Pattern Recognition

Krystian Mikolajczyk

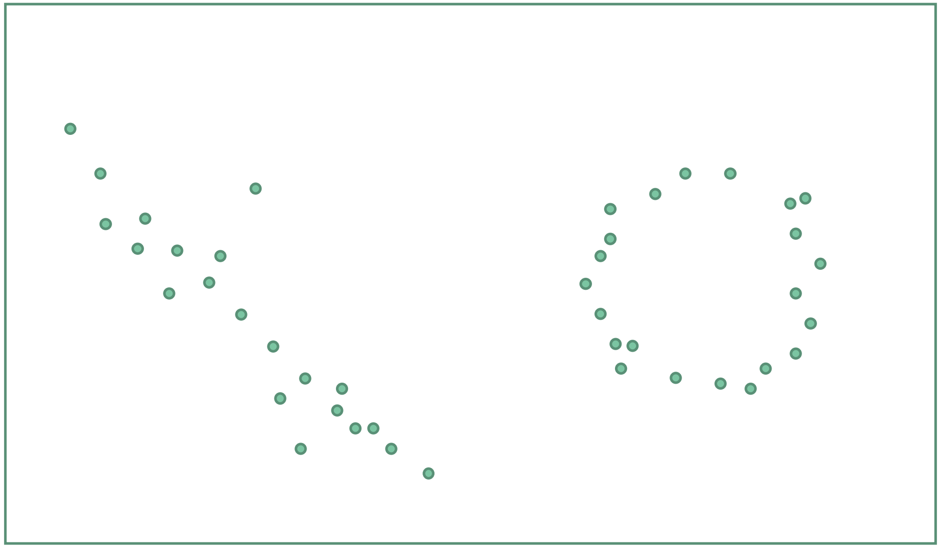
Blackboard

Model Fitting

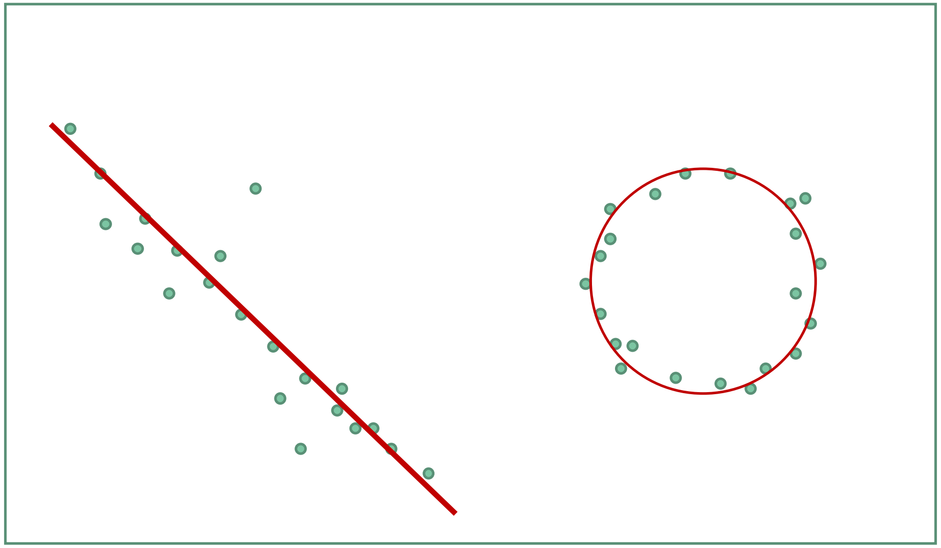
Data Representation

- Non-parametric representation
  - Collection of samples, data points

x_1,y_1
x_2,y_2
.
.
.
.
.
x_N,y_N



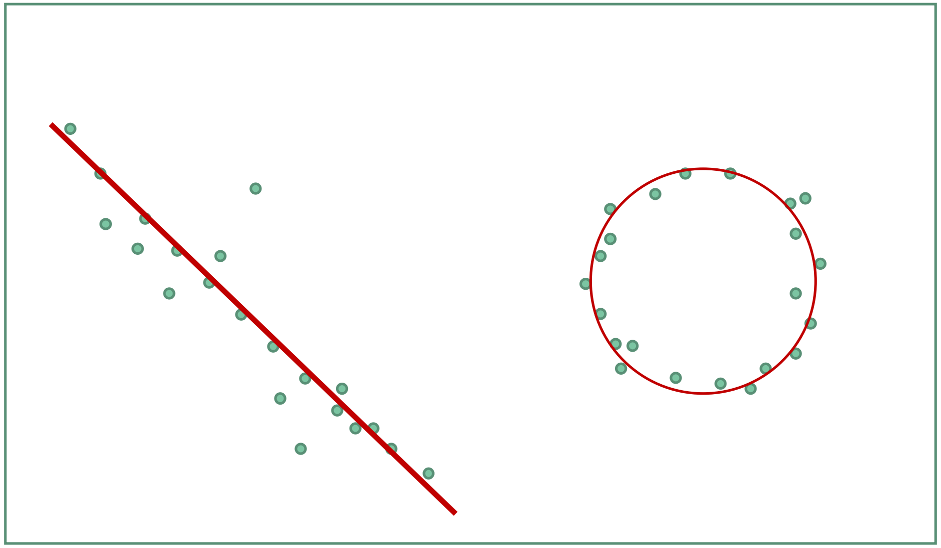
Data Representation



Data Representation

- Parametric representation of a model

$y=ax+b$



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

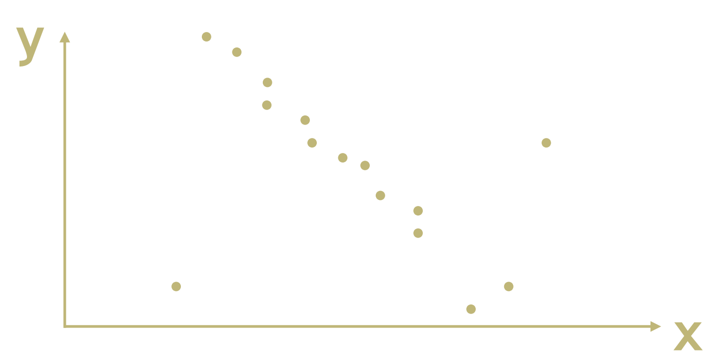
RANSAC

RANSAC (Fischler and Bolles 1981)

- The RANSAC is an algorithm for robust fitting of models in the presence of many data outliers.
  - Given a fitting problem with parameters, estimate the model parameters from the data.
- Assumptions:
  - The model parameters can be estimated from *m* data items out of total *M>>m* items
  - Fit/non-fit can be established for every data item
  - The probability of a randomly selected data item being part of a good model is p_g
  - The probability that the algorithm will exit without finding a good fit (if one exists) is p_{fail}.

Example: find parameters of a line going through the points

$$ax + by + c = 0 \qquad y = ux + v$$



Need 2 points to uniquely define a line (m=2)

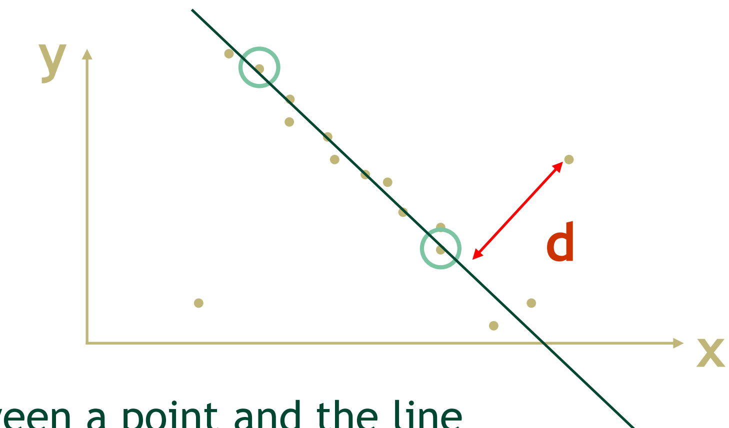
$$P_1(x_1, y_1), P_2(x_2, y_2) \qquad u = \frac{y_1 - y_2}{x_1 - x_2} \qquad v = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}$$

RANSAC

1. selects m data items at random
2. estimates parameters *X=[a,b,...]*
3. finds how many data items *K* out of *M* fit the model with parameter vector *X* within a user given tolerance
4. if *K* is big enough, accept fit and exit with success
5. repeat 1..4 *L* times
6. failed, if you got here

$$\mathrm{d}(ax+by+c=0,(x_0,y_0)) = \frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$$

$$\mathrm{d}(P_1,P_2,(x_0,y_0)) = \frac{|(y_2-y_1)x_0-(x_2-x_1)y_0+x_2y_1-y_2x_1|}{\sqrt{(y_2-y_1)^2+(x_2-x_1)^2}}$$



T_T = *tolerance threshold* maximum distance between a point and the line

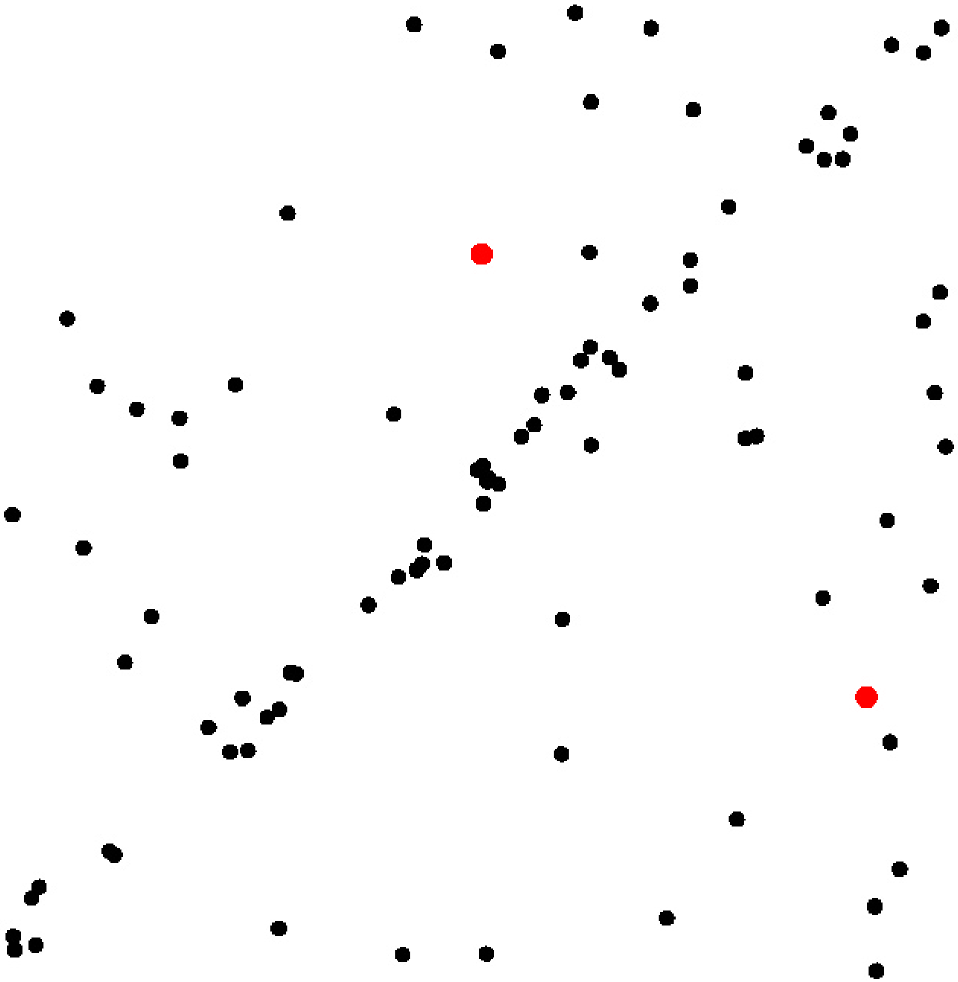
If $d < T_T$ => point fits the model
$Q > T_Q$ = stopping criteria e.g. *0.5*

Q= K / M = quality measure
K – number of inliers (or outliers)

RANSAC

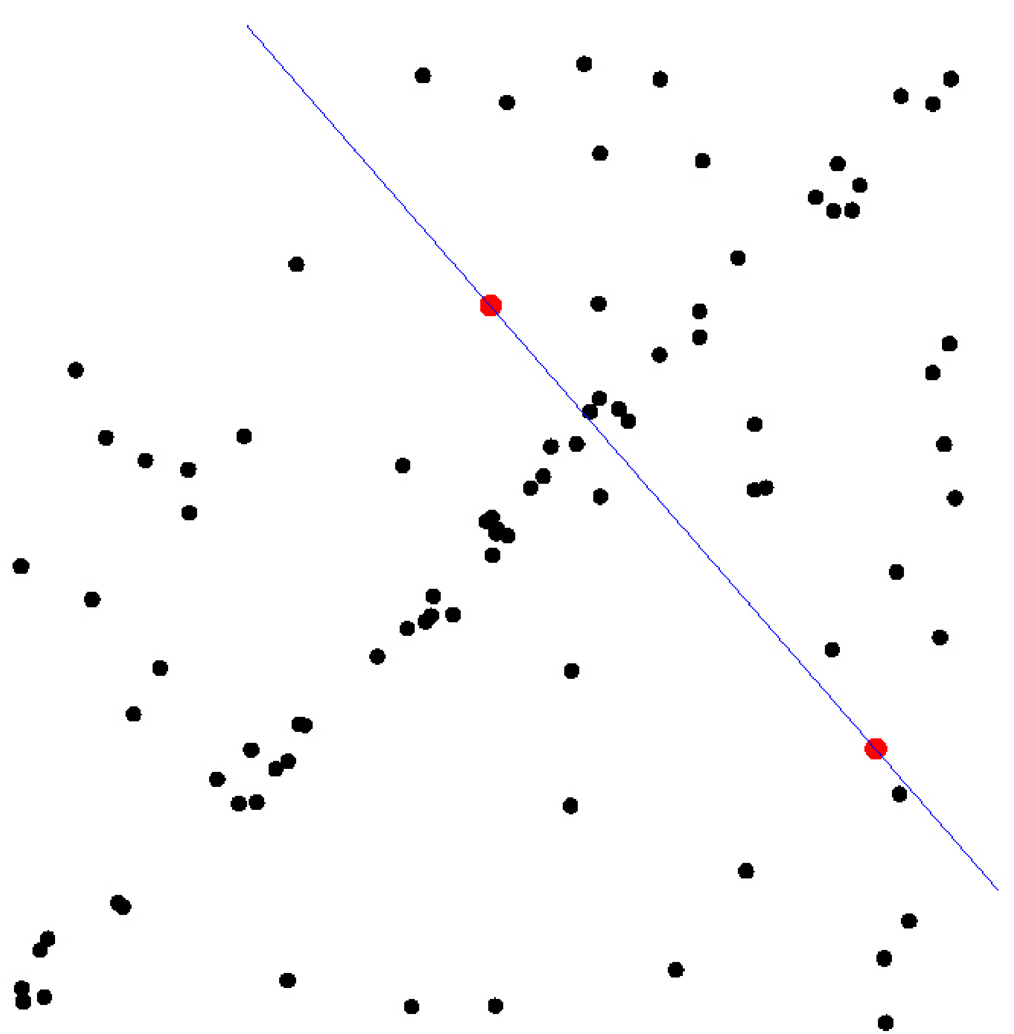
slides credits O. Chum

RANSAC



- Select sample of m points at random

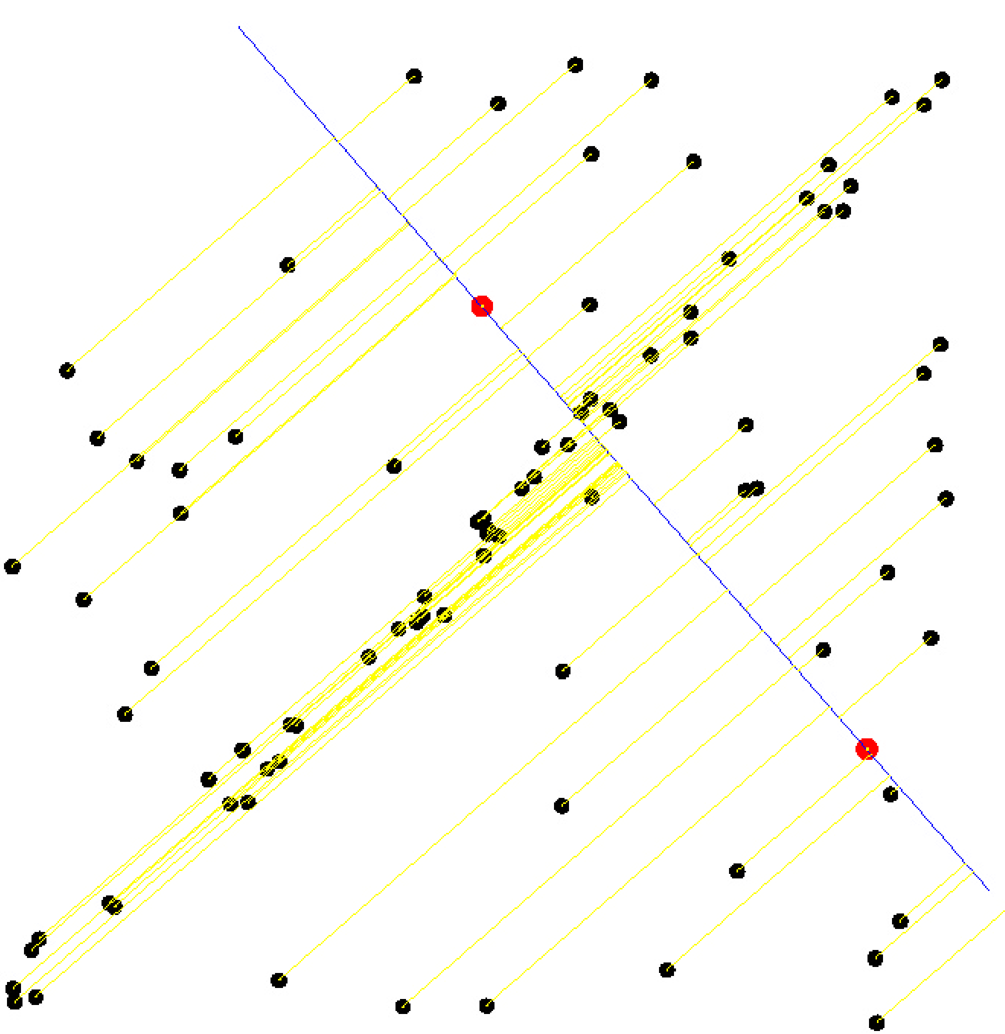
RANSAC



- Select sample of m points at random

- Calculate model parameters that fit the data in the sample

RANSAC



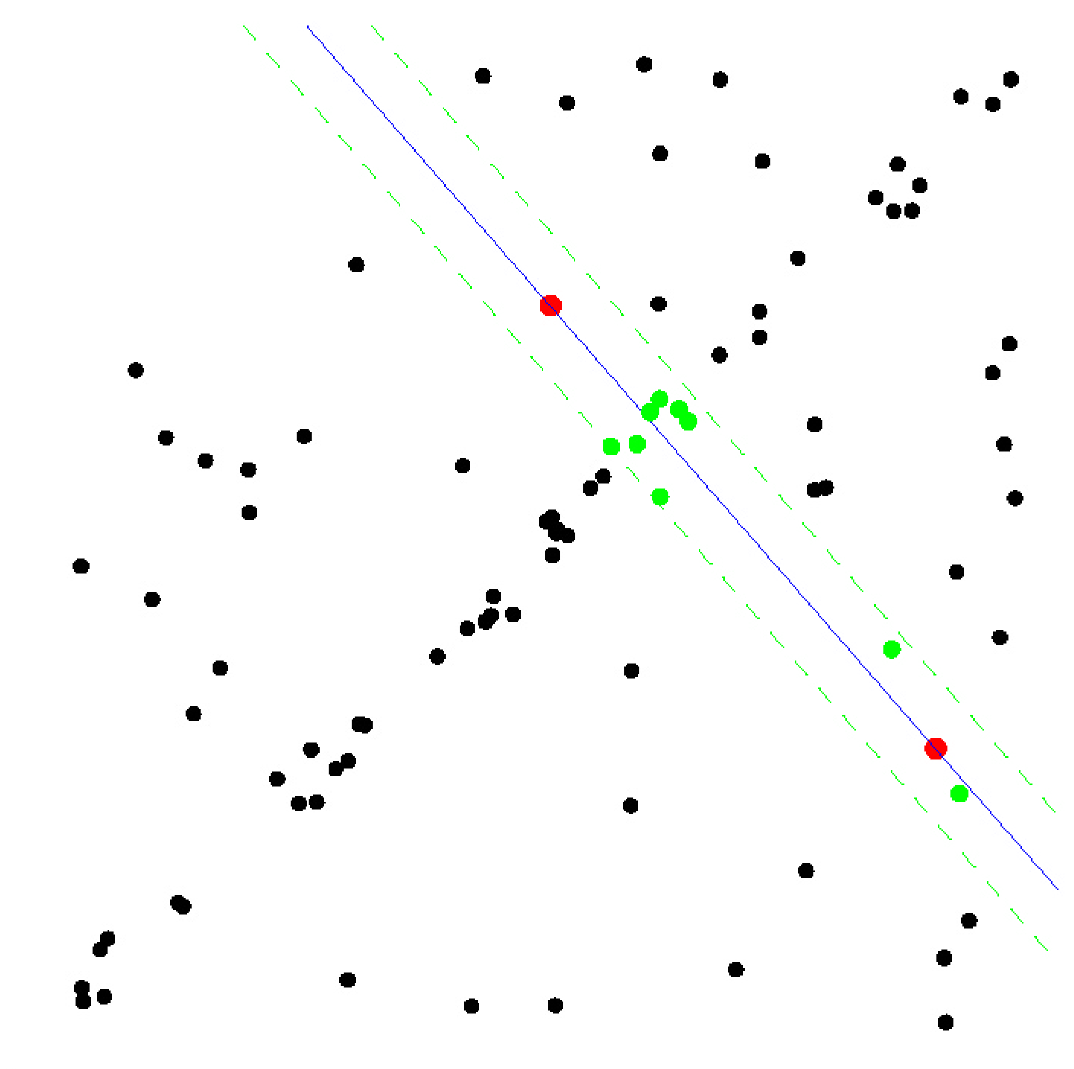
- Select sample of m points at random

- Calculate model parameters that fit the data in the sample

- Calculate error function *d* for each data point

slides credits O. Chum

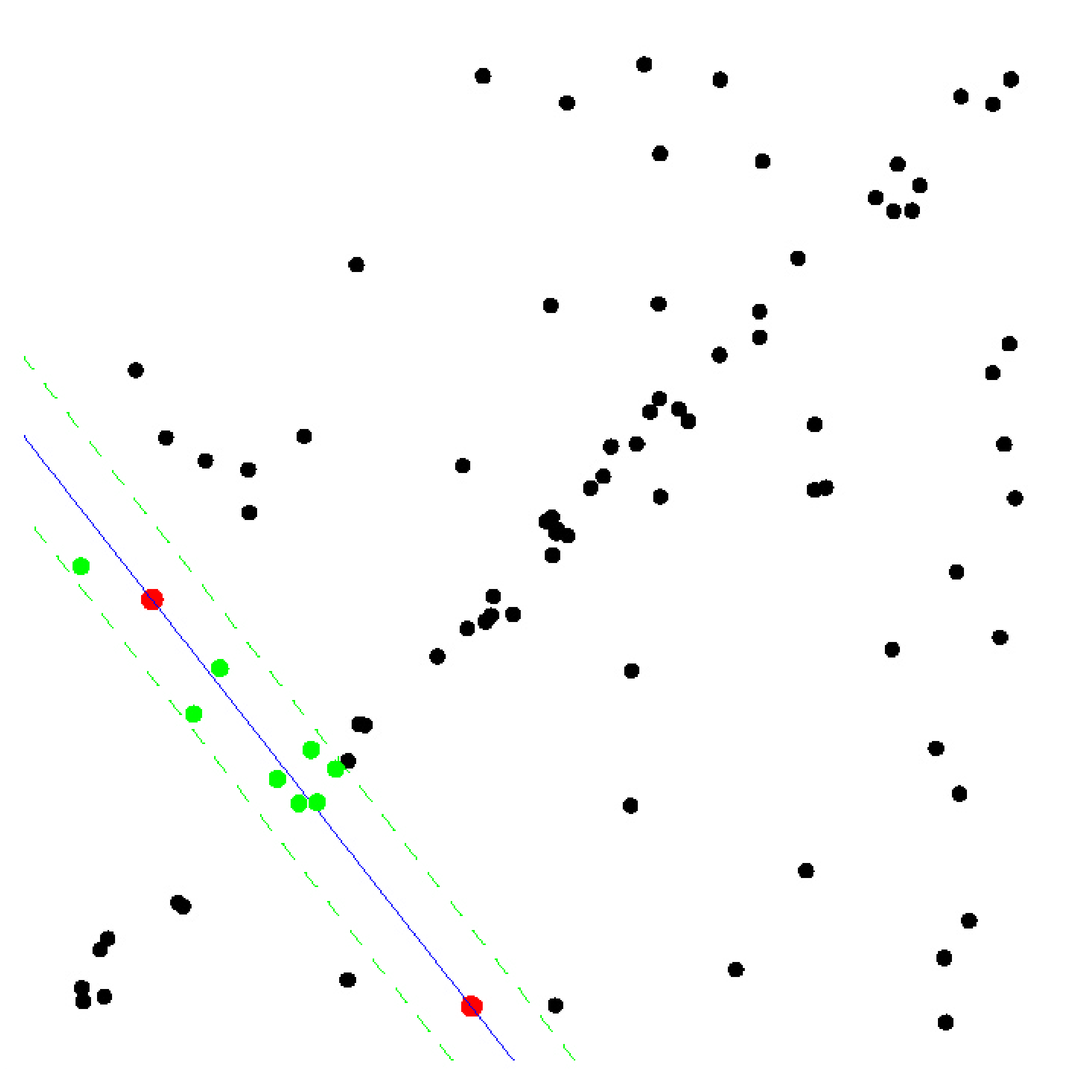
RANSAC



- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis, estimate *Q*

slides credits O. Chum

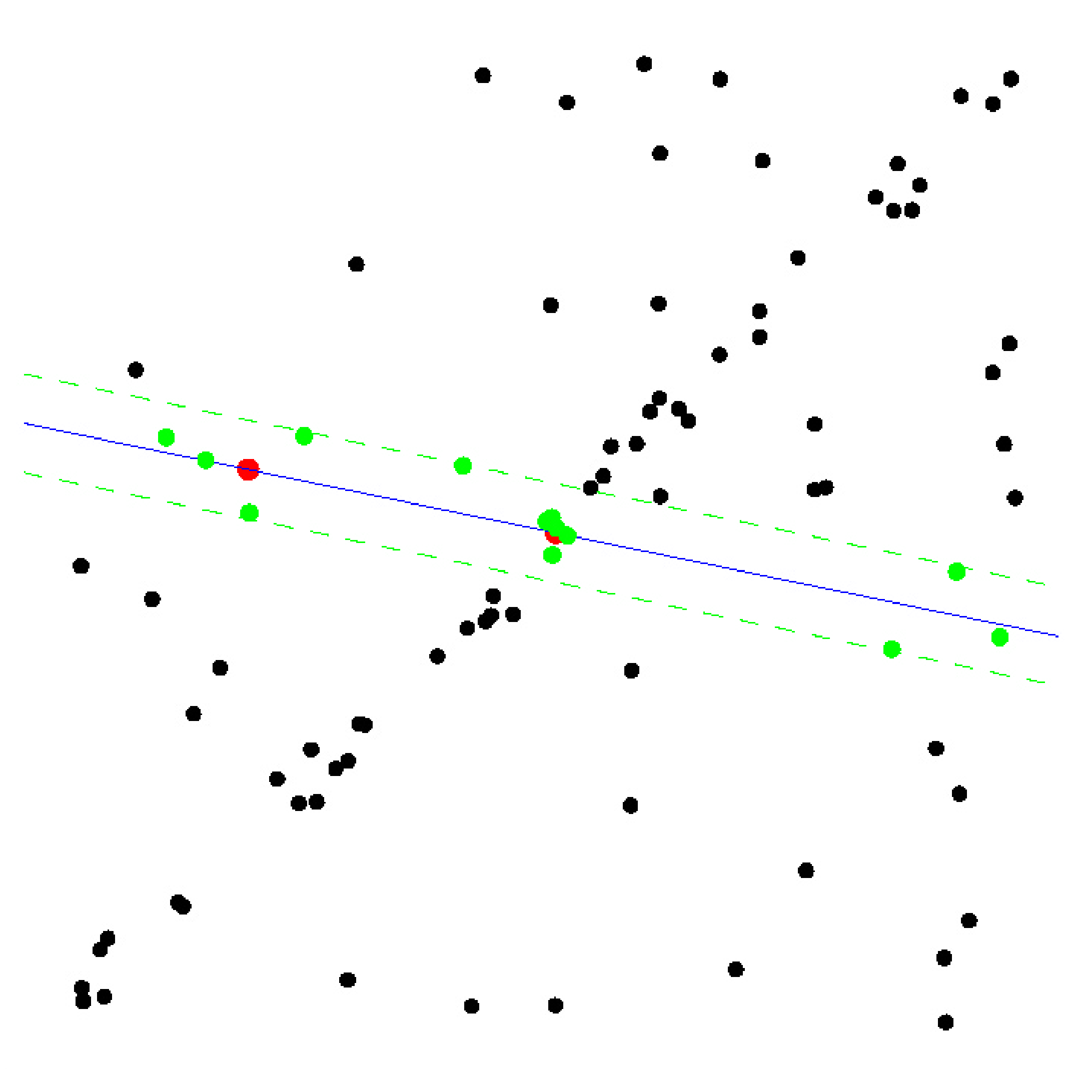
RANSAC



- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis, estimate *Q*
- **Repeat sampling**

slides credits O. Chum

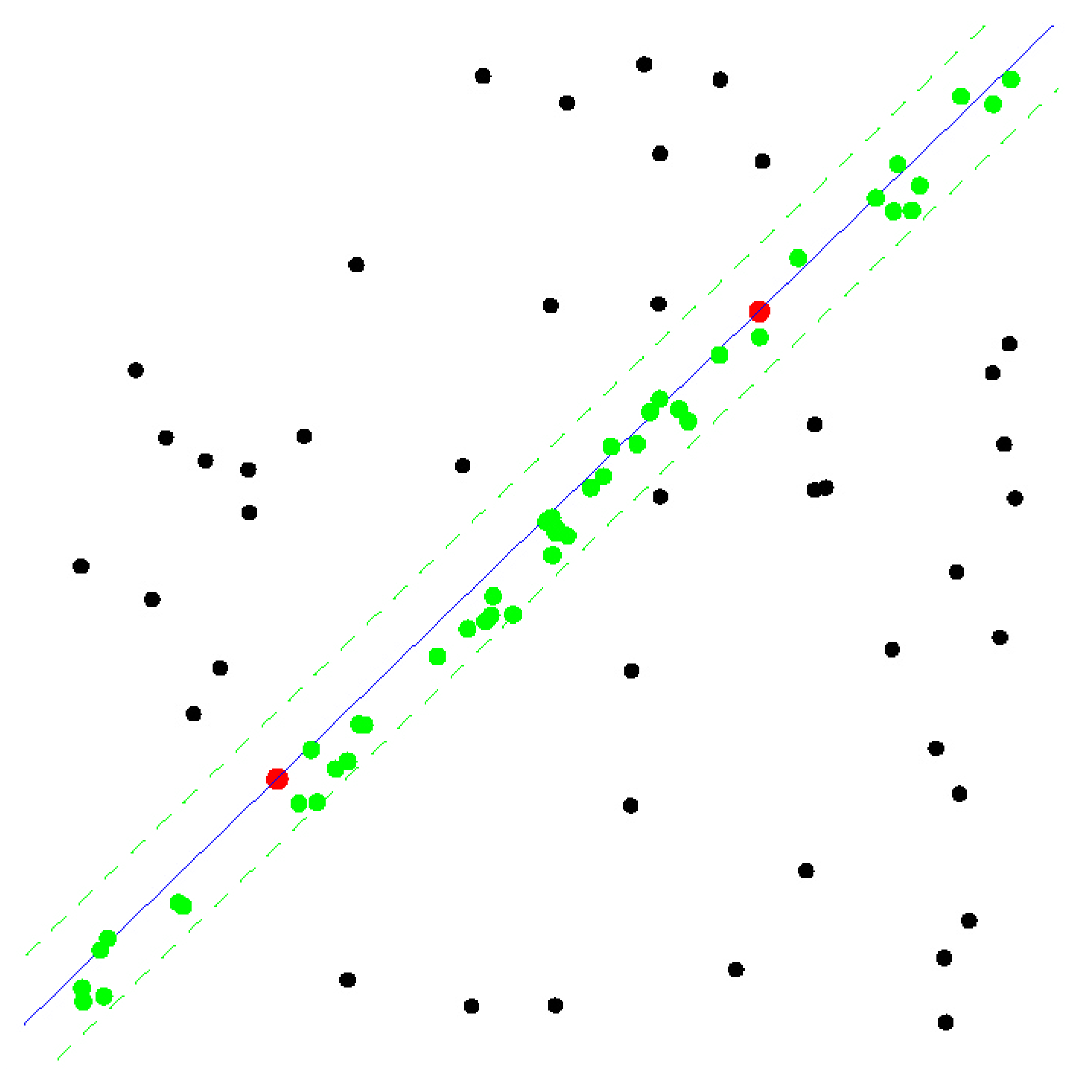
RANSAC



- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis, estimate *Q*
- **Repeat sampling**

slides credits O. Chum

RANSAC



ALL-INLIER SAMPLE

RANSAC time complexity

$O(m\ k\ m_S\ t_m\ M)$

m … number of data items to estimate model parameters

k … number of iterations

t_m … time to compute a single model

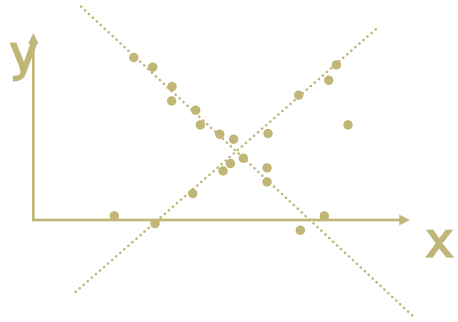
m_S… average number of models per sample

M … total number of items

slides credits O. Chum

RANSAC

- How big *Q* has to be depends on what percentage of the data you think belongs to the structure being fit and how many structures you have in the data.
  - If there are multiple structures then, after a successful fit, remove the fit data and redo RANSAC.



- You can find the number of iterations *L* by:

  p_{fail} = probability of *L* consecutive failures

  - p_{fail} = (prob that a given trial is a failure)^L
  - p_{fail} = (1 - prob that a given trial is a success)^L
  - p_1 = prob that a random data item fits the model $\quad p_1 = \left(\frac{\#inliers}{M}\right)$
  - p_g = (prob that a given trial is a success) $\quad p_g = (p_1)^m$
  - p_{fail} = (1 - (prob that a random data item fits the model)^m)^L $\quad p_{fail} = (1-(p_1)^m)^L$

$$L = \frac{\log(p_{preset\ fail})}{\log(1-(p_1)^m)}$$

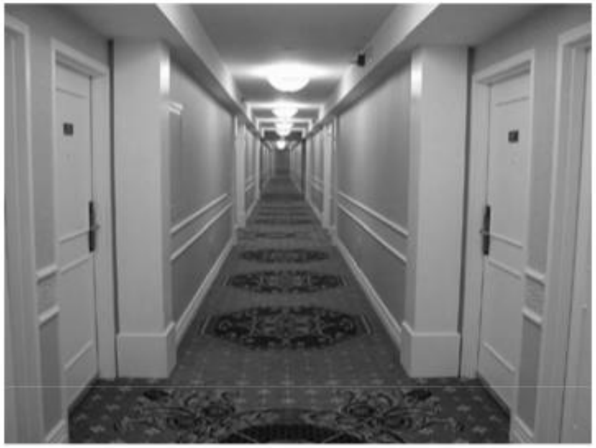
$$p_{preset\ fail} = 1 - p_{success}$$

Usually set very low e.g. 0.01

Hough Transform

Hough Transform

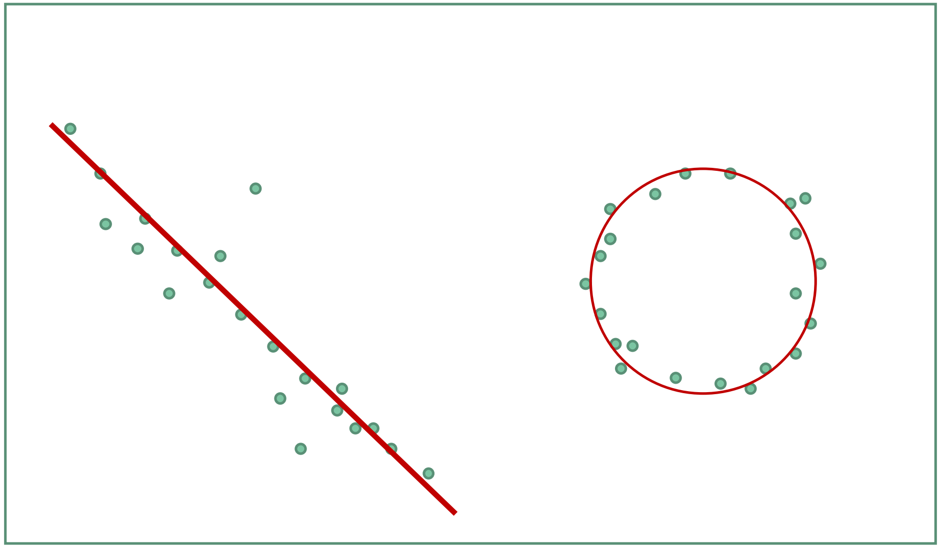
- Image with line structures
  - Point coordinates *P(x,y)* given by white pixels



Hough Transform

- A method to find model parameters that fits the data
  - It was introduced in 1962 (Hough 1962) and first used to find lines in images a decade later (Duda 1972).

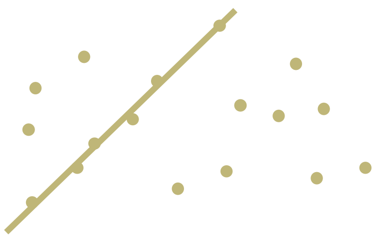
$y=ax+b$

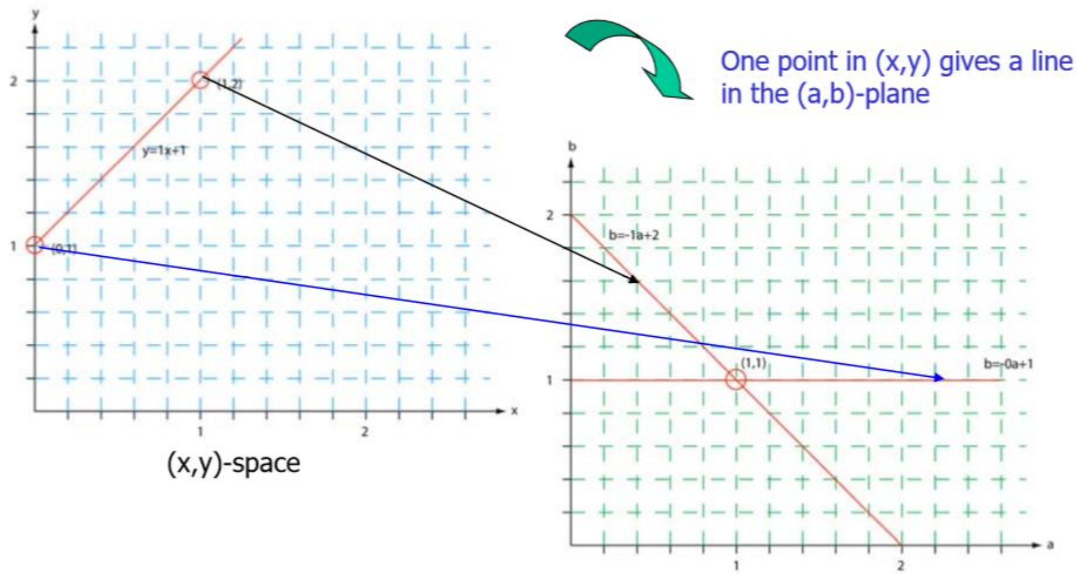


$(x-a)^2+(y-b)^2=r^2$

Hough Transform

- Origin: Detection of straight lines in clutter
  - Basic idea: each candidate point votes for all lines that it is consistent with.
  - Votes are accumulated in quantized array
  - Local maxima correspond to candidate lines
- Representation of a line
  - Usual form $y = a\,x + b$ => $b = y - a\,x$

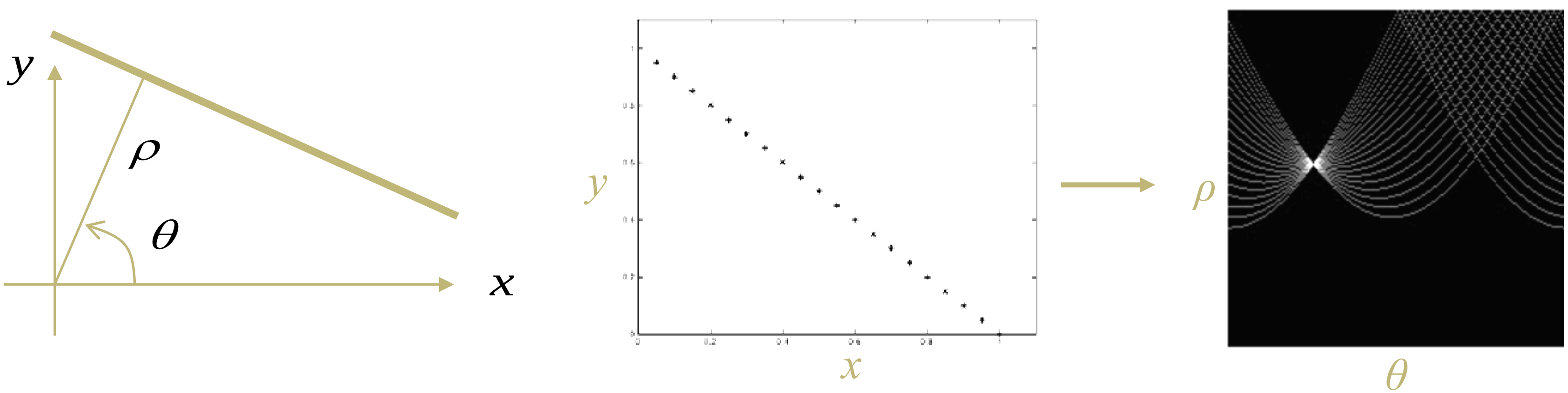




Hough Transform

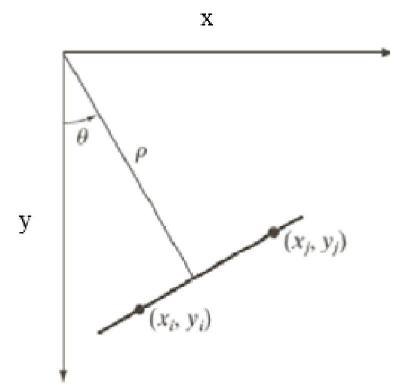
Representation of a line

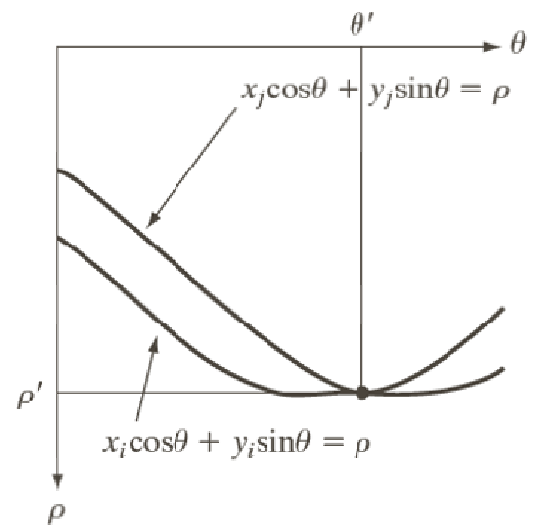
- Usual form $y = a\,x + b$ => $b = y - a\,x$ has a singularity around 90º
- Better parameterization: $x \cos(\theta) + y \sin(\theta) = \rho$



Hough Transform

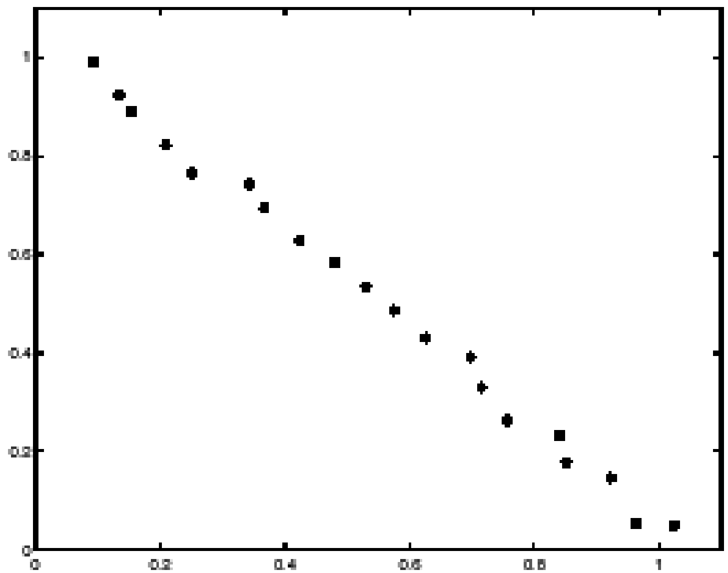
- Each curve in the figure represents the familiy of lines that pass through a particular point (x_i, y_i) in the xy-plane.
- The intersection point (ρ', θ') corresponds to the lines that passes through two points (x_i, y_i) and (x_j, y_j)
- A horizontal line will have θ=0 and ρ equal to the intercept with the y-axis.
- A vertical line will have θ=90 and ρ equal to the intercept with the x-axis.



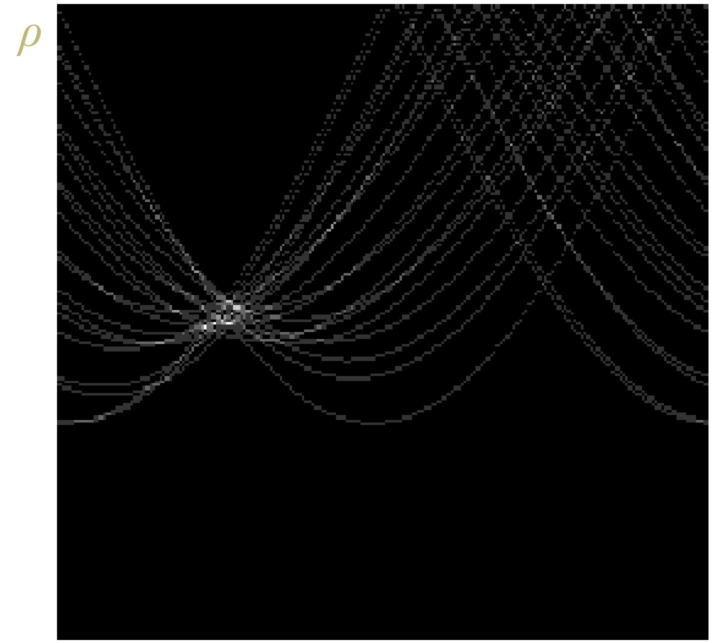


Hough Transform

- Noisy Line



Tokens

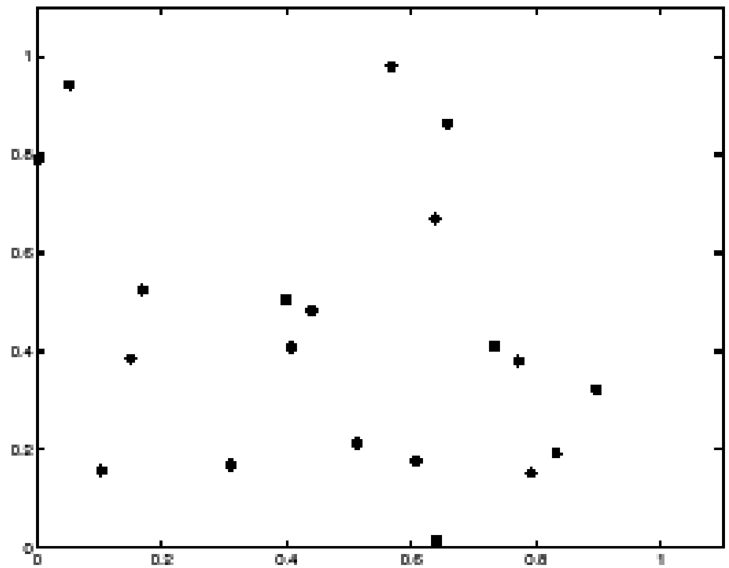


Votes

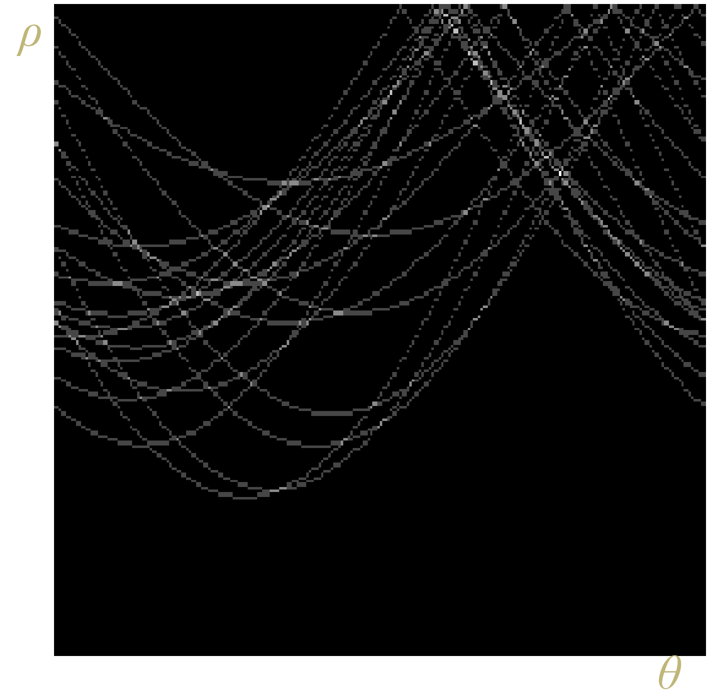
- Problem: Finding the true maximum

Hough Transform

- Noise Input



Tokens

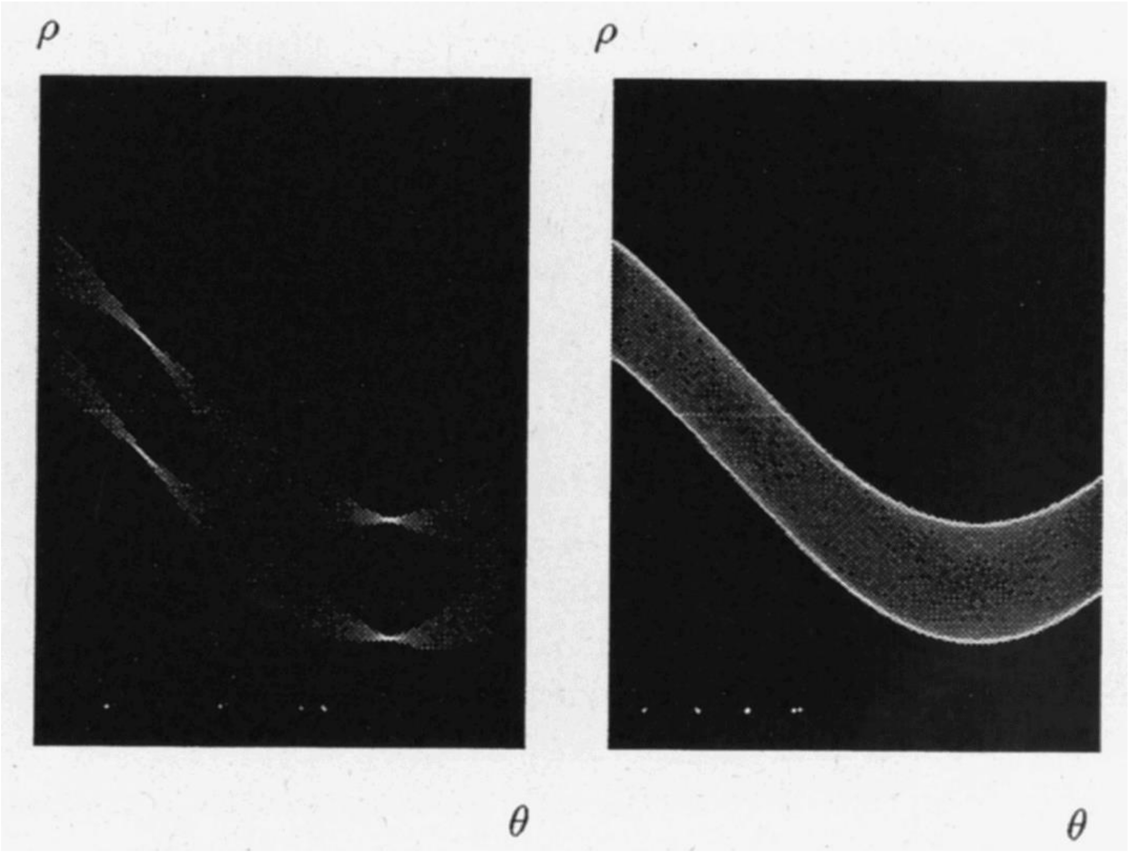


Votes

- Problem: Lots of spurious maxima

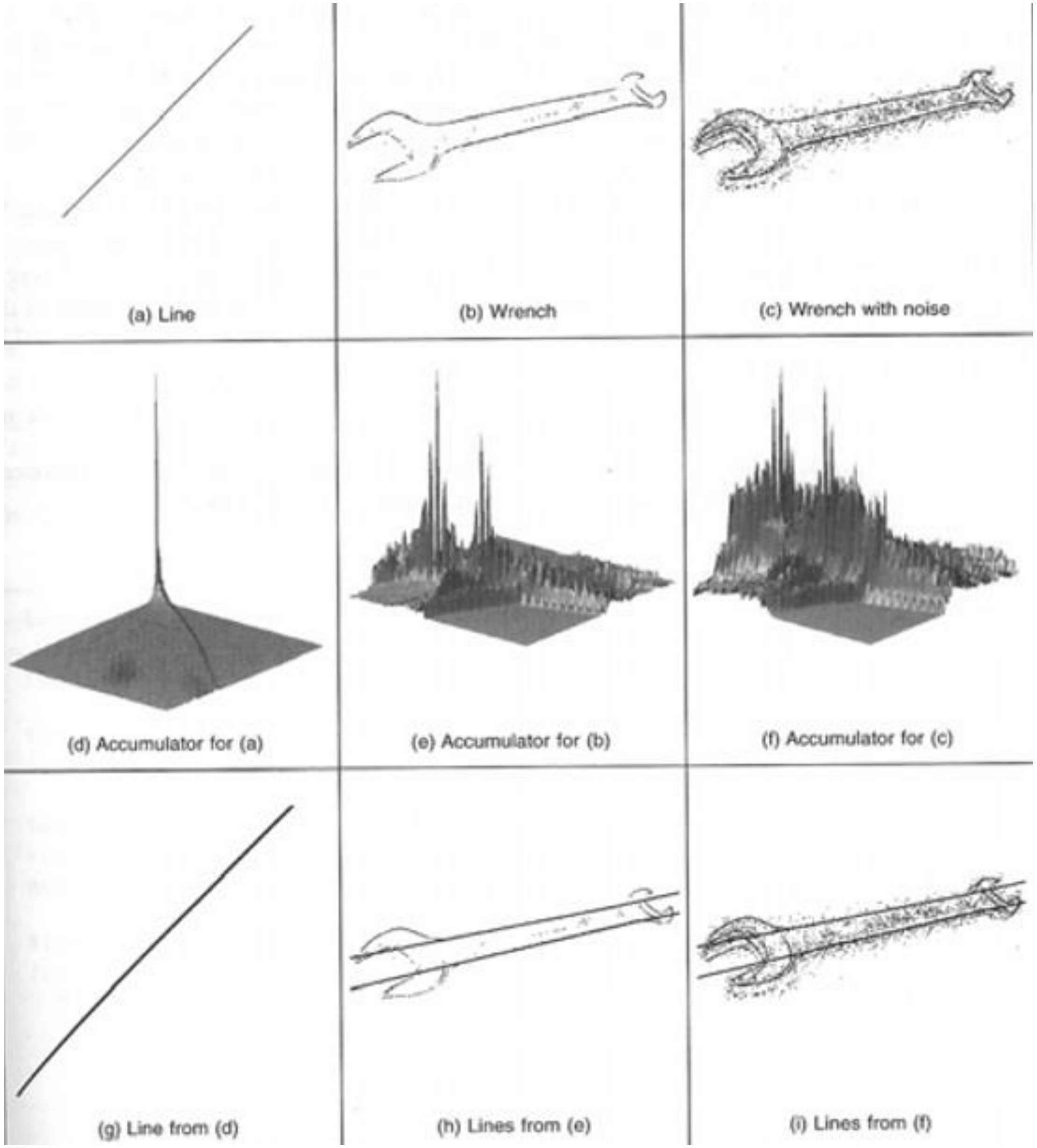
Hough Transform

– A square (left) and a circle (right)



Hough Transform

- Example images



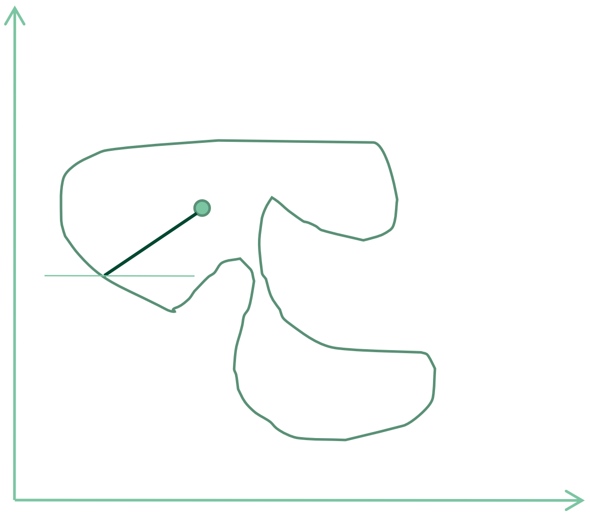
Hough Transform

- Given data points find model parameters to fit to the data
  1. Define model and its parameters
  2. Derive a formula to obtain parameter values given a data point
  3. Define quantization steps and limits of the parameter space
  4. Build accumulative array for the parameter space (Hough space)
  5. Compute parameter values for a data sample and increment the cell in the array that corresponds to these values
  6. Repeat point 5 for every data sample
  7. Perform non maxima suppression in the array
  8. Recover the parameter values that correspond to the maxima.

Generalized Hough Transform

- Finding free form shapes which cannot be parameterised with a small number of parameters
  1. A reference point is chosen inside
  2. A line is constructed joining the reference point and the boundary
  3. The boundary direction is found at the point of intersection of line and boundary
  4. A reference table is constructed

ϕ	(r_1,α_1)		
ϕ_1	(r_1,α_1)	…	(r_1,α_1)
:	:	…	:
ϕ_k	(r_k,α_k)	…	(r_k,α_k)

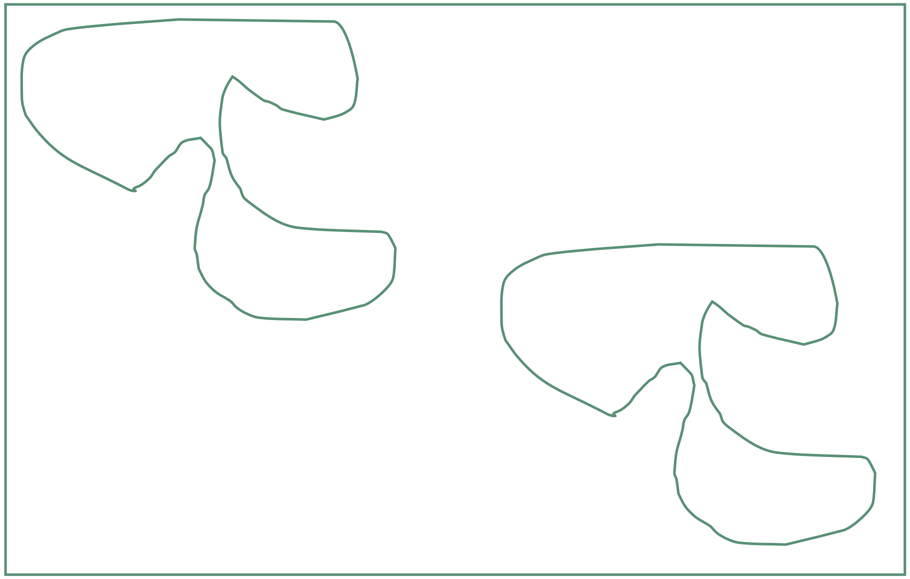


Generalized Hough Transform

- Find location of the shape in the image
  1. Construct 2D accumulative array for x,y location of reference point
  2. Determine edge orientation at every edge point
  3. Given one point use reference table to get corresponding (r,α) values
  4. Calculate position of the reference point

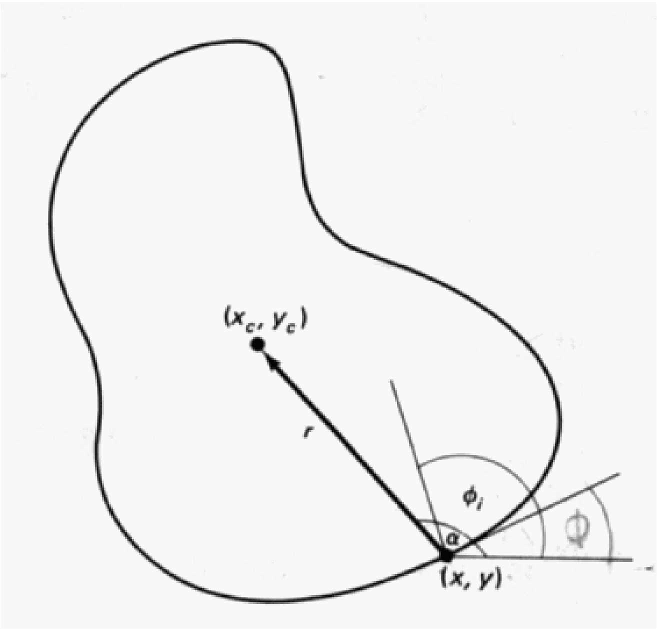
$$x_r = x_\phi + r_\phi \cos(\alpha_\phi) \qquad y_r = y_\phi + r_\phi \sin(\alpha_\phi)$$

ϕ	(r_1,α_1)		
ϕ_1	(r_1,α_1)	...	(r_1,α_1)
:	:	...	:
ϕ_k	(r_k,α_k)	...	(r_k,α_k)



Generalized Hough Transform

- Generalization for an arbitrary contour or shape
  - Choose reference point for the contour (e.g. center)
  - For each point on the contour remember where it is located w.r.t. to the reference point
  - E.g. if the center is the reference point: remember radius r and angle relative to the tangent of the contour
  - Recognition: whenever you find a contour point, calculate the tangent angle and 'vote' for all possible reference points



  - Instead of reference point, can also vote for transformation

⇒ The same idea can be used with local features!

When is the Hough transform useful?

- Textbooks wrongly imply that it is useful mostly for finding lines
  - In fact, it can be very effective for recognizing arbitrary shapes or objects
- The key to efficiency is to have each feature (token) determine as many parameters as possible
  - For example, lines can be detected much more efficiently from small edge elements (or points with local gradients) than from just points
  - For object recognition, each token should predict location, scale, and orientation (4D array)
- Bottom line: The Hough transform can extract feature groupings from clutter in linear time!

Comparison

Gen. Hough Transform

More robust.

- Advantages
  - Can be very effective for recognizing arbitrary patterns
  - Can handle high percentage of outliers (>95%)
  - Extracts groupings from clutter in linear time
- Disadvantages
  - Quantization issues
  - Only practical for small number of dimensions (up to 4)
  - Handles missing and occluded data very gracefully.
  - Can be adapted to many types of forms, not just lines
- Improvements available
  - Probabilistic Extension
  - Continuous Voting Space

  [Leibe04]

RANSAC

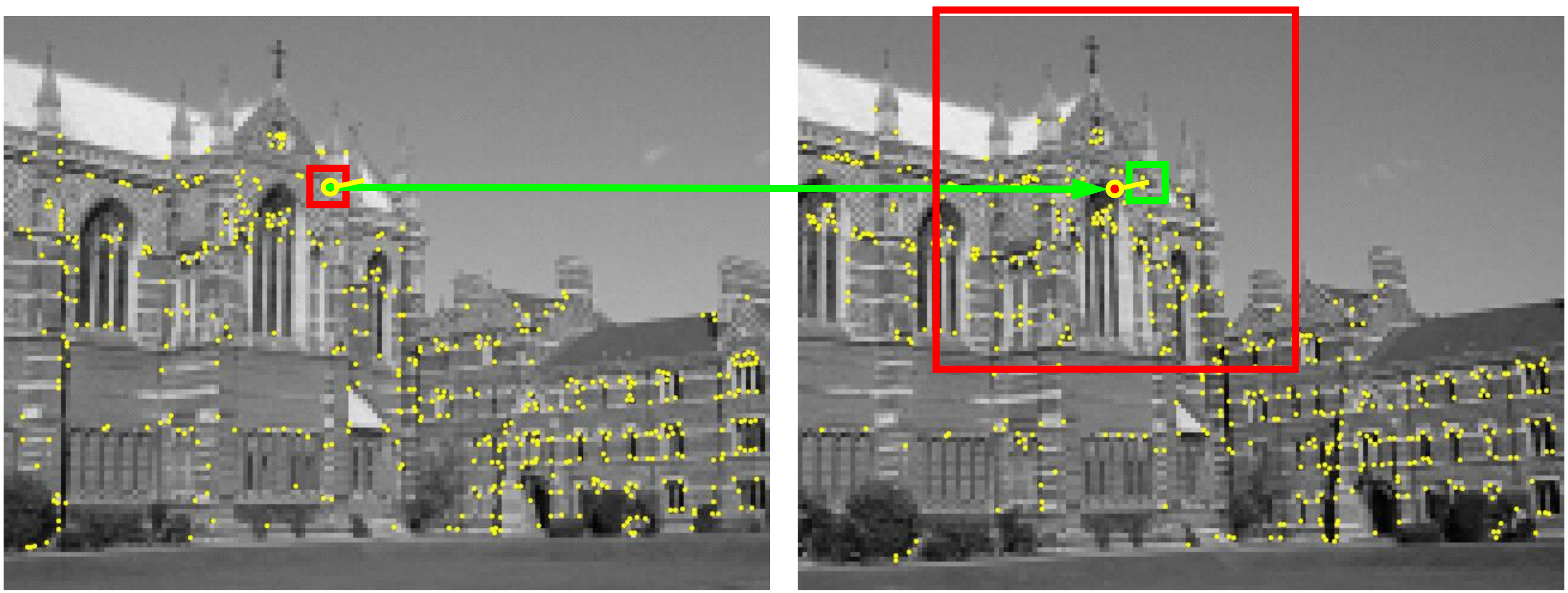
- Advantages
  - General method suited to large range of problems
  - Conceptually simple and easy to implement
  - Independent of number of dimensions
- Disadvantages
  - Only handles moderate number of outliers (<50%)
- Many variants available, e.g.
  - PROSAC: Progressive RANSAC [Chum05]
  - Preemptive RANSAC [Nister05]

Applications

- Sony Aibo
(Evolution Robotics)
  - Recognize
docking station
  - Communicate
with visual cards

Example: Finding Feature Matches

- Find best stereo match within a square search window (here 300 pixels^2)



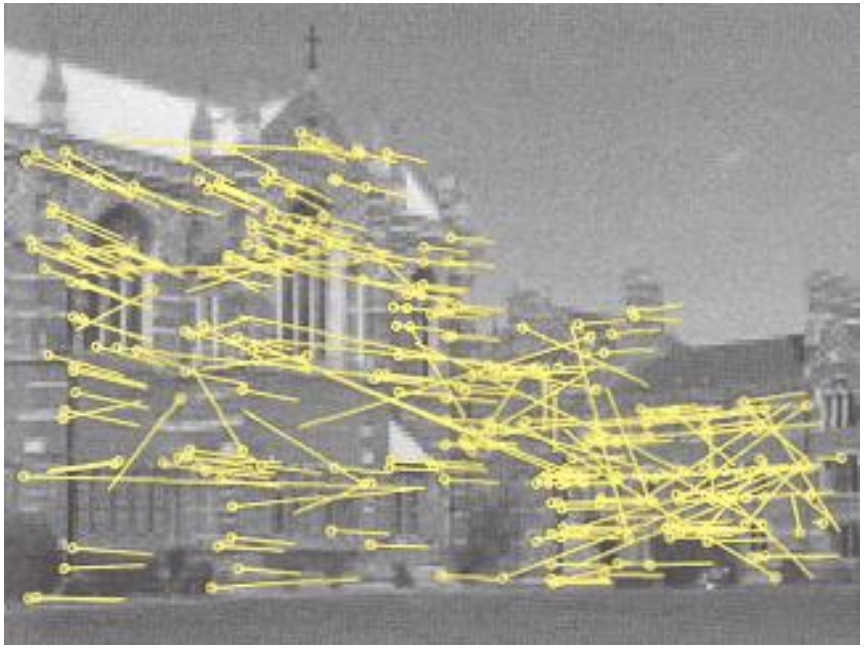
from Hartley & Zisserman

T. Tuytelaars, B. Leibe

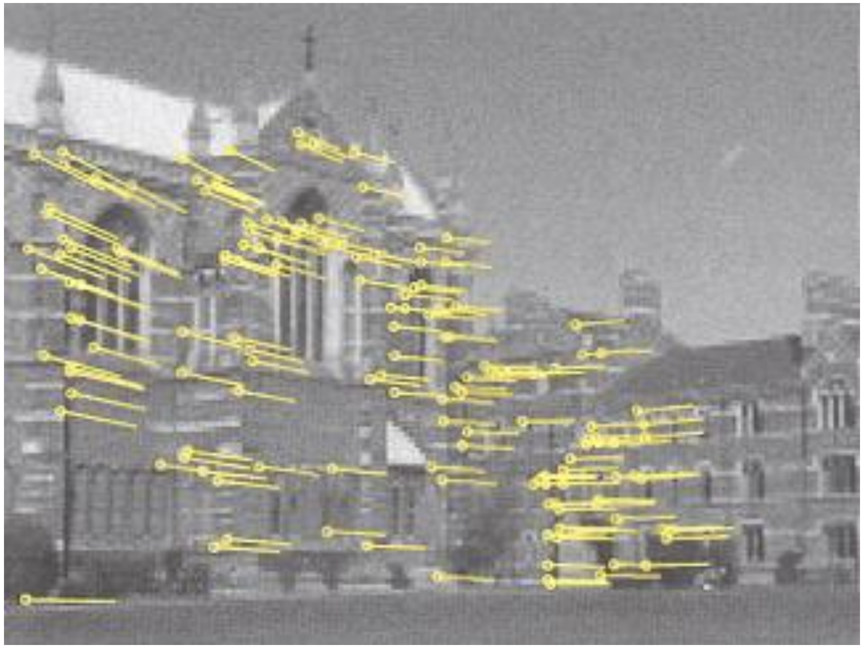
Example: Finding Feature Matches

- Find best stereo match within a square search window (here 300 pixels^2)

before RANSAC



after RANSAC



from Hartley & Zisserman