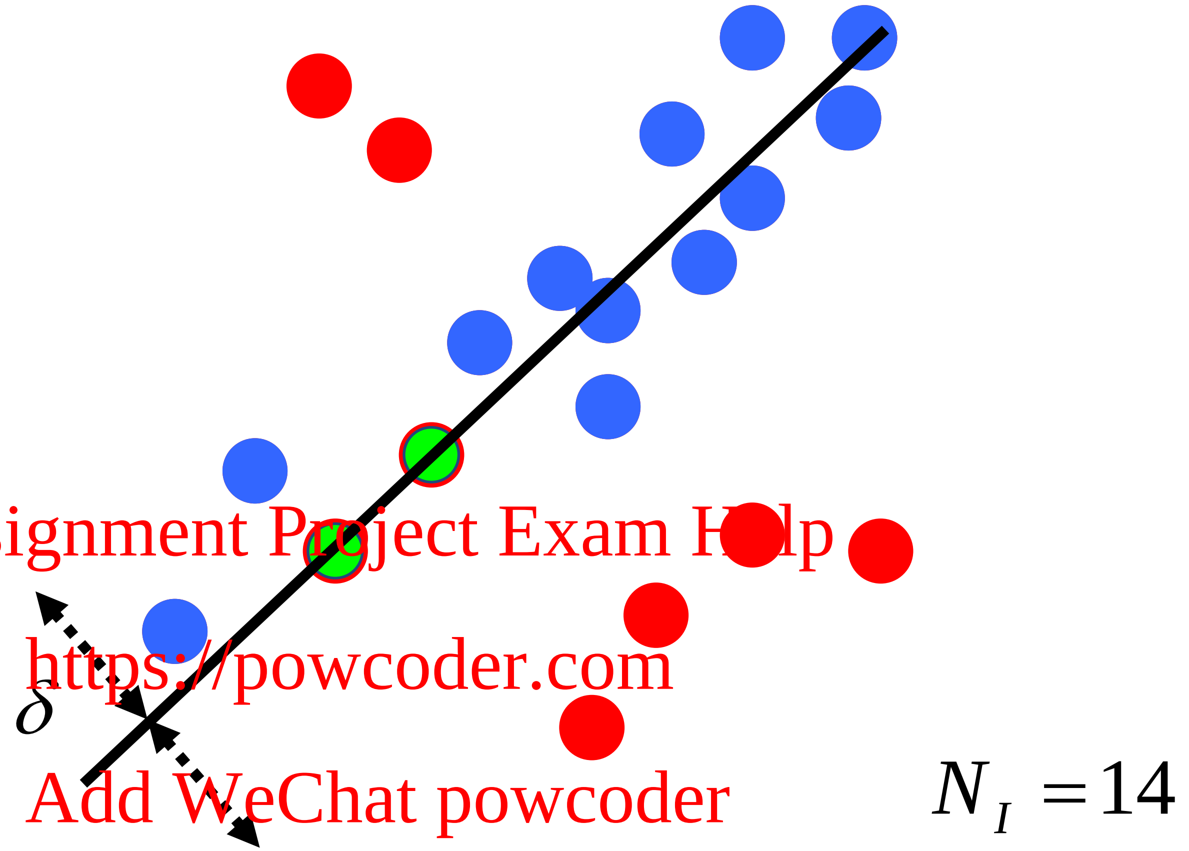
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Parameters in RANSAC

- Number of trials S
- Number of sampled points k
- Fitting the line to these k points
- Finding inliers to this line among the remaining points (i.e., points whose distance from the line is less than δ)

How to choose parameters?

- **Number of trials S**
  - Choose S so that, with probability P, at least one random sample is free from outliers (e.g. P=0.99) (outlier ratio: e)
- **Number of sampled points k**
  - Minimum number needed to fit the model
- **Distance threshold δ**
  - Choose δ so that a good point with noise is likely (e.g. prob=0.95) within threshold

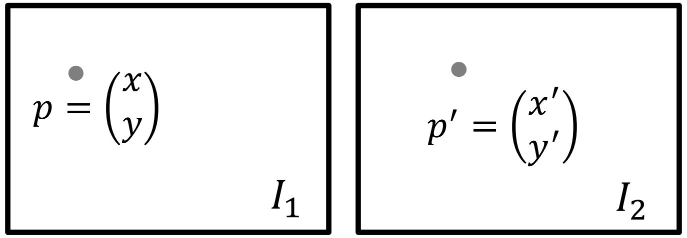
$$1 - P = (1 - (1 - e)^k)^S$$

$$\boxed{S = \frac{\log(1-P)}{\log(1-(1-e)^k)}}$$

	proportion of outliers e						
k	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

Image Stitching using Affine Transform - example

- **Affine transformation estimation with RANSAC**



1. Randomly sample k data ($k \geq 3$).
   Here, compare when $k = 3, 4$.

2. Estimate the affine transformation T by solving $\boldsymbol{Mx} = \boldsymbol{b}$.

3. Score by computing the number of inliers satisfying $|Tp - p'|^2 < \delta^2$ from all matches.

   Repeat 1~3 steps S times ($T_1, \ldots, T_S$).

4. Select the best affine transformation T_B from ($T_1, \ldots, T_S$).

5. Re-estimate the affine transformation by solving $\boldsymbol{Mx} = \boldsymbol{b}$ with T_B's inliers.

Image Stitching using Affine Transform - example

Input: A set of N matched points $MP = \{(p_0, p_0'), (p_1, p_1') \dots, (p_{N-1}, p_{N-1}')\}$
Output: Affine transform T_F

S: the number of trials
count_mat: $S \times 1$ vector
IN: a set of inliers

Initialize count_mat to 0
for i = 0 ~ S-1
    Randomly select k matched points from MP (Usually, k=3,4)
    Estimate an affine transformation T_i with k matched points
    for j = 0 ~ N-1
        if $|T_i p_j - p_j'|^2 < \delta^2$
            count_mat[i]++

Choose the best affine transformation $T \leftarrow T_K$ where $K = arg \max_i$ count_mat[i]

$IN = NULL$
for j = 0 ~ N-1
    if $|Tp_j - p_j'|^2 < \delta^2$
        $IN \leftarrow IN \cup \{(p_j, p_j')\}$

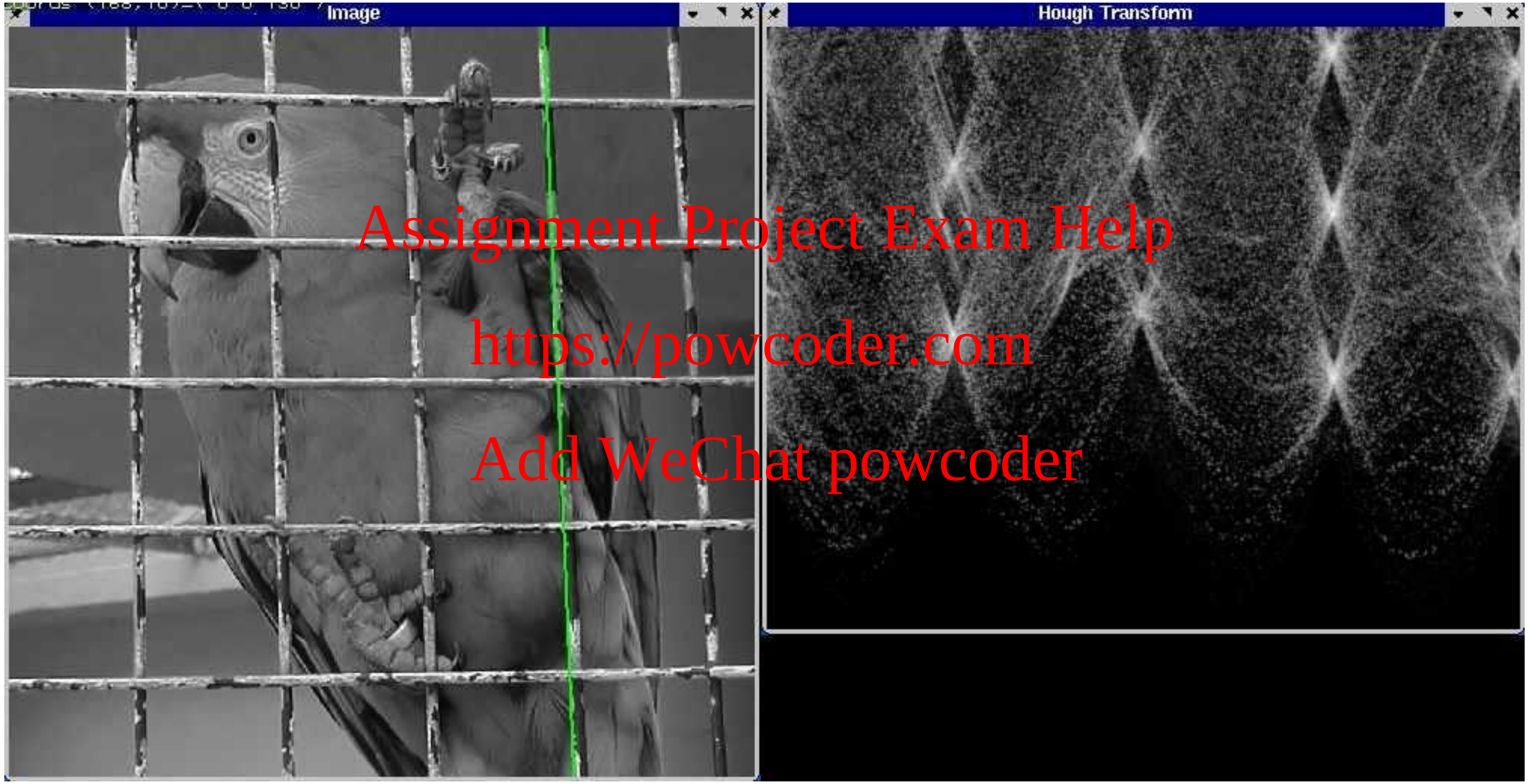
Re-estimate an affine transformation T_F with IN

RANSAC- Applications

RANSAC: Conclusions

- **Good**
  - Robust to outliers
  - Applicable for larger number of objective function parameters than Hough transform
  - Optimization parameters are easier to choose than Hough transform
- **Bad**
  - Computational time grows quickly with fraction of outliers and number of parameters
  - Not good for getting multiple fits
  - Lots of parameters to tune
  - Doesn't work well for low inlier ratios (too many iterations, or can fail completely)
- **Common applications**
  - Computing a homography (e.g., image stitching)
  - Estimating fundamental matrix (relating two views)

Hough transform



Fitting: Review

- **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**

Methods

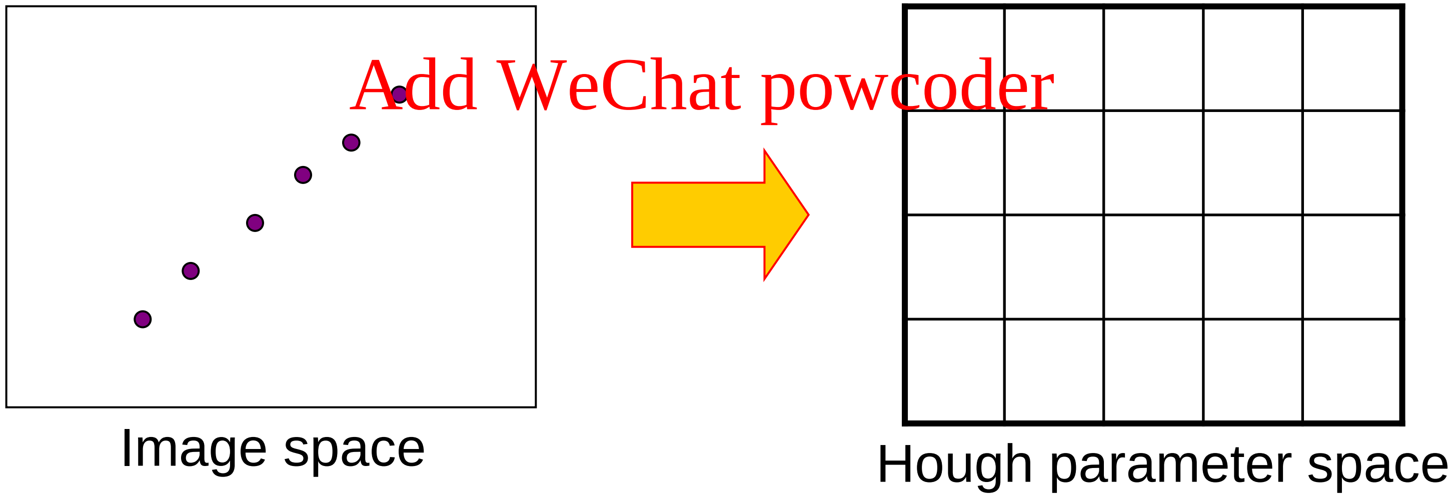
- Least squares
- RANSAC
- Hough Transform

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

Hough Transform

- **An early type of voting scheme**
  - 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
    - Find maximum or local maxima in grid



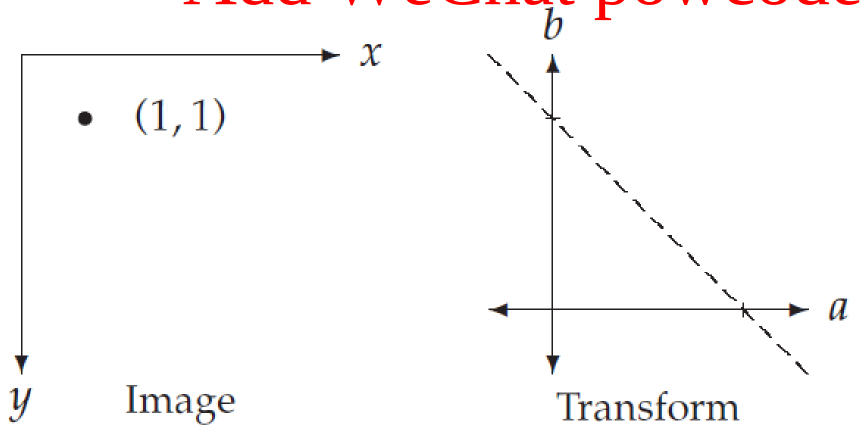
P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Hough Transform: Fitting Multiple Lines

- **One simple example using Hough Transform**
  - Fitting lines, when a set of sparse points are given.

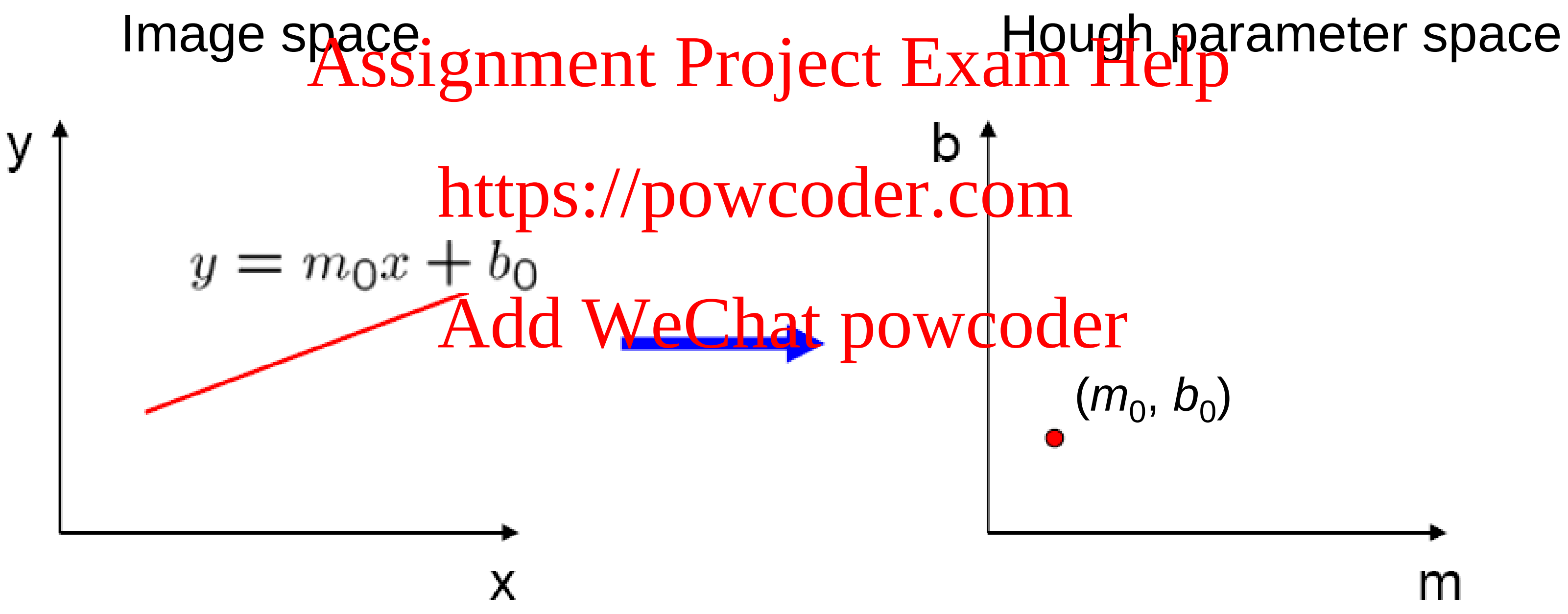
- **Key idea**: transform $(x, y) \rightarrow (a, b)$

$y = ax + b$ $\Rightarrow$ $1 = a + b$



Parameter space representation

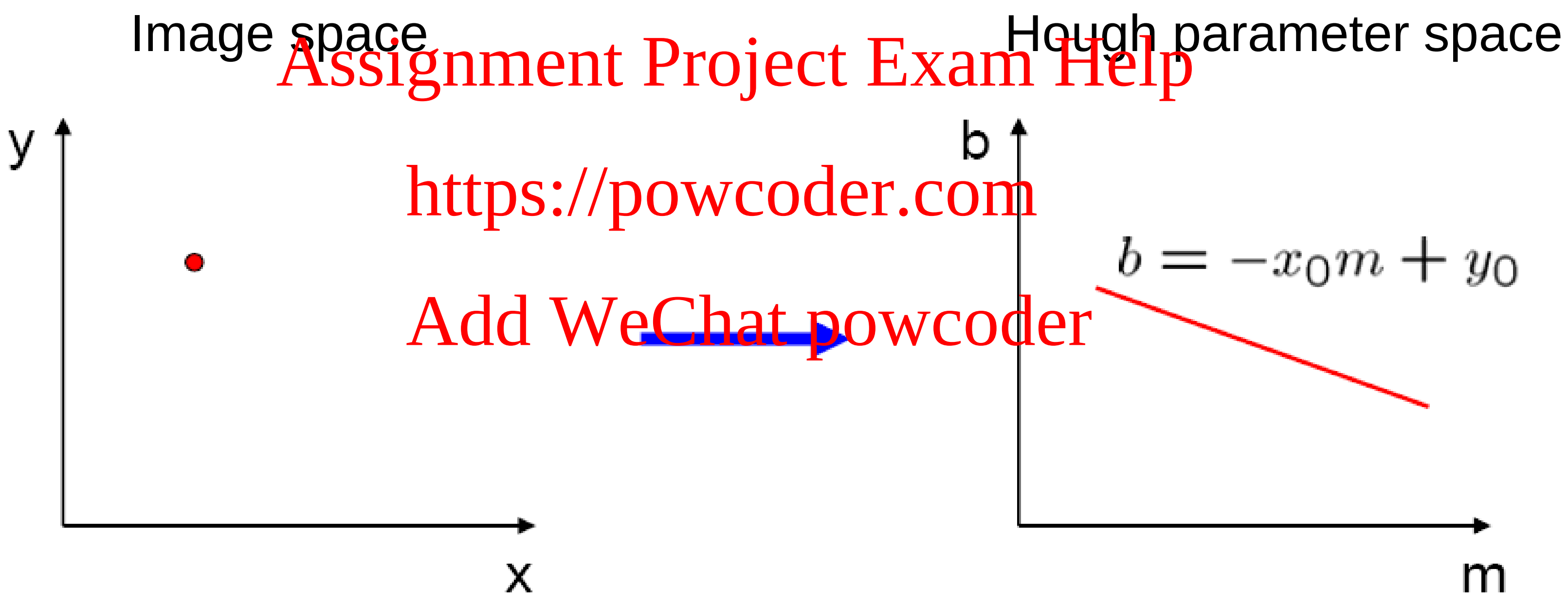
- **A line in the image corresponds to a point in Hough space**



Source: S. Seitz

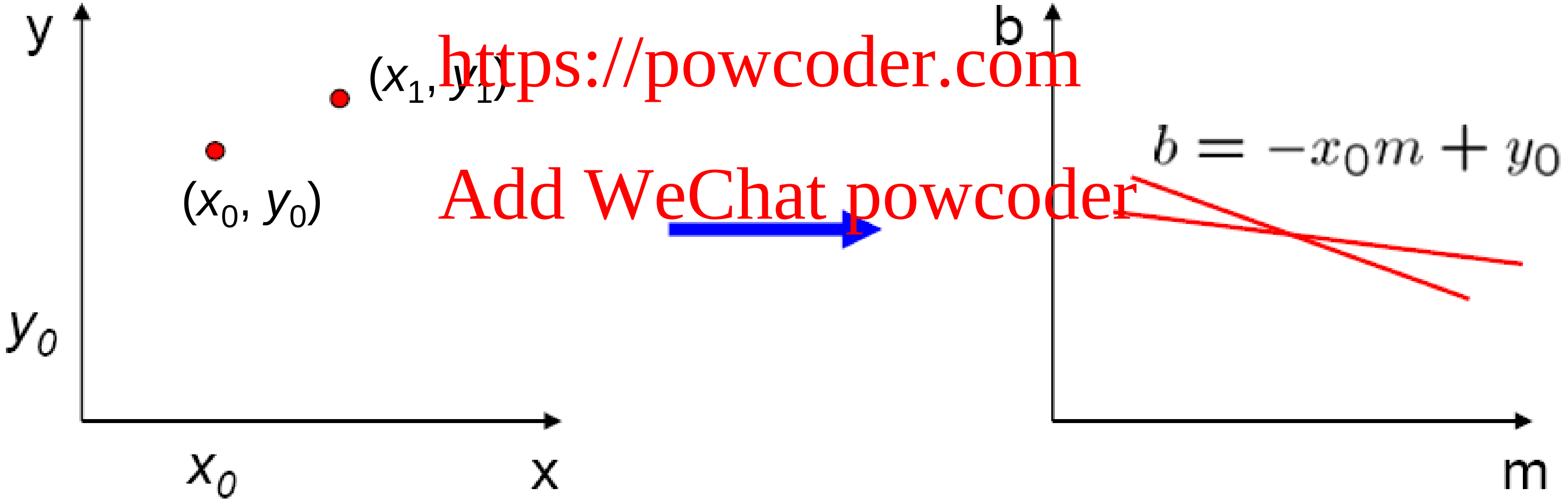
Parameter space representation

- **What does a point (x_0, y_0) in the image space map to the Hough space?**
  - Answer: the solutions of $y_0 = mx_0 + b$
  - This is a line in Hough space



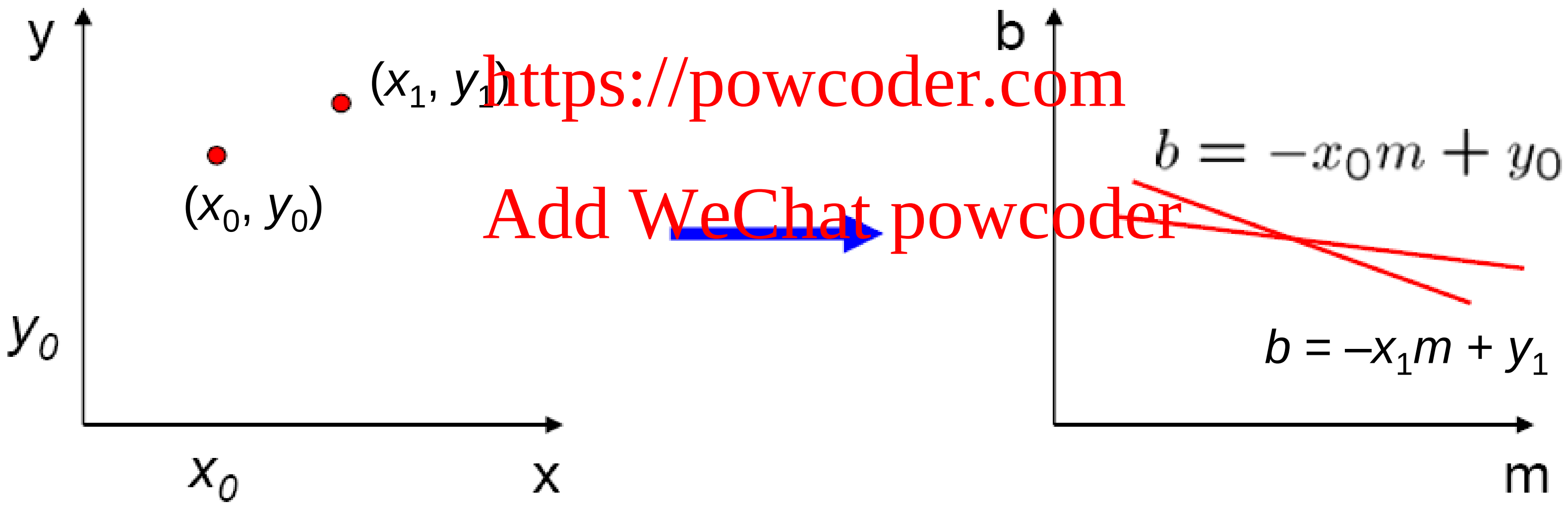
Parameter space representation

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



Parameter space representation

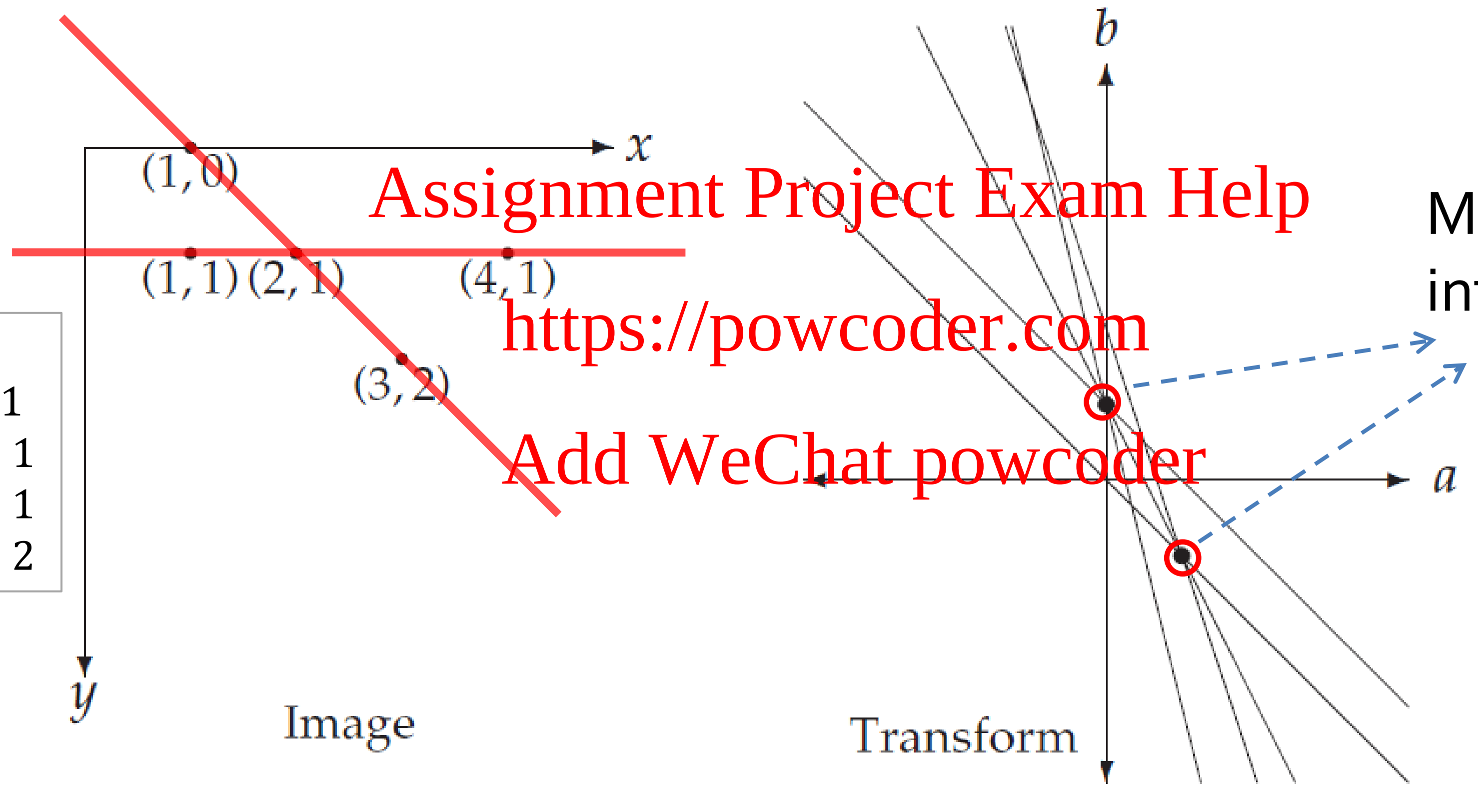
- **Where is the line that contains both (x_0, y_0) and (x_1, y_1)?**
  - It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$



Hough Transform: Fitting Multiple Lines

- Suppose we have five points

$(1,0) \rightarrow b = -a$
$(1,1) \rightarrow b = -a + 1$
$(2,1) \rightarrow b = -2a + 1$
$(4,1) \rightarrow b = -4a + 1$
$(3,2) \rightarrow b = -3a + 2$



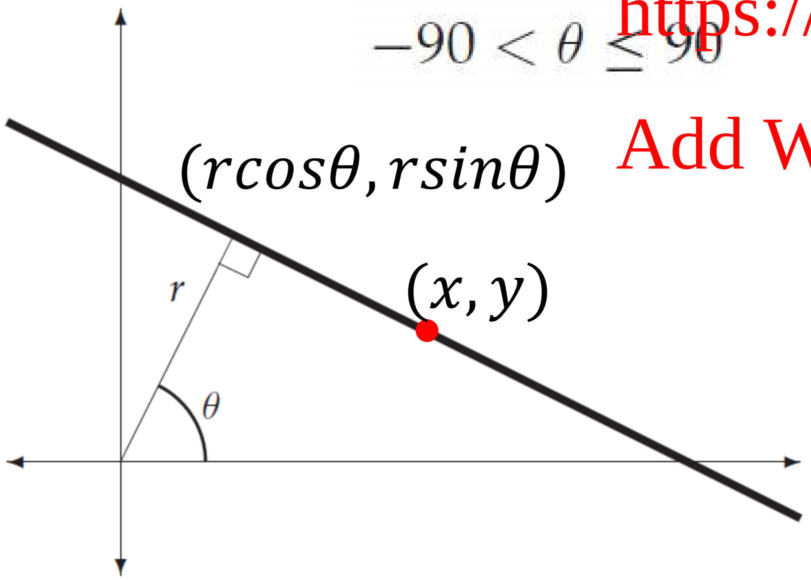
Maximum intersections!

$(a, b) = (0,1)$
$(a, b) = (1, -1)$

Hough Transform: Fitting Multiple Lines

- **Problem with *(a,b)* space**
  - Unbounded parameter domains
    - $y = ax + b$ is not able to model a vertical line
  - Let's use different type of parameterization (r,θ)!

$-90 < \theta \leq 90$

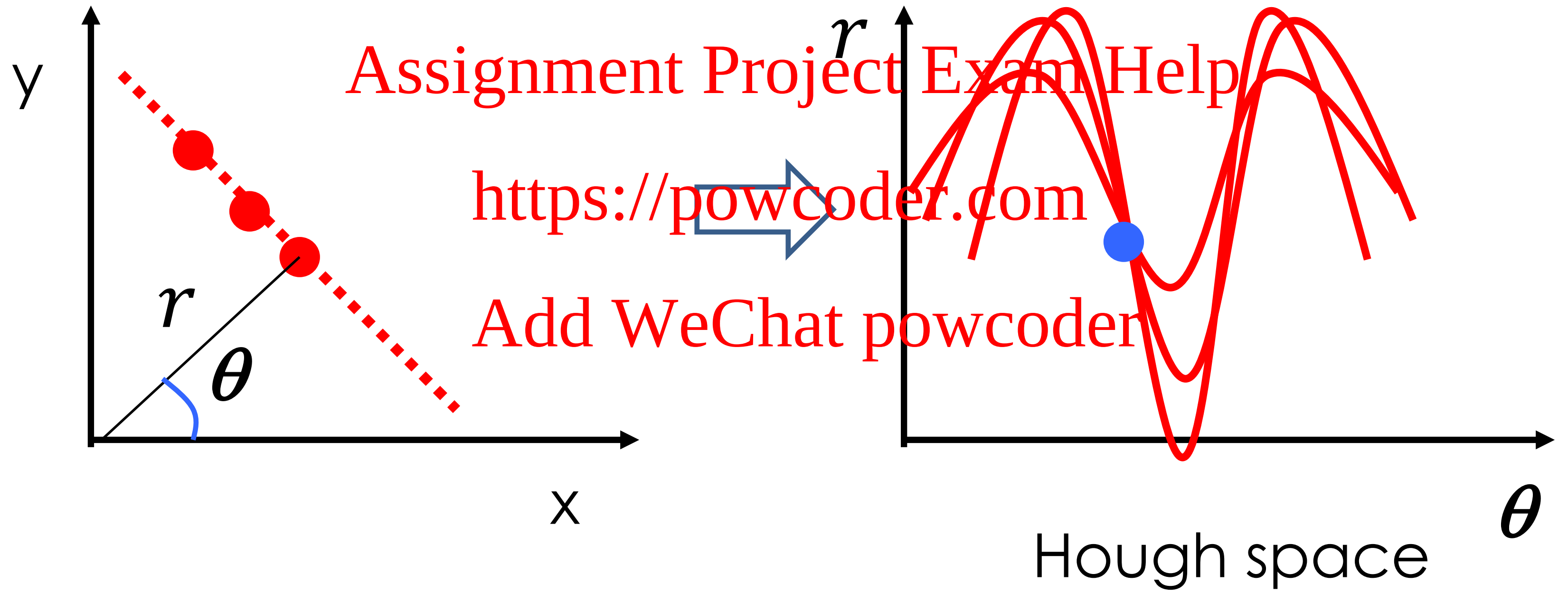


$$\frac{y - r\sin\theta}{x - r\cos\theta} = -\tan(90 - \theta) = -\frac{\cos\theta}{\sin\theta}$$

$$\Rightarrow \boxed{x\cos\theta + y\sin\theta = r}$$

Hough Transform: Fitting Multiple Lines

Use a polar representation for the parameter space



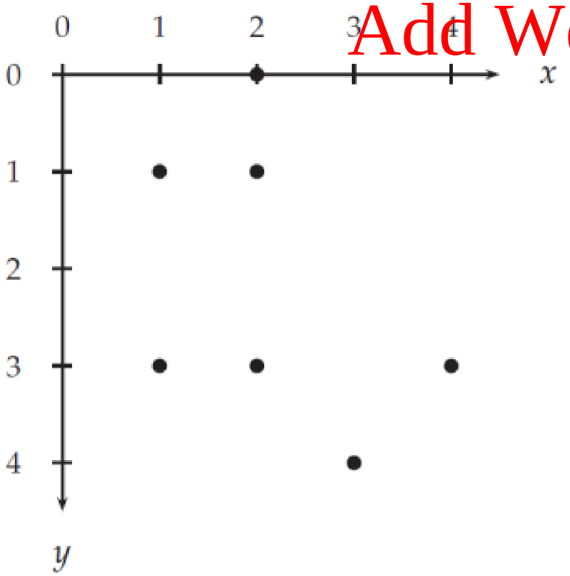
$$\boxed{x cos\theta + y sin\theta = r}$$

Hough Transform: Fitting Multiple Lines

- One problem still happens when handling **continuous** parameters (r, θ) on the discrete domain.
  - Too many possible (r, θ)!
  - So, let's just use a few number of **quantized** (r, θ)

Example

- If we discretize θ to use only four values: -45°, 0°, 45°, 90°

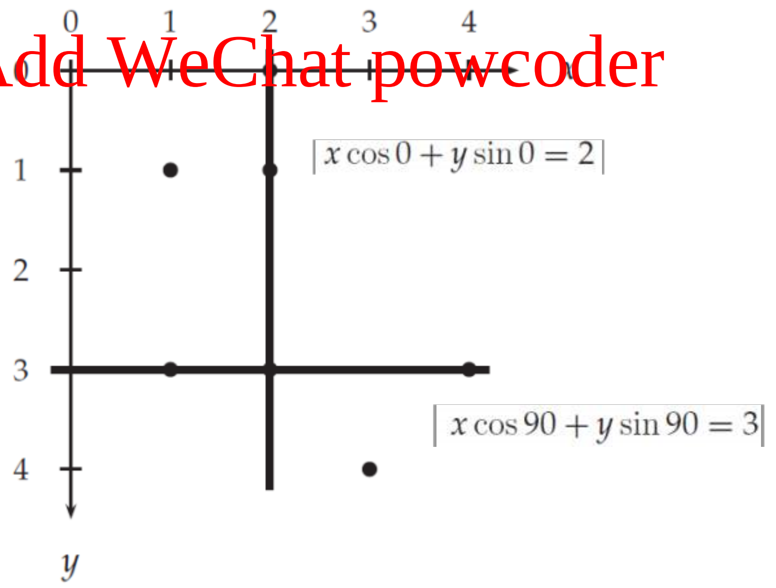


(x, y)	$-45°$	$0°$	$45°$	$90°$
$(2,0)$	1.4	2	1.4	0
$(1,1)$	0	1	1.4	1
$(2,1)$	0.7	2	2.1	1
$(1,3)$	-1.4	1	2.8	3
$(2,3)$	-0.7	2	3.5	3
$(4,3)$	0.7	4	4.9	3
$(3,4)$	-0.7	3	4.9	4

Hough Transform: Fitting Multiple Lines

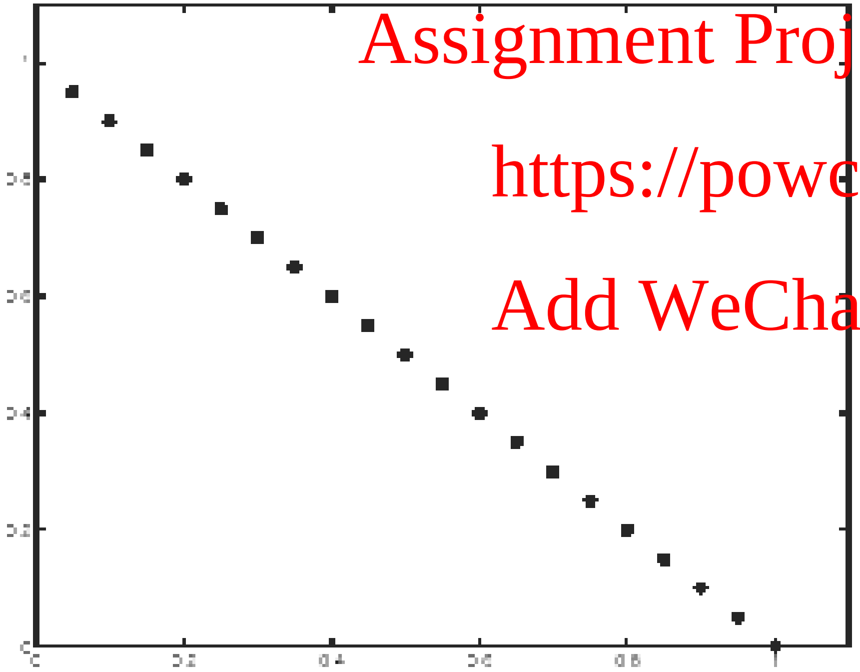
- The accumulator array contains **how many times** each value of (r, θ) appears in the table

	-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	

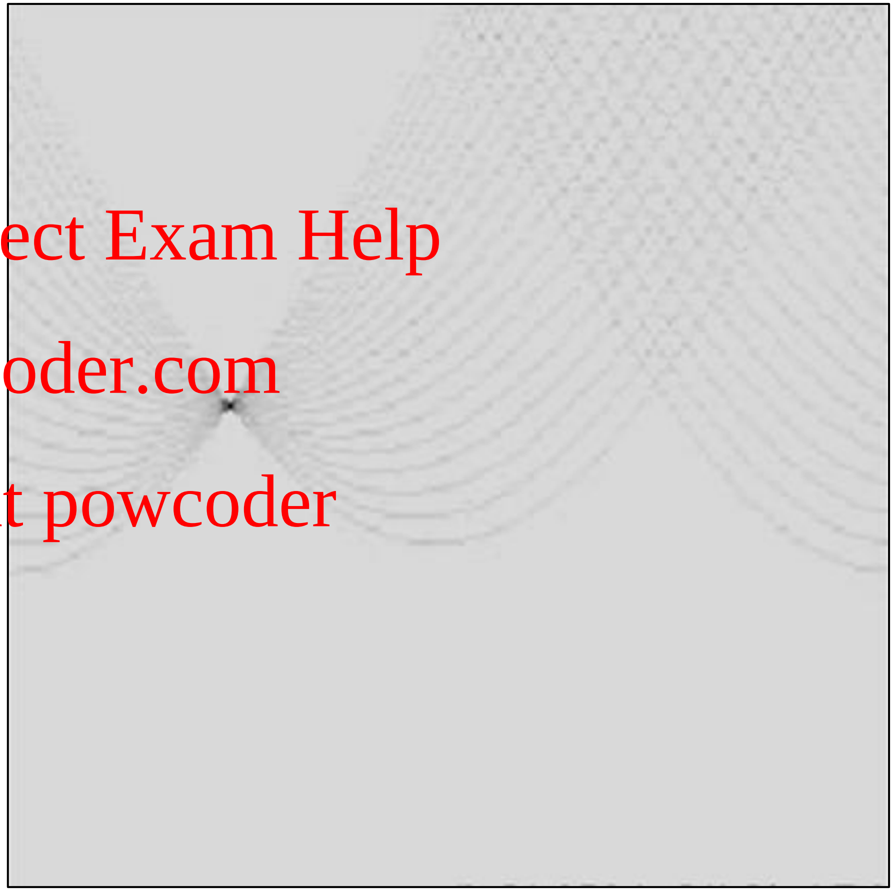


Hough Transform: Fitting Multiple Lines

Clean data



Features



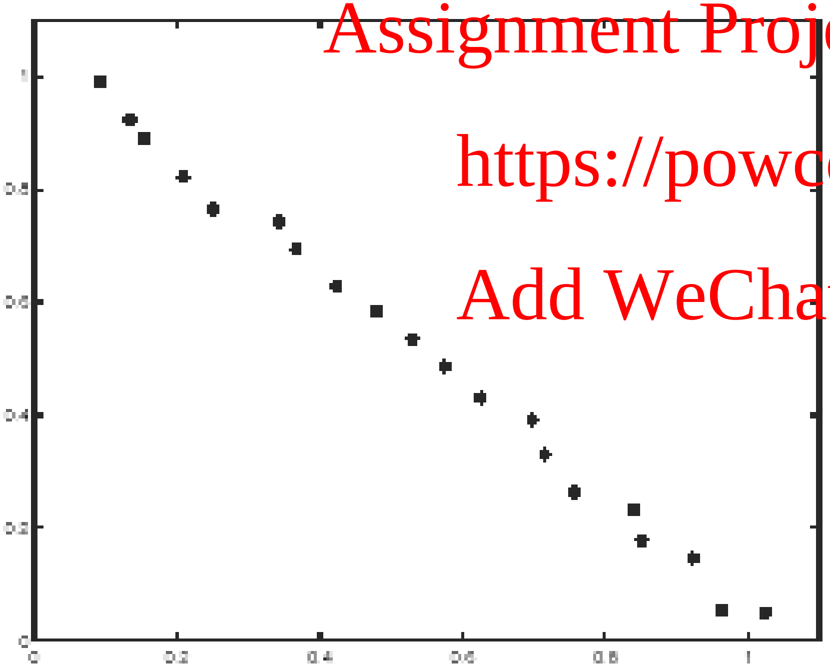
Votes

Slide from S. Savarese

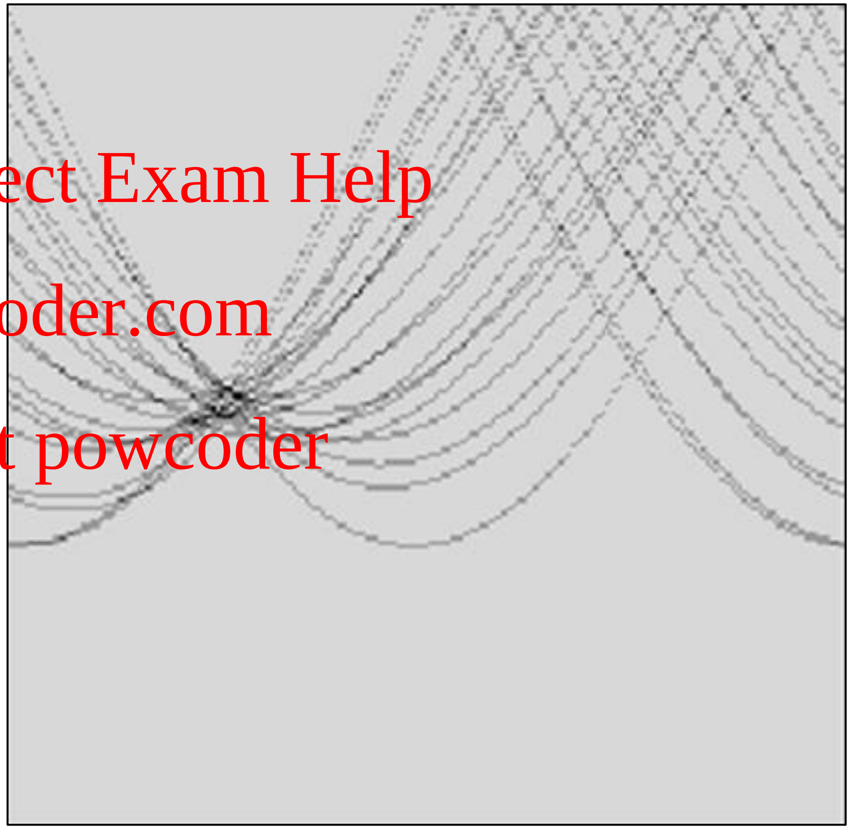
Hough Transform: Fitting Multiple Lines

- **Issue: noisy data**

Noisy data



Features

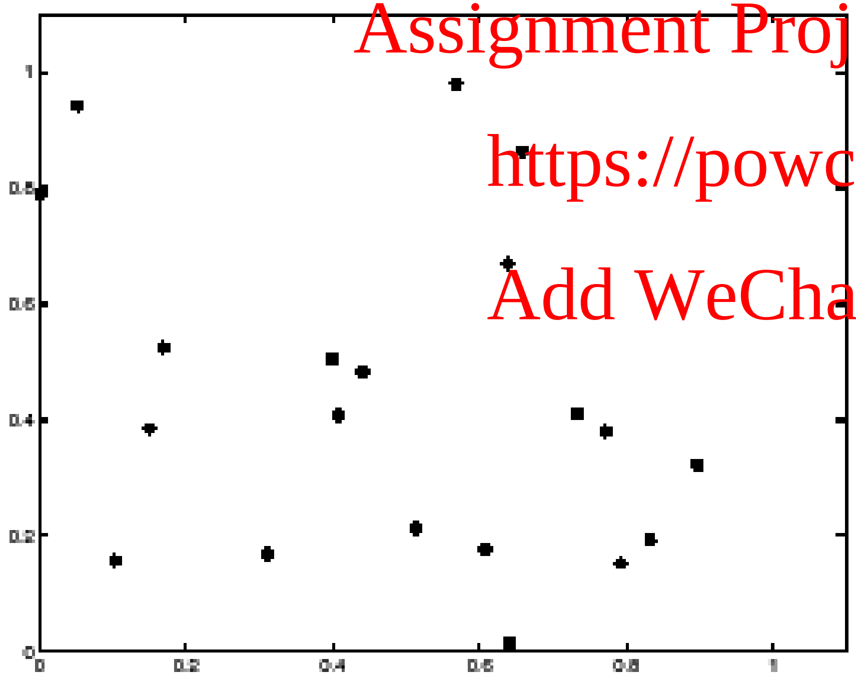


Votes

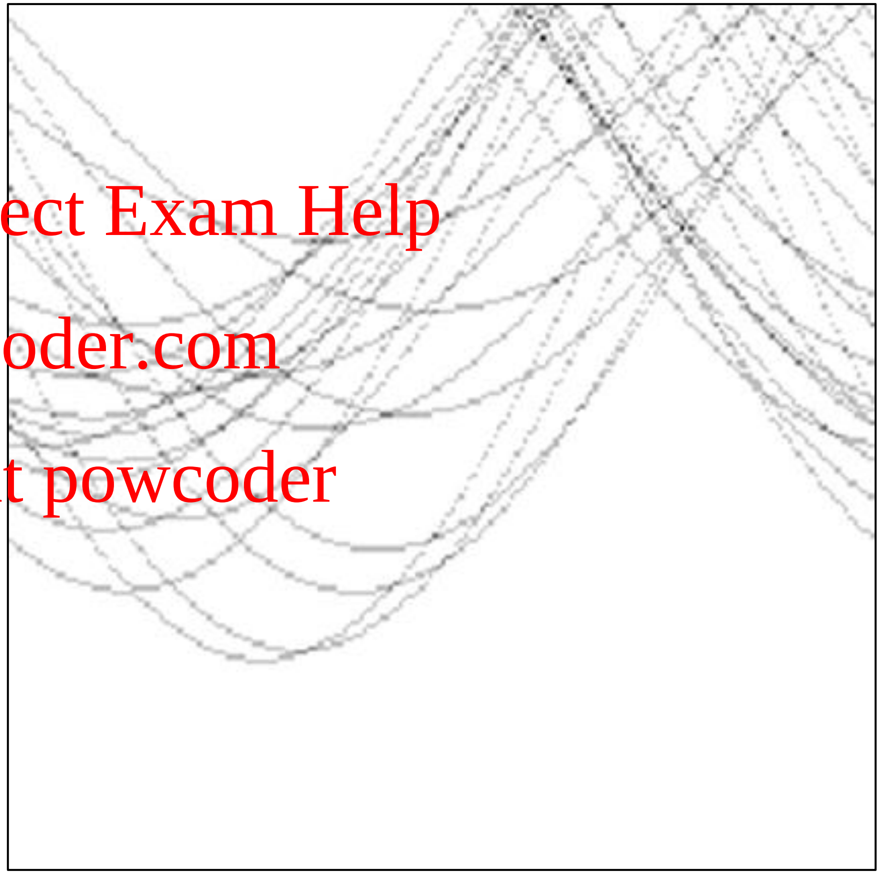
Slide from S. Savarese

Hough Transform: Fitting Multiple Lines

- **Issue: spurious peaks due to uniform noise**



Features



Votes

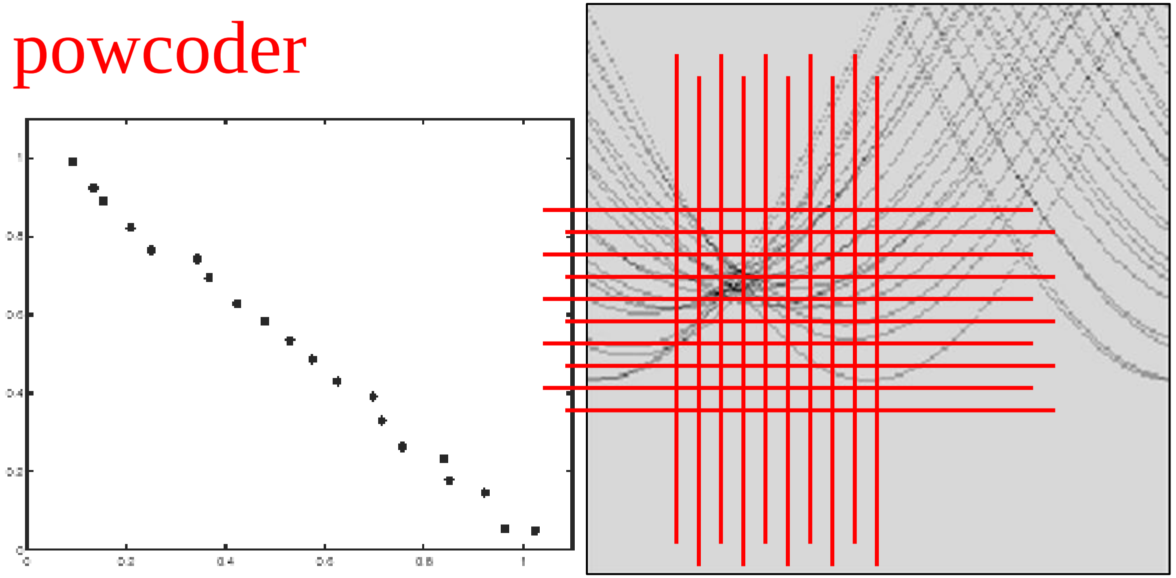
Dealing with noise

- **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

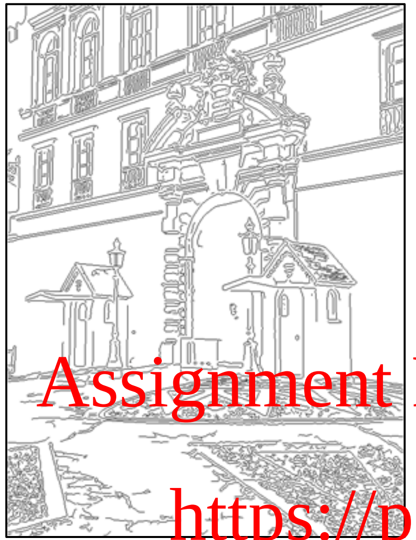
- **Increment neighboring bins (smoothing in accumulator array)**

- **Try to get rid of irrelevant features**

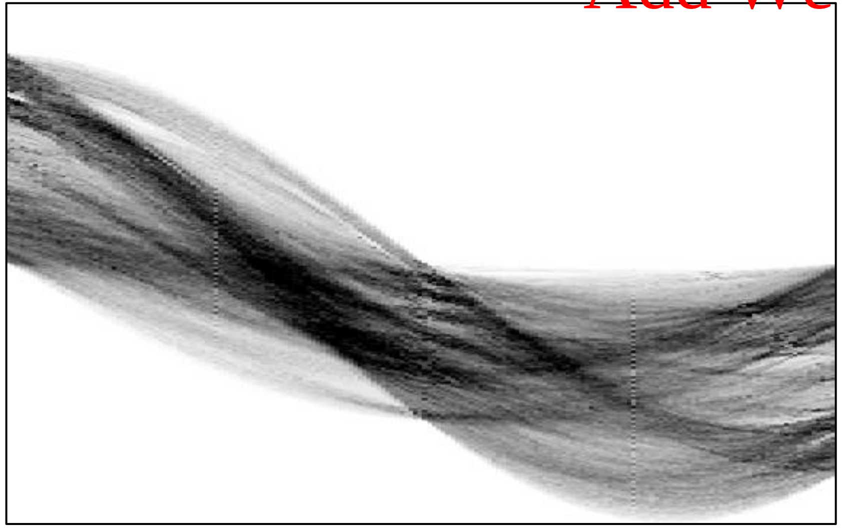
  - E.g., take only edge points with significant gradient magnitude



Hough Transform: Fitting Multiple Lines



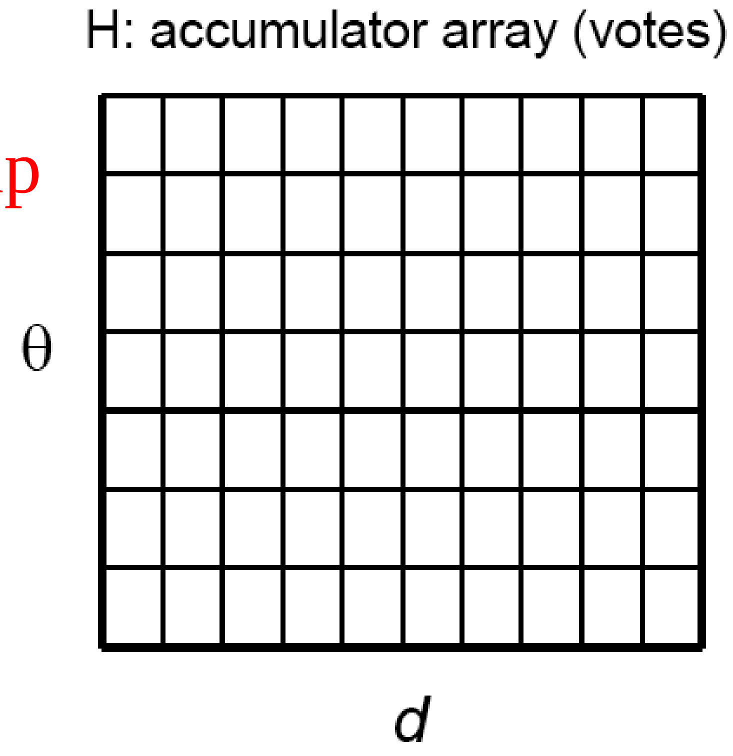
Canny Edge Detector



Find peaks and post-process

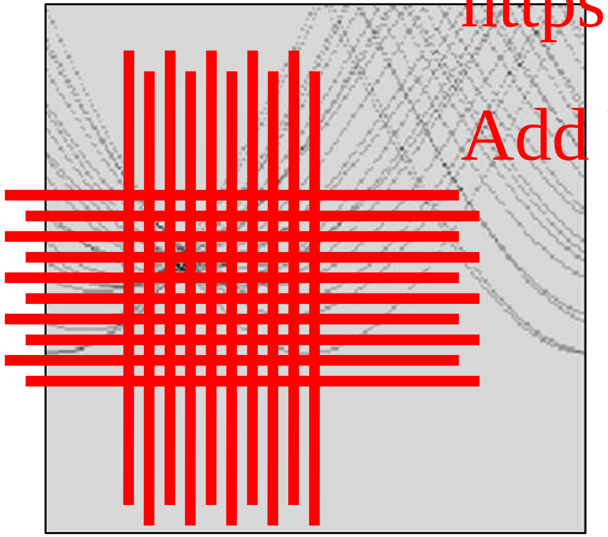
Algorithm outline

```
Initialize accumulator H to all zeros
for each feature point (x,y) in the image
    for θ = 0 to 180
        ρ = x cos θ + y sin θ
        H(θ, ρ) = H(θ, ρ) + 1
    end
end
Find the value(s) of (θ, ρ) where H(θ,ρ) is
a local maximum
The detected line in the image is given by
ρ = x cos θ + y sin θ
```

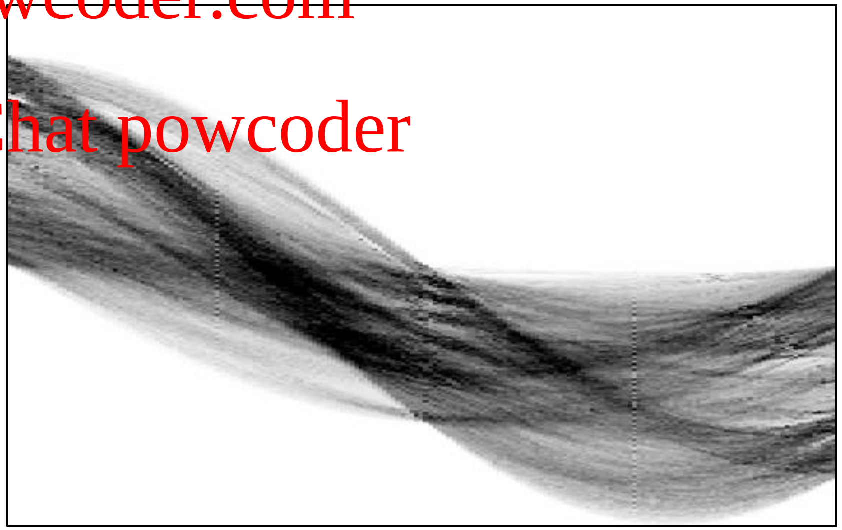



Hough Transform: Fitting Multiple Lines

- **Practical considerations**
  - Bin size
  - Smoothing
  - Finding multiple lines
  - Finding line segments



Adjusting bin size and the amount of smoothing



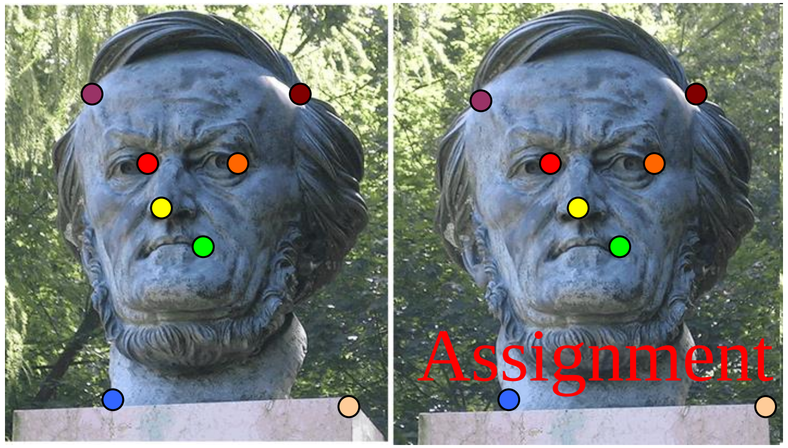
Finding multiple lines & line segments

Application

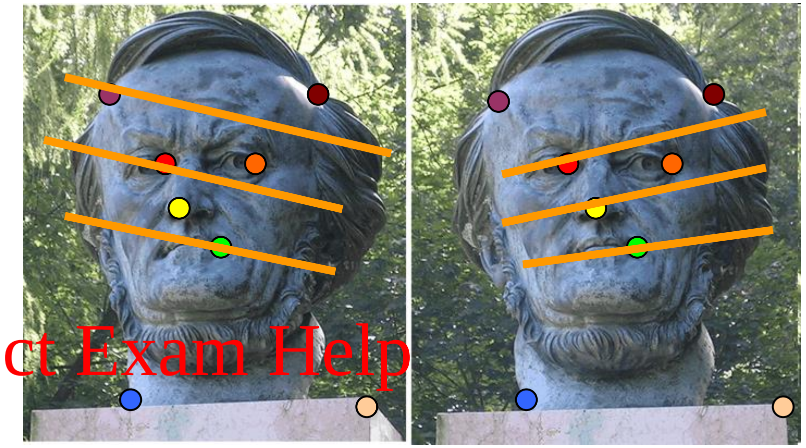
Hough Transform: Conclusions

- **Good**
  - Robust to outliers: each point votes separately
  - Fairly efficient (much faster than trying all sets of parameters)
  - Provides multiple good fits
- **Bad**
  - Some sensitivity to noise
  - Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot
  - Not suitable for more than a few parameters
  - grid size grows exponentially
- **Common applications**
  - Line fitting (also circles, ellipses, etc.)
  - Object instance recognition (parameters are affine transform)

Feature Matching and Fitting



Feature matching



Fundamental matrix estimation

Feature matching

Image stitching using geometric transform estimation

Feature Matching and Fitting

Feature Matching Modules

Parameter Estimation

Robust Estimation

Robust Estimation

1) Robust least square
2) RANSAC
3) M-estimator
4) Least Median of Square (LMedS)
And so on…

Parameter Examples

1) 2D Geometric transform
2) Fundamental matrix
3) 3D Camera motion
   (3D rotation/translation)
And so on

Parameter Estimation

1) Least squares fit
   (or Total least square)
2) Hough transform
3) Iterative Closest Points (ICP)
And so on

Note) ICP does NOT require feature matching modules

Next topic

- **Images are high-dimensional signals, consisting of redundant pixels. How can we group them?**
  - Prerequisite
    - Review EBU6230 Image/Video Processing – Week3: Edges

EBU7240
Computer Vision

- Grouping: thresholding, K-means -

Semester 1, 2021

Changjae Oh

Contents

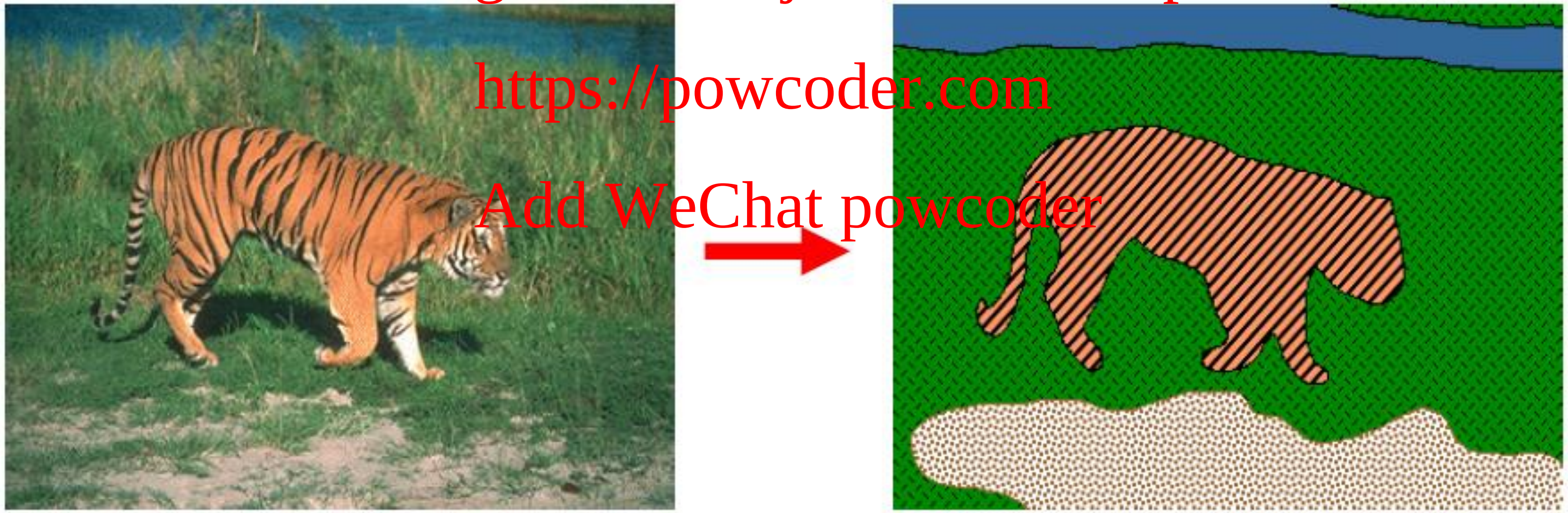
- **Unsupervised Segmentation**
  - Thresholding-based segmentation
  - K-mean clustering
- **Interactive segmentation**

Contents

- **Unsupervised Segmentation**
  - Thresholding-based segmentation
  - K-mean clustering
- **Interactive segmentation**

Introduction

- **Image segmentation**
  - Divide an image into **non-overlapping regions (object)** with similar information



Simple Thresholding

- **Single threshold**

$$\text{A pixel becomes} \begin{cases} \text{white if its gray level is} > T, \\ \text{black if its gray level is} \le T. \end{cases}$$


```
>> r=imread('rice.tif');
>> imshow(r),figure,imshow(r>110)
```


FIGURE 9.1 *Thresholded image of rice grains.*

Simple Thresholding

- **Single threshold**
  - Shows the hidden aspects of an image

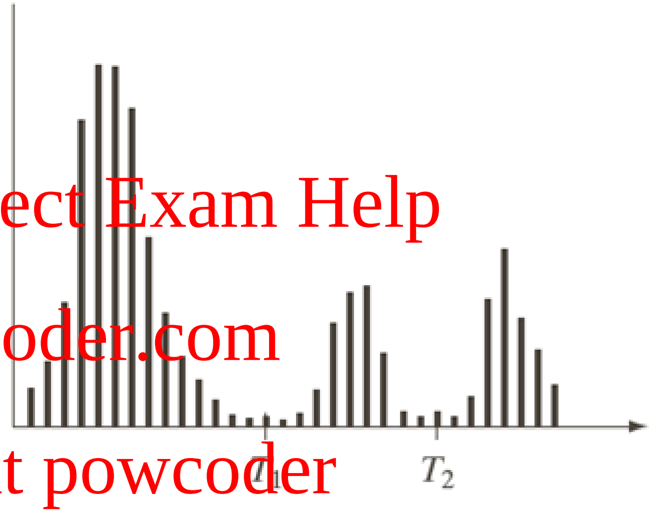
```
>> p=imread('paper1.tif');
>> imshow(p),figure,imshow(p>241)
```


FIGURE 9.3 *The paper image and result after thresholding.*

Simple Thresholding

- **Double Thresholding**

$$g(x,y)=\begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \le T_2 \\ c & \text{if } f(x,y) \le T_1 \end{cases}$$



How to predict appropriate threshold?

```
>> n=imread('nodules1.tif');
>> imshow(n);
>> n1=im2bw(n,0.35);
>> n2=im2bw(n,0.75);
>> figure,imshow(n1),figure,imshow(n2)
```


$T = 90$ $T = 192$

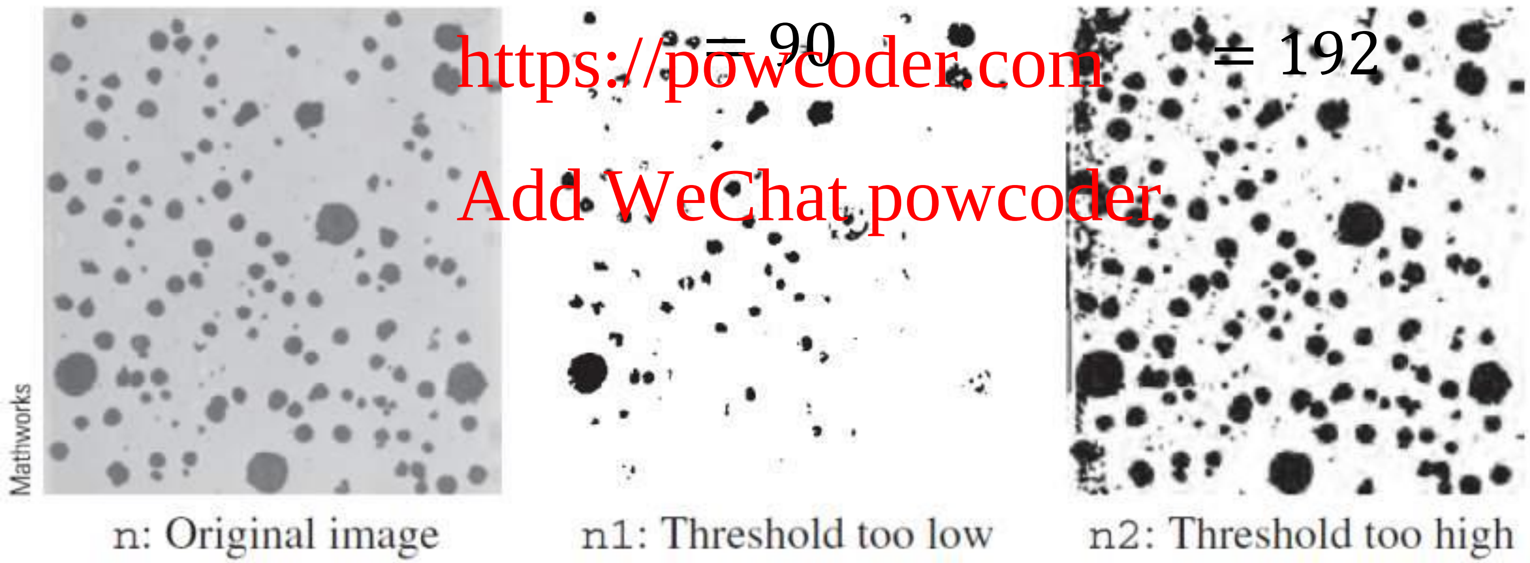


FIGURE 9.6 *Attempts at thresholding.*

How to predict appropriate threshold?

- **Watch out the histogram**
  - The threshold can exist between two dominant distributions.

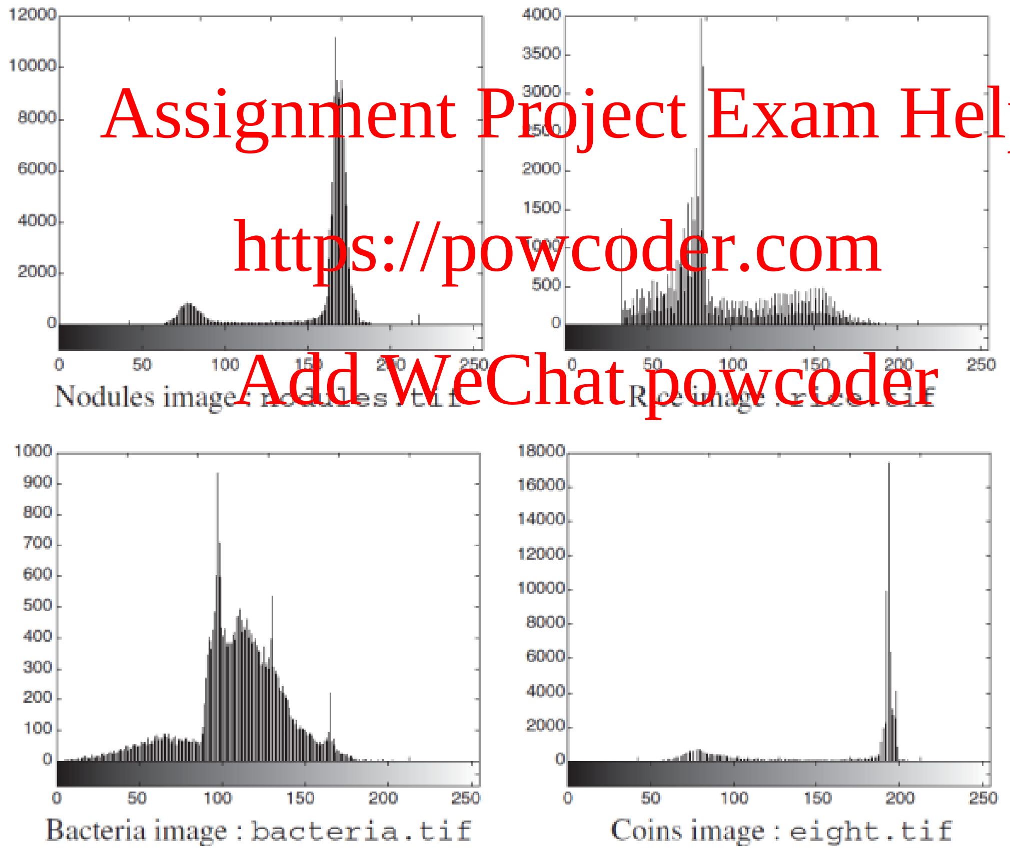


FIGURE 9.7 *Histograms.*

How to predict appropriate threshold?

- **Watch out the histogram**
  - The threshold can exist between two dominant distributions.

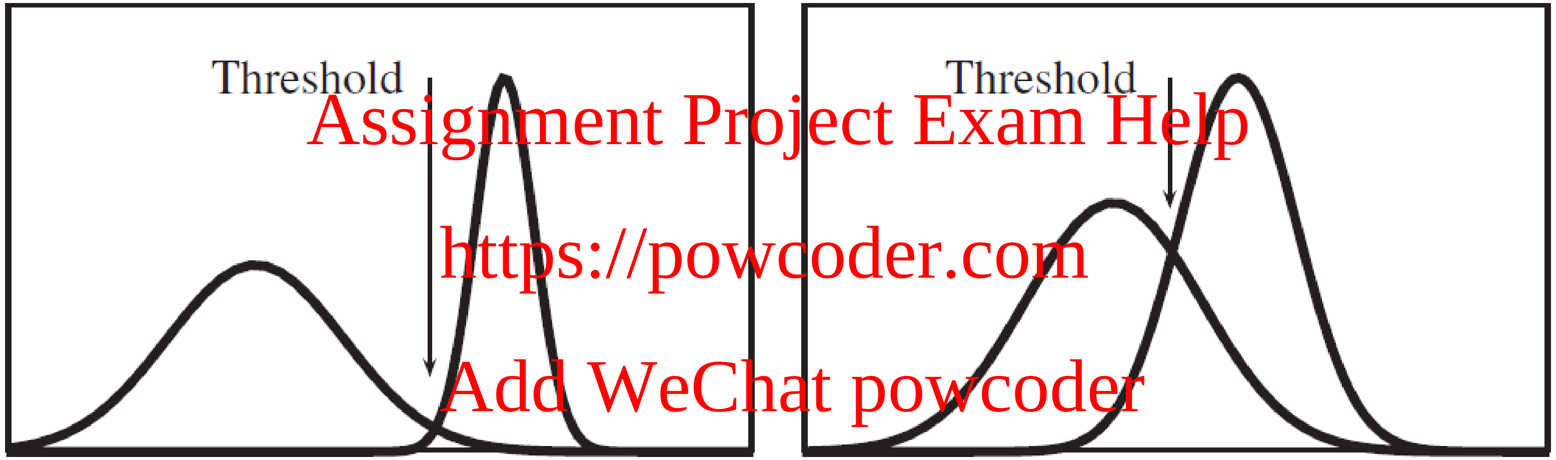
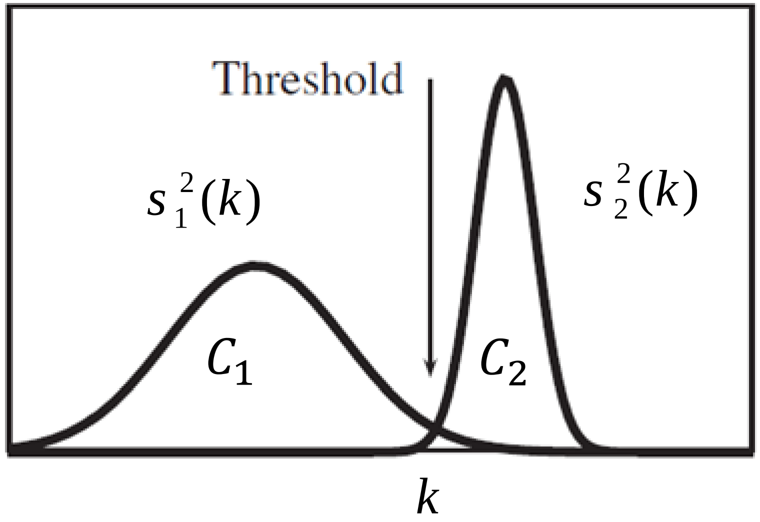


FIGURE 9.8 *Splitting up a histogram for thresholding.*

Otsu's Method: Optimum Global Thresholding

- **Basic assumption: *bimodal* signal (two representative distributions)**

- **Key idea: Exhaustively search** for the threshold that **minimizes the within-class variance** (the variance within the class)
  - **Minimizing** the within-class (intra-class) variance $\sigma_W^2(k)$: A weighted sum of variances of the two classes C_1 and C_2
  - **Maximizing** the between-class (inter-class) variance $\sigma_B^2(k)$



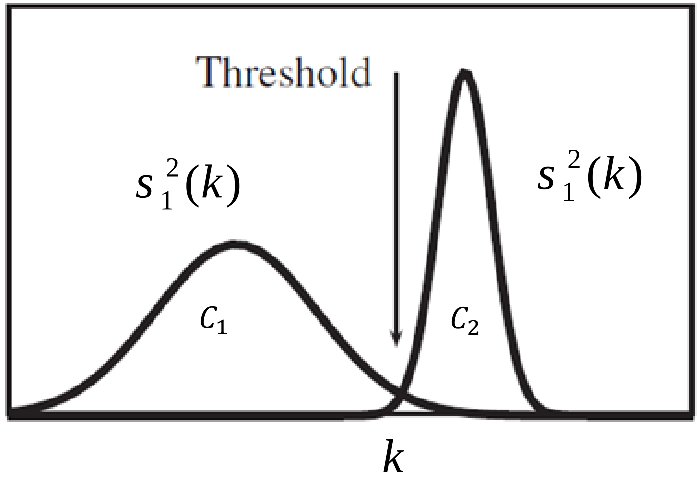
Otsu's Method: Optimum Global Thresholding

- **The within-class (intra-class) variance**

$$\sigma_W^2(k) = q_1(k)\sigma_1^2(k) + q_2(k)\sigma_2^2(k)$$

$q_1(k)$, $m_1(k)$, and $\sigma_1^2(k)$: probability sum, mean, and variance at C_1 $(0 \sim k)$
$q_2(k)$, $m_2(k)$, and $\sigma_2^2(k)$: probability sum, mean, and variance at C_2 $(k+1 \sim L-1)$

- **You can simply seek the optimal threshold k_{opt} by minimizing the within-class variance.**



$$k_{opt} = \arg\min_k \sigma_W^2(k)$$

⇨ Using threshold k_{opt}, segment an image into two regions

For $M \times N$ image with L intensity levels,

$n(i)$: the number of pixels with intensity i

$$p(i) = \frac{n(i)}{MN} \qquad \sum_{i=0}^{L-1} p(i) = 1$$

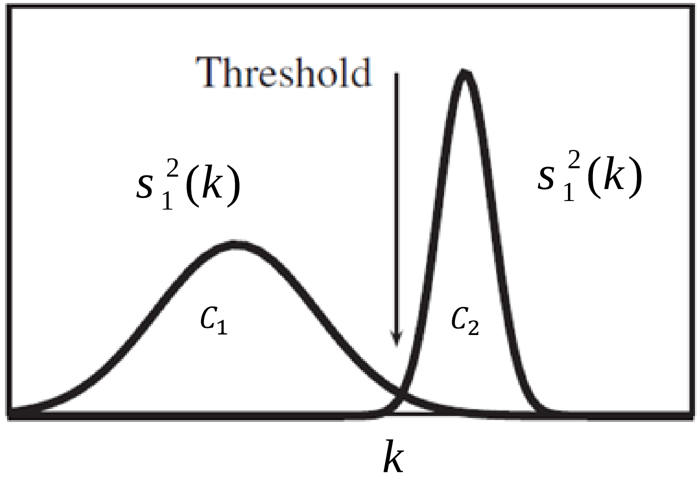
$q_1(k)$, $m_1(k)$, and $\sigma_1^2(k)$: probability sum, mean, and variance at C_1 ($0 \sim k$)
$q_2(k)$, $m_2(k)$, and $\sigma_2^2(k)$: probability sum, mean, and variance at C_2 ($k+1 \sim L-1$)

$$q_1(k) = \sum_{i=0}^{k} p(i) \qquad q_2(k) = \sum_{i=k+1}^{L-1} p(i)$$

$$m_1(k) = \frac{\sum_{i=0}^{k} ip(i)}{\sum_{i=0}^{k} p(i)} = \frac{1}{q_1(k)} \sum_{i=0}^{k} ip(i) \qquad m_2(k) = \frac{\sum_{i=k+1}^{L-1} ip(i)}{\sum_{i=k+1}^{L-1} p(i)} = \frac{1}{q_2(k)} \sum_{i=k+1}^{L-1} ip(i)$$

$$\sigma_1^2(k) = \frac{1}{q_1(k)} \sum_{i=0}^{k} [i - m_1(k)]^2 p(i) = \frac{1}{q_1(k)} \sum_{i=0}^{k} i^2 p_i - m_1^2(k)$$

$$\sigma_2^2(k) = \frac{1}{q_2(k)} \sum_{i=k+1}^{L-1} [i - m_2(k)]^2 p(i) = \frac{1}{q_2(k)} \sum_{i=k+1}^{L-1} i^2 p(i) - m_2^2(k)$$

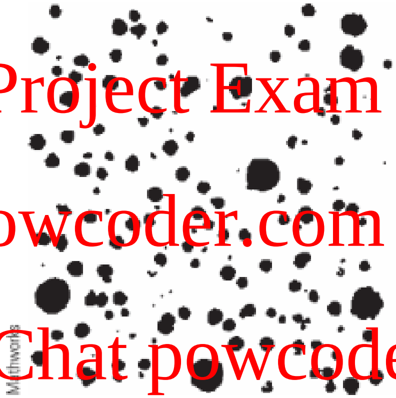


Otsu's method in MATLAB

```
>> tn=graythresh(n)

tn =

    0.5804

>> r=imread('rice.tif');
>> tr=graythresh(r)

tr =

    0.4902

>> b=imread('bacteria.tif');
>> tb=graythresh(b)

tb =

    0.3765

>> e=imread('eight.tif');
>> te=graythresh(e)

te =

    0.6490
```

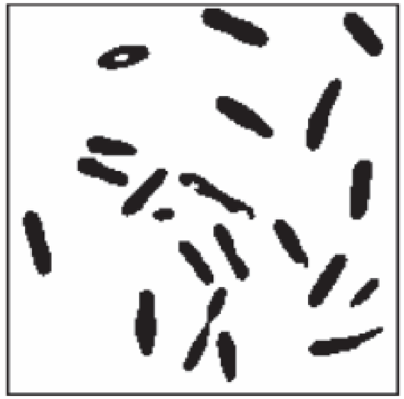
Note) An image is assumed to be normalized to 0~1.

```
>> imshow(im2bw(n,tn))
>> figure,imshow(im2bw(r,tr))
>> figure,imshow(im2bw(b,tb))
>> figure,imshow(im2bw(e,te))
```

Nodules

Rice



Bacteria

Coins

Adaptive Thresholding Using Moving Averages

- **Adaptive thresholding based on moving averages**
  - Works well when objects are small with respect to an image size.
  - Quite useful in document processing.

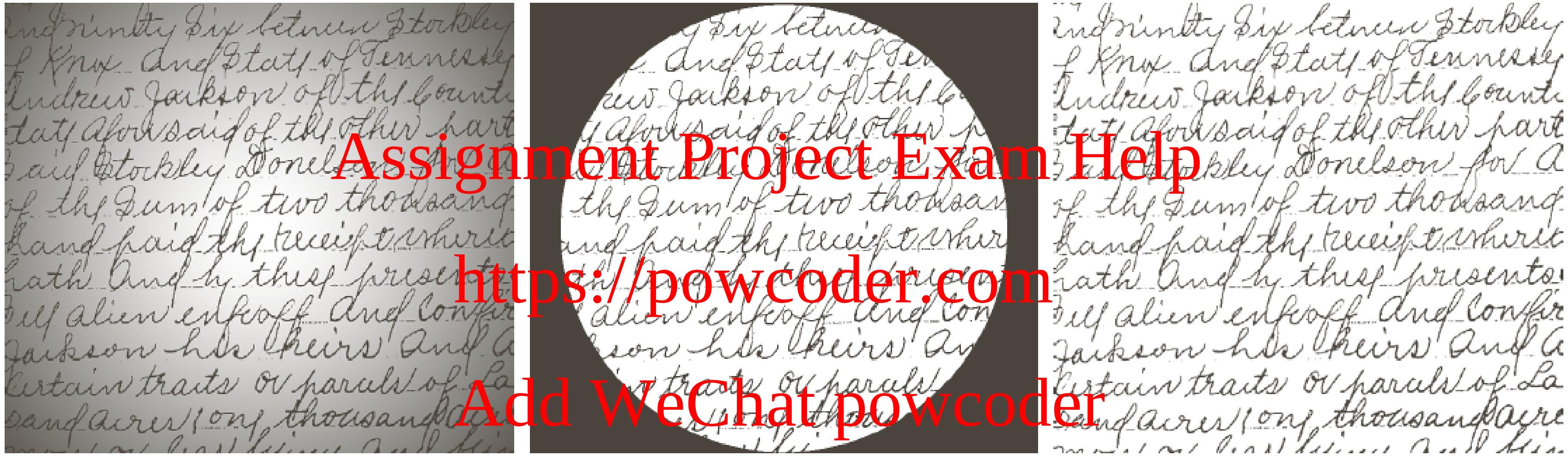
$$g(i,j) = \begin{cases} 1 & if\ I(i,j) > T(i,j) \\ 0 & otherwise \end{cases}$$

$$T(i,j) = b \times m(i,j)$$

Using the mean intensity $m(i,j)$

$$m(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(i+s, j+t)$$

Adaptive Thresholding Using Moving Averages

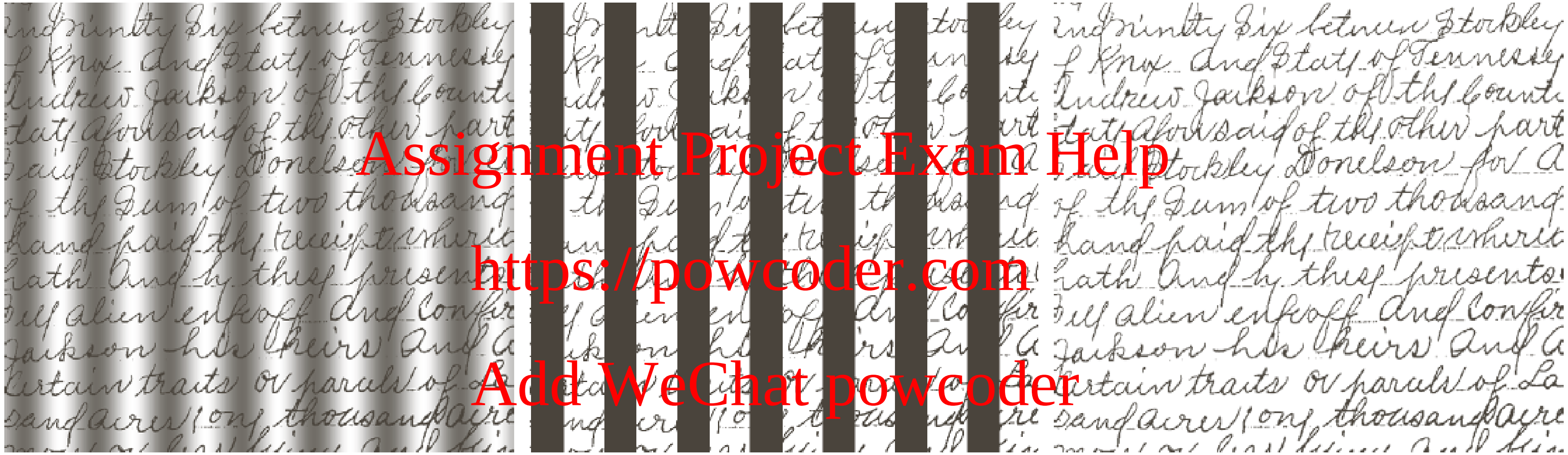


a b c

FIGURE 10.49 (a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

[From "Digital Image Processing", Rafael C. Gonzalez and Richard Eugene Woods]

Adaptive Thresholding Using Moving Averages



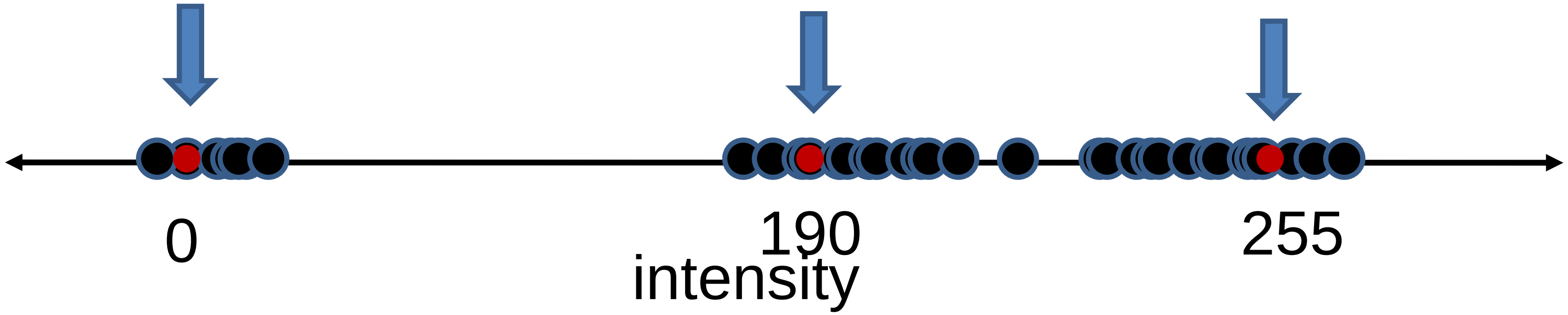
a b c

FIGURE 10.50 (a) Text image corrupted by sinusoidal shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

[From "Digital Image Processing", Rafael C. Gonzalez and Richard Eugene Woods]

Contents

- **Image Segmentation**
  - Thresholding
  - Simple thresholding
  - Otsu method
  - Adaptive thresholding
- **K-mean clustering**



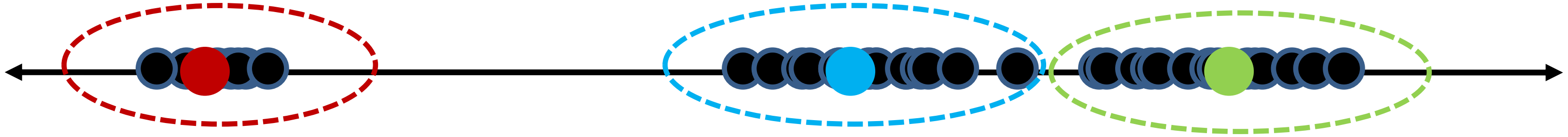
- Goal: choose three “centers” as the *representative* intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center ci:

$$\sum_{\text{clusters } i} \; \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Clustering

- **With this objective, it is a "*chicken and egg*" problem:**
  - If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.

  - If we knew the **group memberships**, we could get the centers by computing the mean per group.



K-means Clustering

- **Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.**

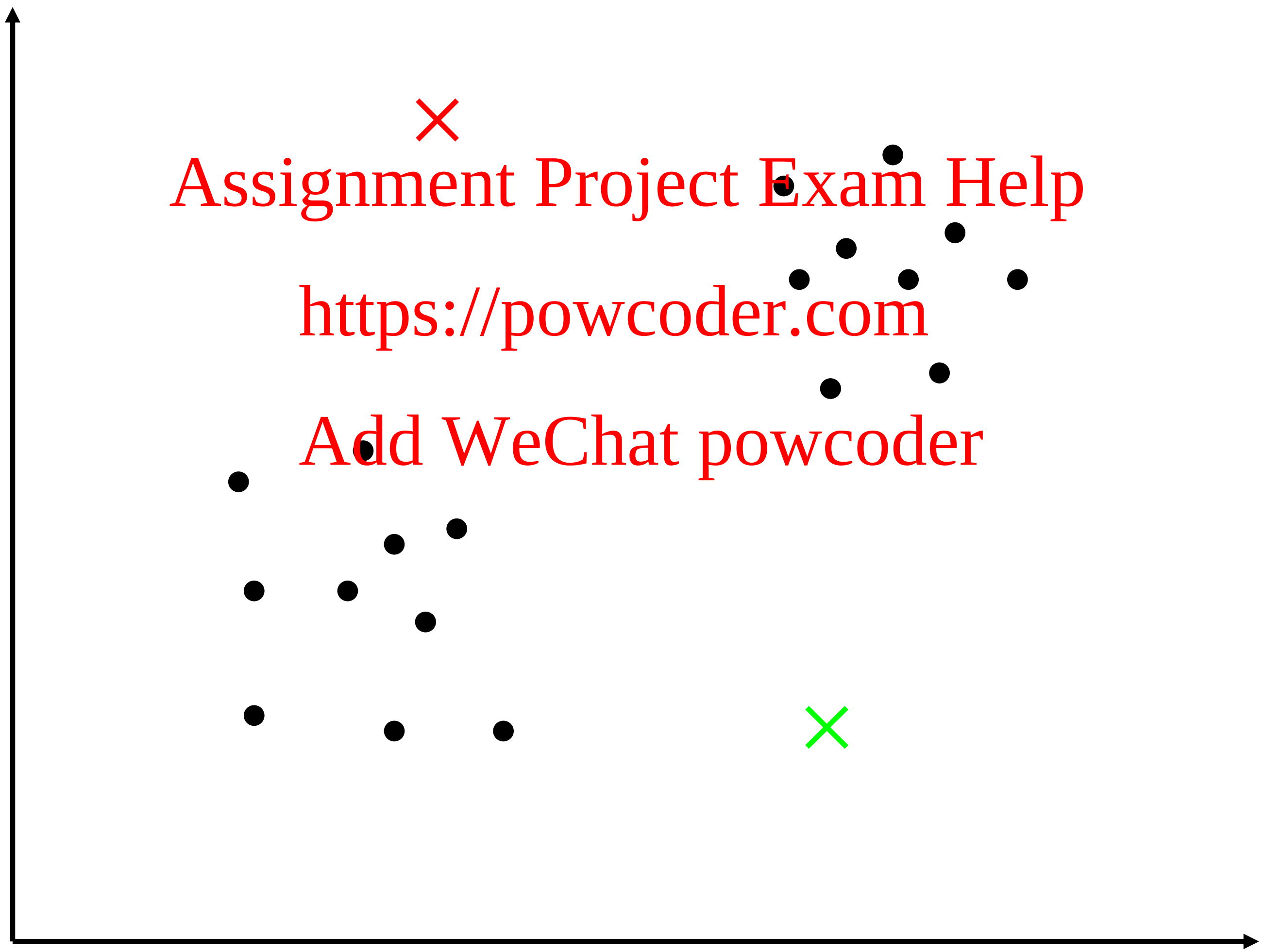
  1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean of points in cluster i
  4. If c_i have changed, repeat Step 2

Properties

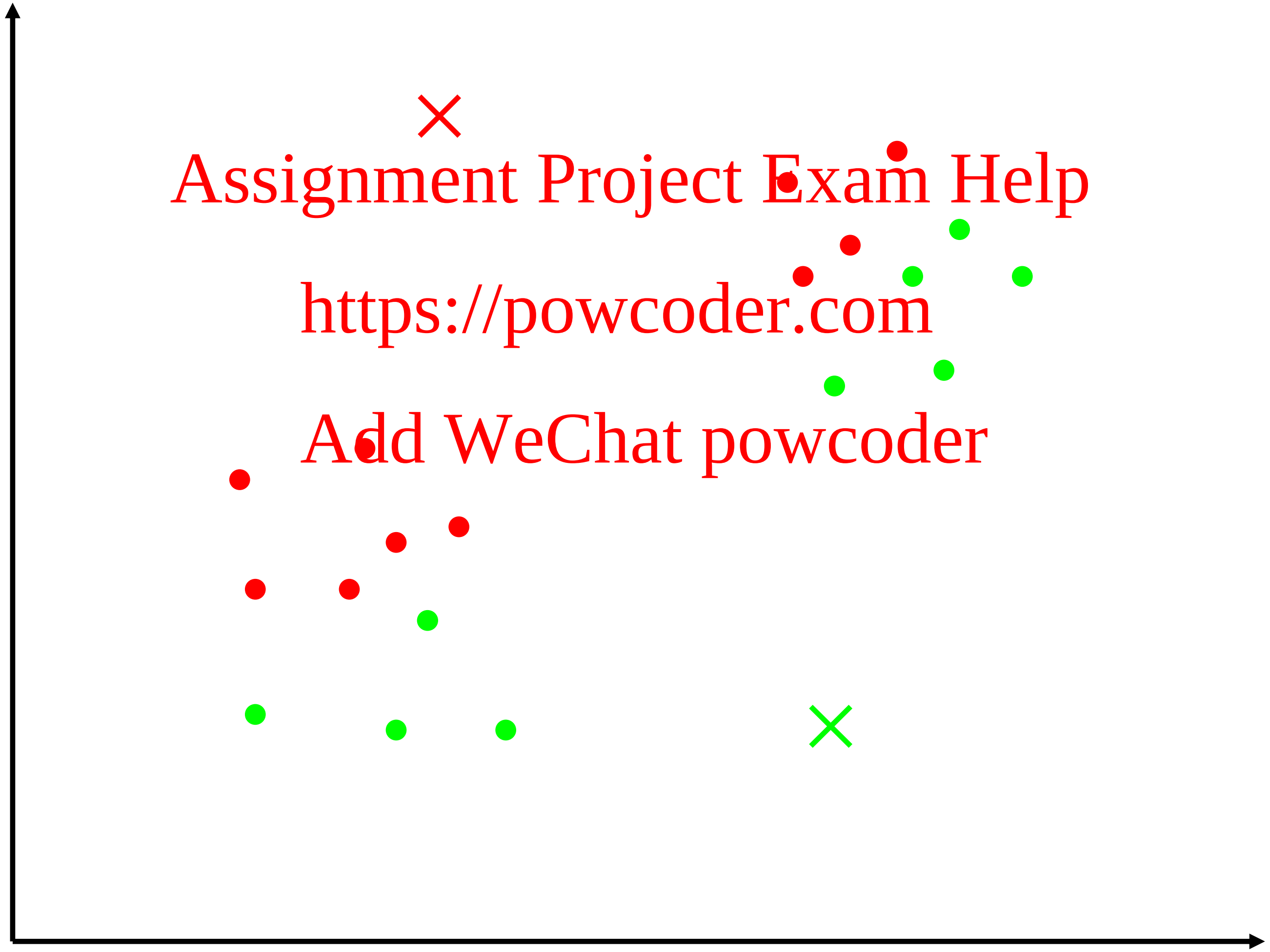
- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

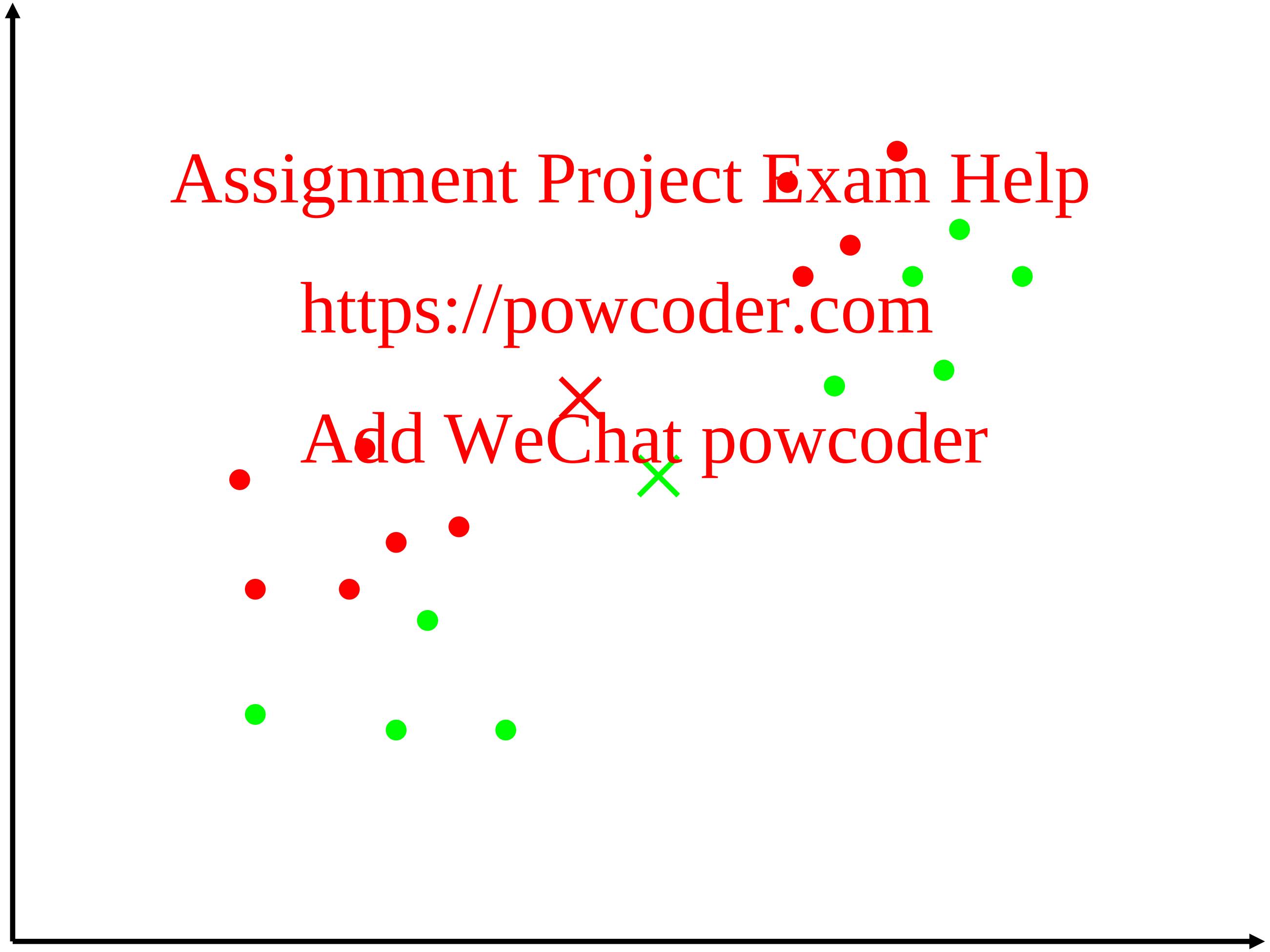
K-means Clustering



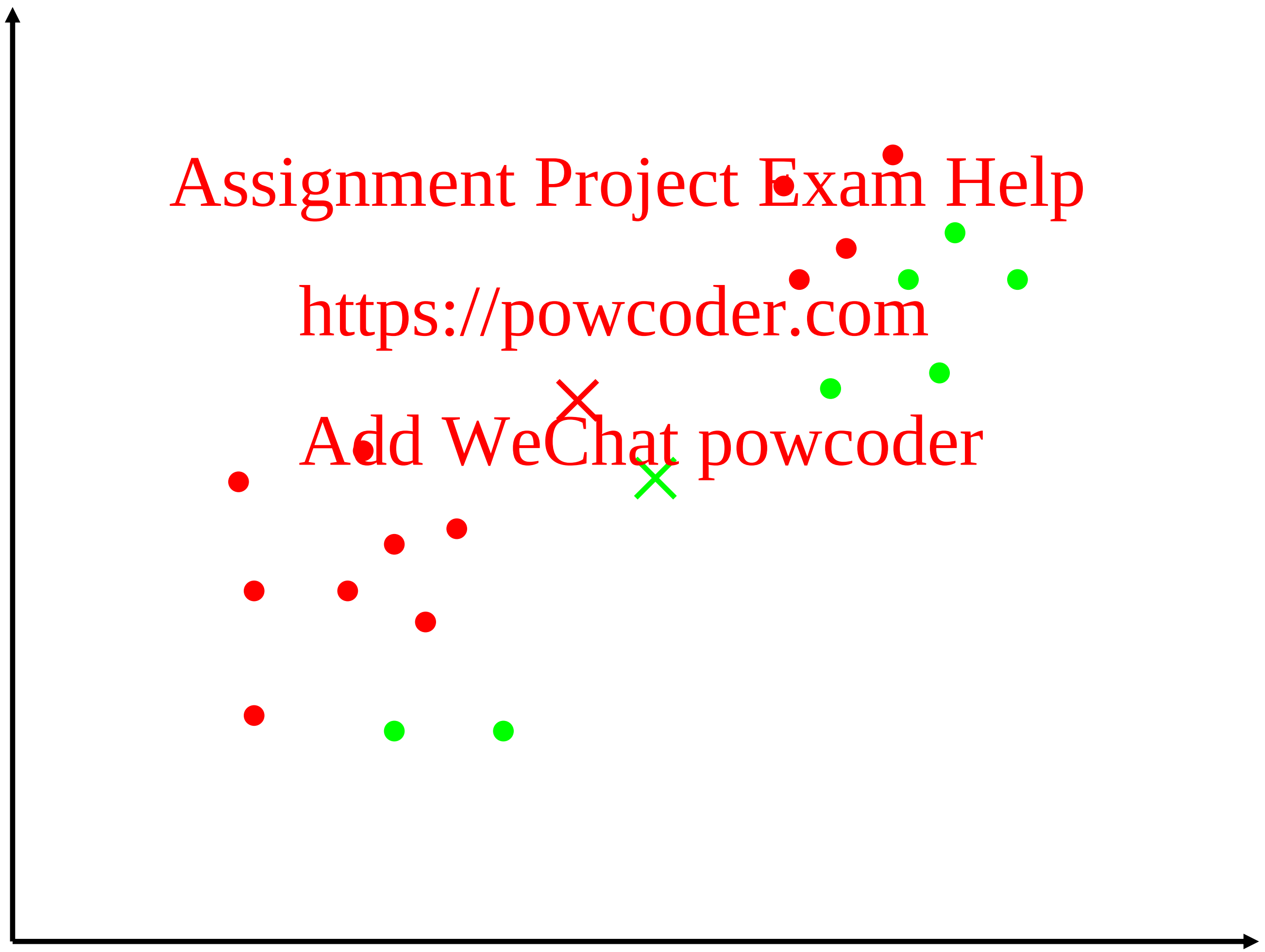
K-means Clustering



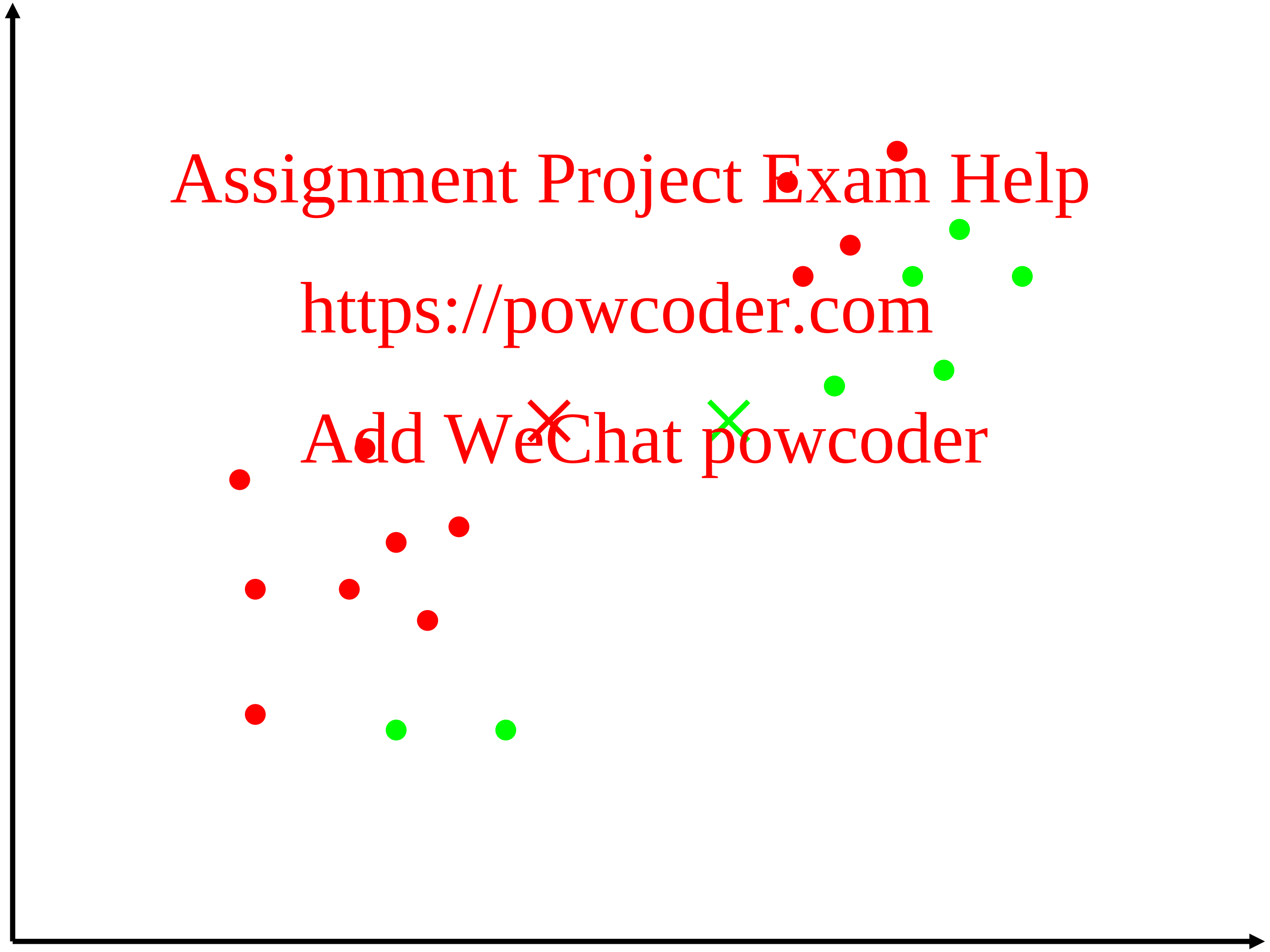
K-means Clustering



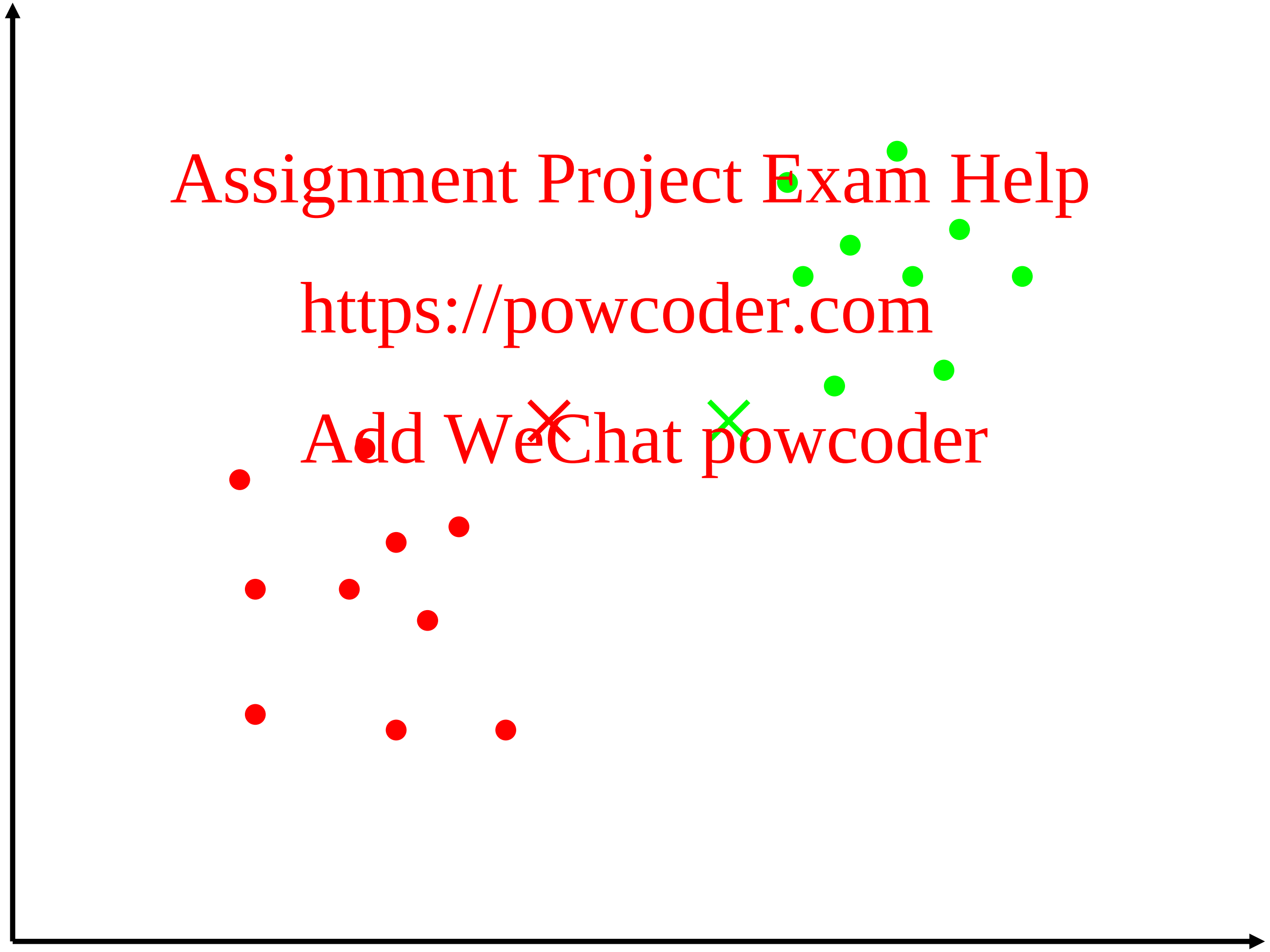
K-means Clustering



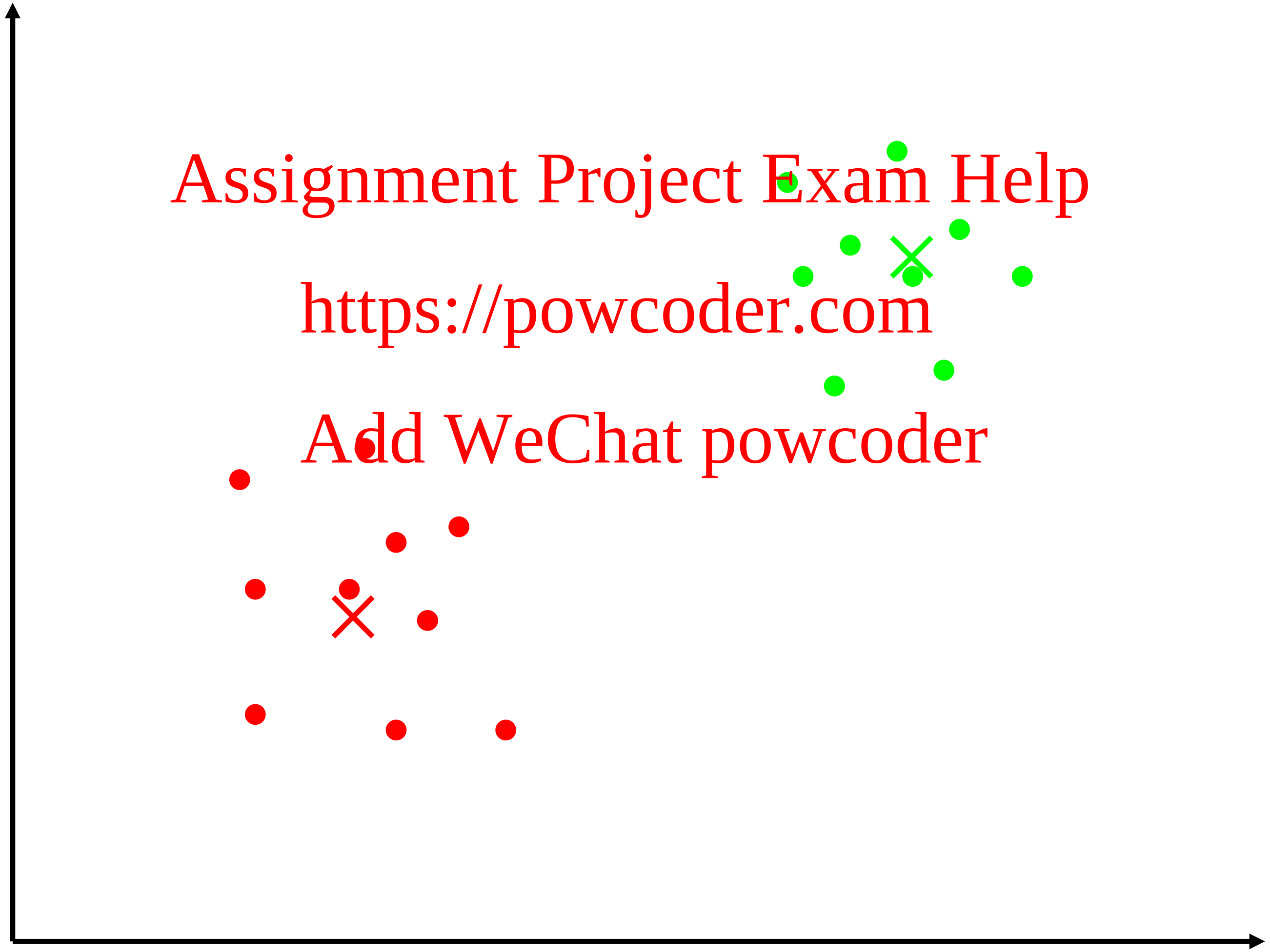
K-means Clustering



K-means Clustering



K-means Clustering



Feature Space on Image Segmentation

- **Depending on what we choose as the *feature space*, we can group pixels in different ways.**
  - Grouping pixels based on **intensity** similarity

  - Clusters based on intensity similarity don't have to be spatially coherent.

  - **Feature space: intensity value (1-d)**

Feature Space on Image Segmentation

K=2


```
img_as_col = double(im(:));
cluster_membs = kmeans(img_as_col, K);

labelim = zeros(size(im));
for i=1:k
   inds = find(cluster_membs==i);
   meanval = mean(img_as_column(inds));
   labelim(inds) = meanval;
end
```

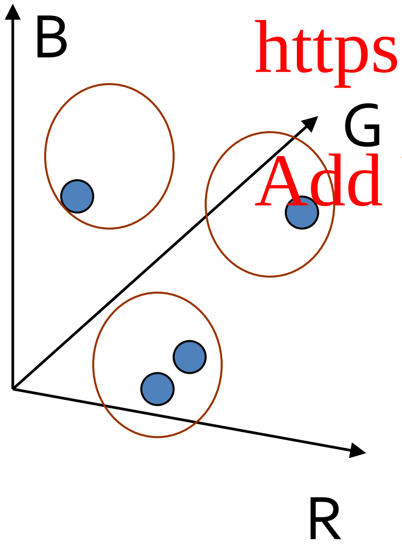
K=3

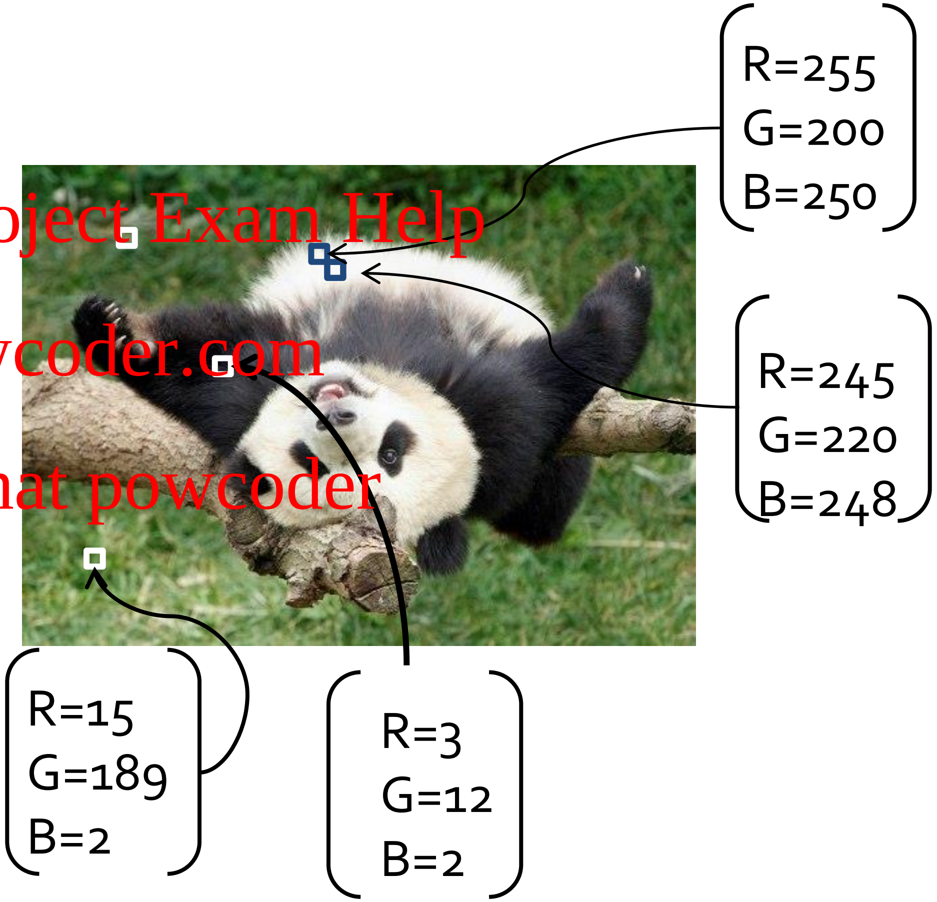
Slide from Kristen Grauman

Feature Space on Image Segmentation

- **Depending on what we choose as the *feature space*, we can group pixels in different ways.**

Grouping pixels based on **color** similarity



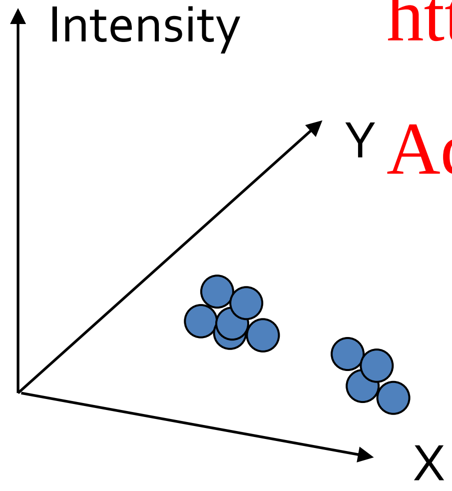


Feature space: color value (3-d)

Feature Space on Image Segmentation

- **Depending on what we choose as the *feature space*, we can group pixels in different ways.**

Grouping pixels based on
intensity+position similarity



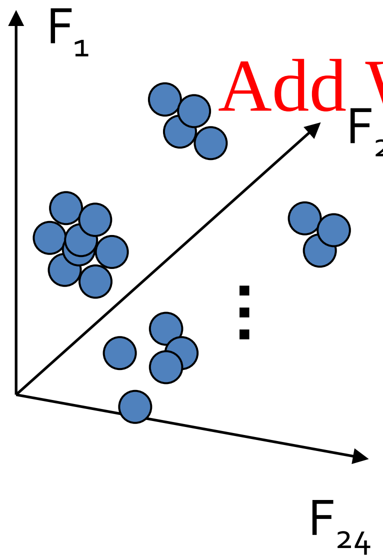
Feature space: 3-d

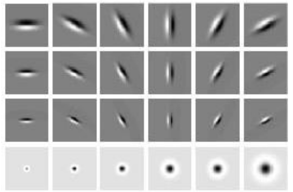
Both regions are black, but if we also include **p osition (x,y),** then we could group the two int o distinct segments; way to encode both simil arity & proximity.

Feature Space on Image Segmentation

- **Depending on what we choose as the *feature space*, we can group pixels in different ways.**

Grouping pixels based on **texture** similarity



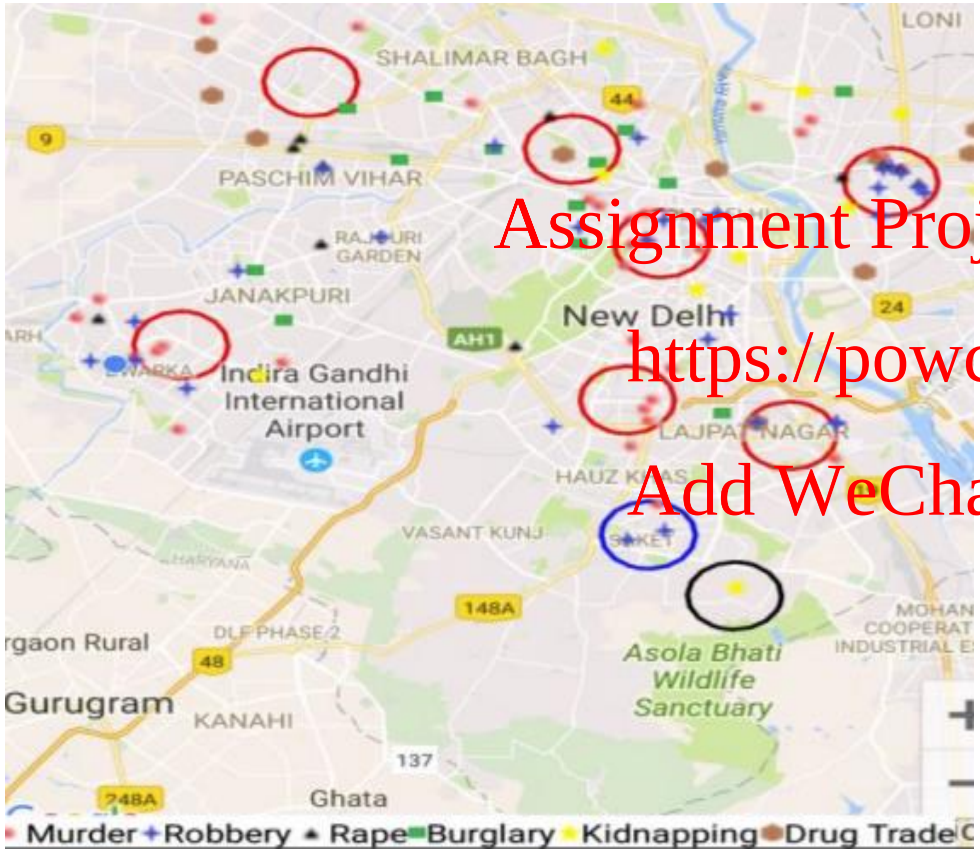


Filter bank of 24 filters

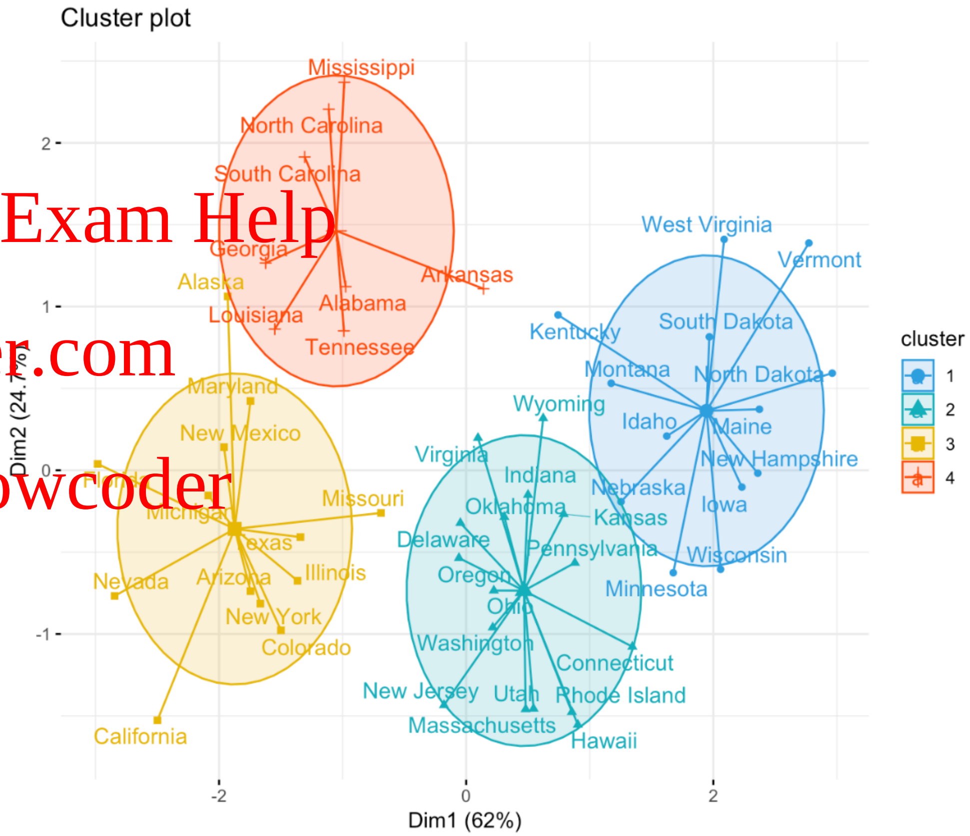
Feature space: filter bank responses (e.g., 24-d)

Applications

- **Not just for images!**



Crime localisation



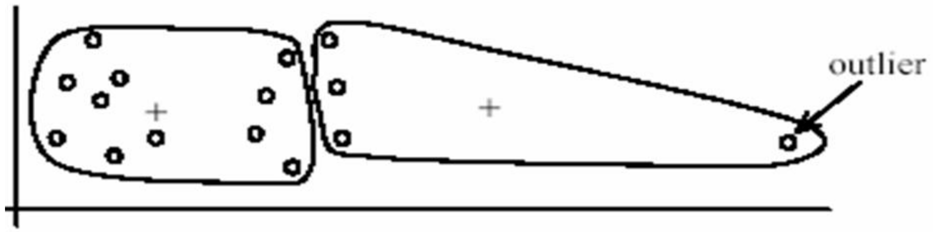
Data analysis

https://www.grdjournals.com/uploads/article/GRDJE/V02/I05/0176/GRDJEV02I050176.pdf

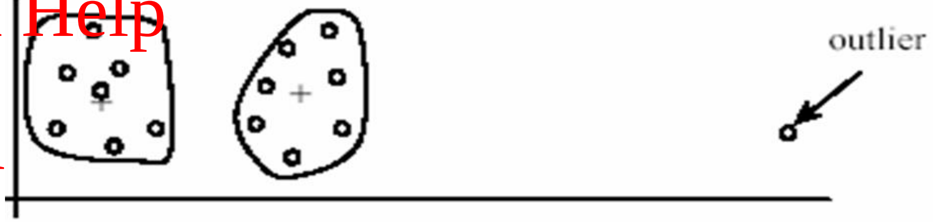
K-means: Summary

- **Pros**
  - Simple, fast to compute
  - Converges to local minimum of within-cluster squared error

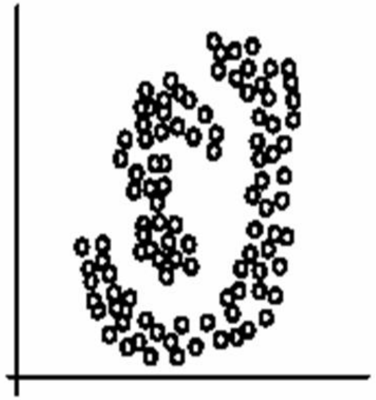
- **Cons/issues**
  - Setting k?
  - Sensitive to initial centers
  - Sensitive to outliers
  - Detects spherical clusters
  - Assuming means can be computed



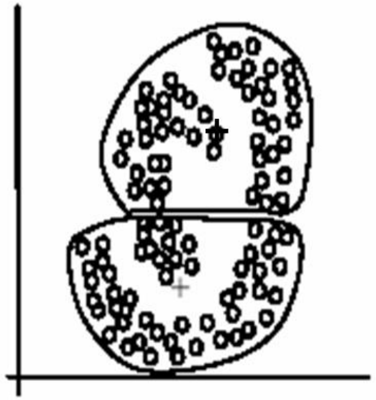
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters

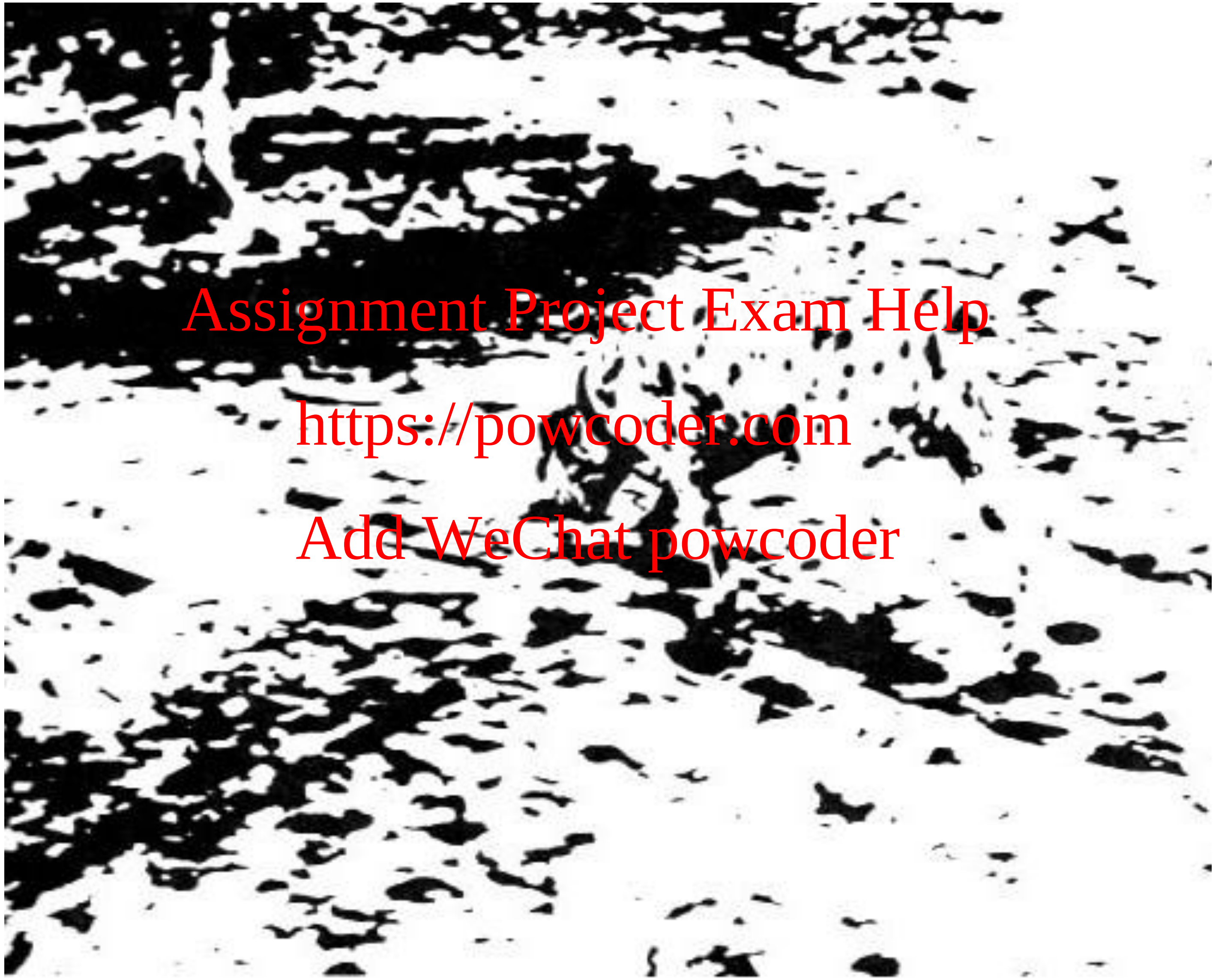


(B): *k*-means clusters

Contents

- **Unsupervised Segmentation**
  - Thresholding-based segmentation
  - K-mean clustering
- **Interactive segmentation**

"Traditional" image segmentation



http://optical-illusions.wikia.com/wiki/Emergence

Segmentation as labeling

- Suppose we want to segment an image into foreground and background
  - Binary pixel labeling problem

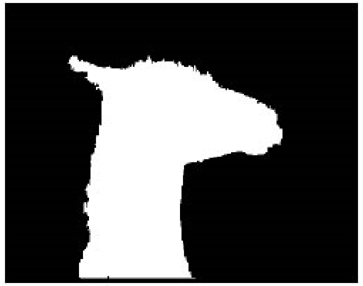
Segmentation as labeling

- Suppose we want to segment an image into foreground and background
  - Binary pixel labeling problem
  - Naturally arises in interactive settings

User scribbles

Labeling by energy minimization

- Define a labeling c as an assignment of each pixel to a class (foreground or background)



- Find the labeling that minimizes a global energy function:

$$E(\mathbf{c}\,|\,\mathbf{x}) = \sum_{i} \underbrace{f_i(c_i, \mathbf{x})} + \sum_{i,j \in e} \underbrace{g_{ij}(c_i, c_j, \mathbf{x})}$$

Pixels

Unary potential (local data term): score for pixel i and label c_i

Neighboring pixels

Pairwise potential (context or smoothing term)

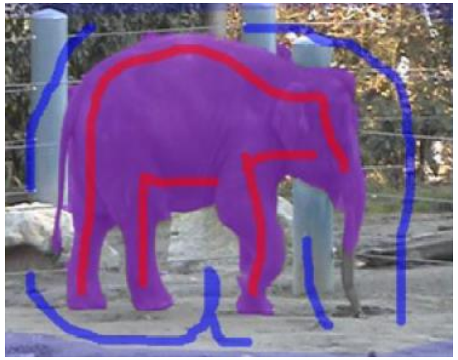
- These are known as Markov Random Field (MRF) or Conditional Random Field (CRF) functions

Segmentation by energy minimization

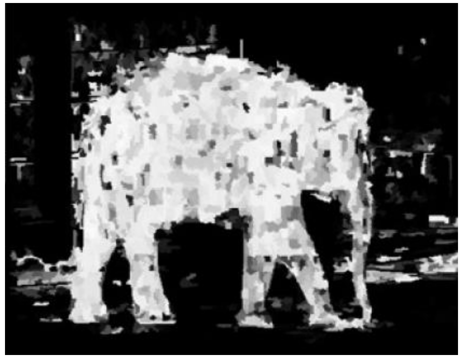
$$E(\mathbf{c} \mid \mathbf{x}) = \sum_{i} f_i(c_i, \mathbf{x}) + \sum_{i,j \in \varepsilon} g_{ij}(c_i, c_j, \mathbf{x})$$

- Unary potentials: $f_i(c_i, \mathbf{x}) = -\log P(c_i \mid \mathbf{x}_i)$
  - Cost is infinity if label does not match the user scribble
  - Otherwise, it is computed based on a color model of user-labeled pixels

User scribbles



P(foreground | x_i)



Segmentation by energy minimization

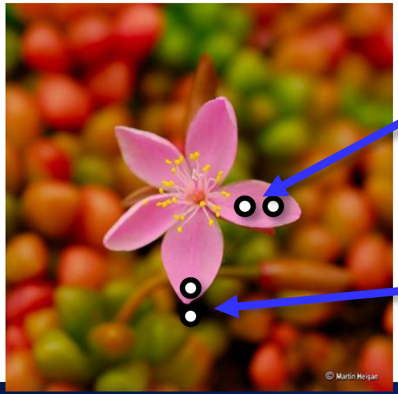
$$E(\mathbf{c} \mid \mathbf{x}) = \sum_{i} f_i(c_i, \mathbf{x}) + \sum_{i,j \in e} g_{ij}(c_i, c_j, \mathbf{x})$$

- Unary potentials: $f_i(c_i, \mathbf{x}) = -\log P(c_i \mid \mathbf{x}_i)$

- Pairwise potentials: $g_{ij}(c, c', \mathbf{x}) = w_{ij} \left| c - c' \right|$

  - Neighboring pixels should have the same label unless they look very different

Affinity between pixels i and j



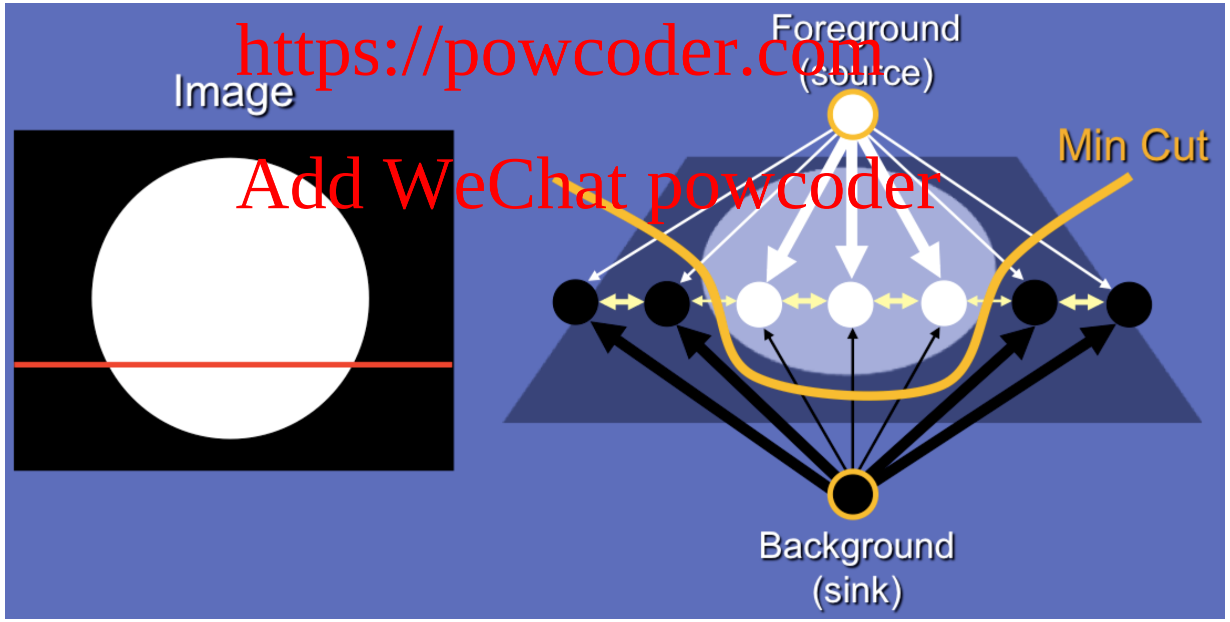
high affinity

low affinity

Segmentation by energy minimization

$$E(\mathbf{c}\,|\,\mathbf{x}) = \sum_{i} f_i(c_i, \mathbf{x}) + \sum_{i,j \in e} g_{ij}(c_i, c_j, \mathbf{x})$$

- Can be optimized efficiently by finding the minimum cut in the following graph:



Y. Boykov and M. Jolly, [Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images](), ICCV 2001

Grabcut

- **Segmentation with bounding box as an interaction**
  - Included as a function in MS PPT

https://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

Next Topic

- How can we compare between a known measurement (the standard) an d the measurement from camera?