# Lecture 9

# Fitting and Matching

- Problem formulation
- Least square methods
- RANSAC
- Hough transforms
- Multi-model fitting
- Fitting helps matching!

**Reading:**
[HZ] Chapter: 4 “Estimation – 2D projective transformation”
Chapter: 11 “Computation of the fundamental matrix F”
[FP] Chapter:10 “Grouping and model fitting”

Some slides of this lectures are courtesy of profs. S. Lazebnik & K. Grauman

# Fitting

## Goals:

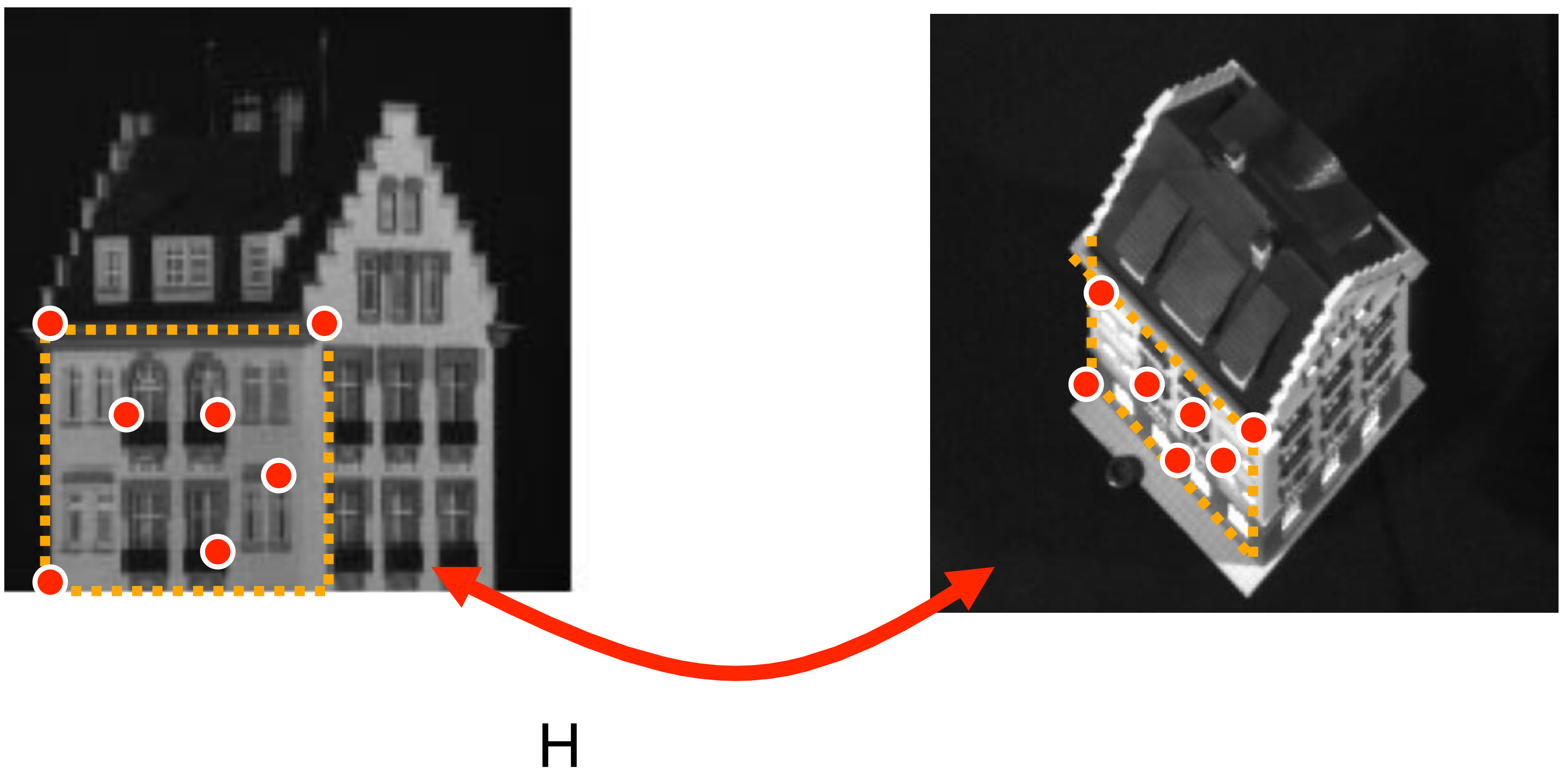
- Choose a parametric model to fit a certain quantity from data
- Estimate model parameters

- Lines
- Curves
- Homographic transformations
- Fundamental matrices
- Shape models

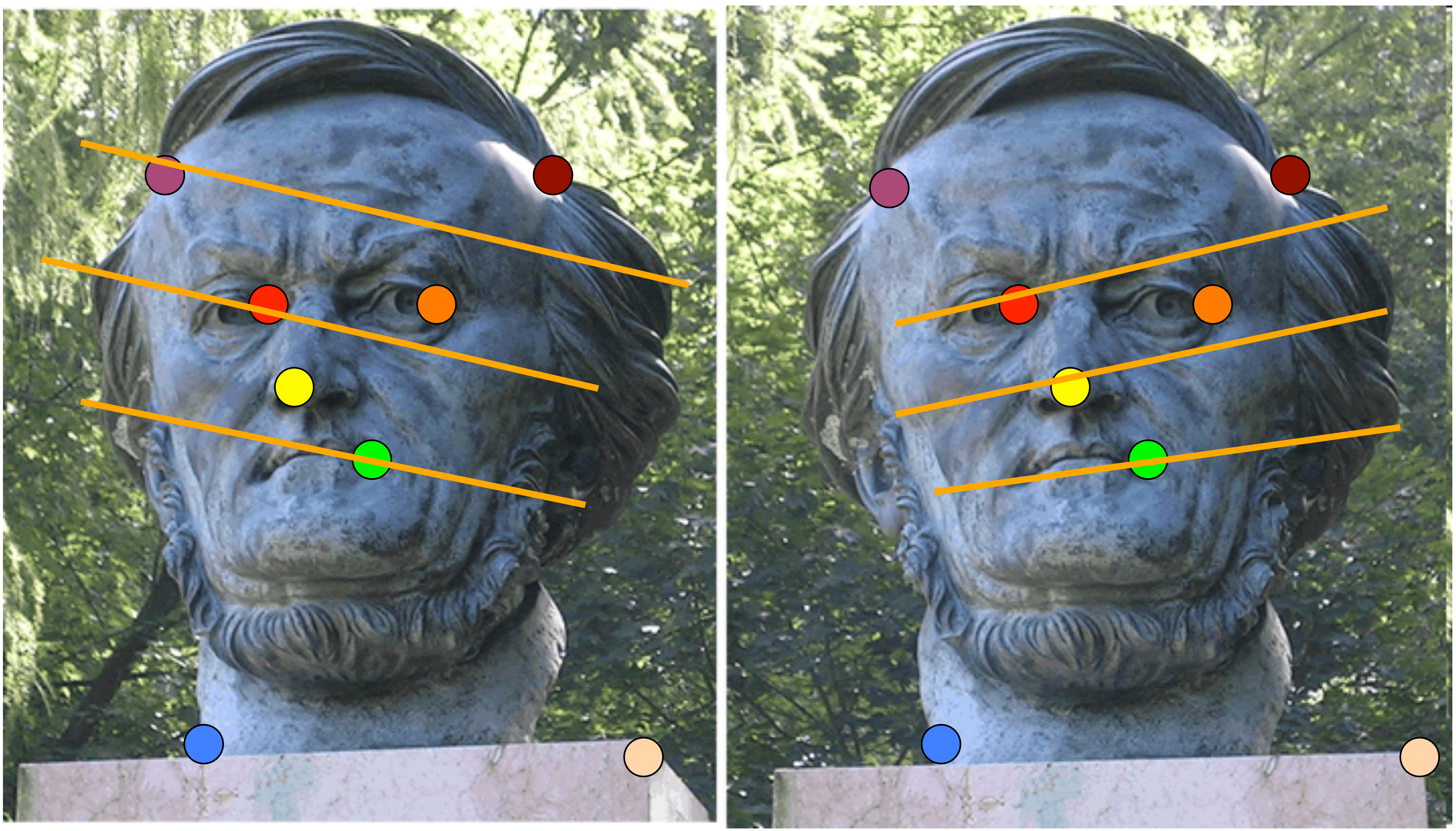
# Example: fitting lines

(for computing vanishing points)

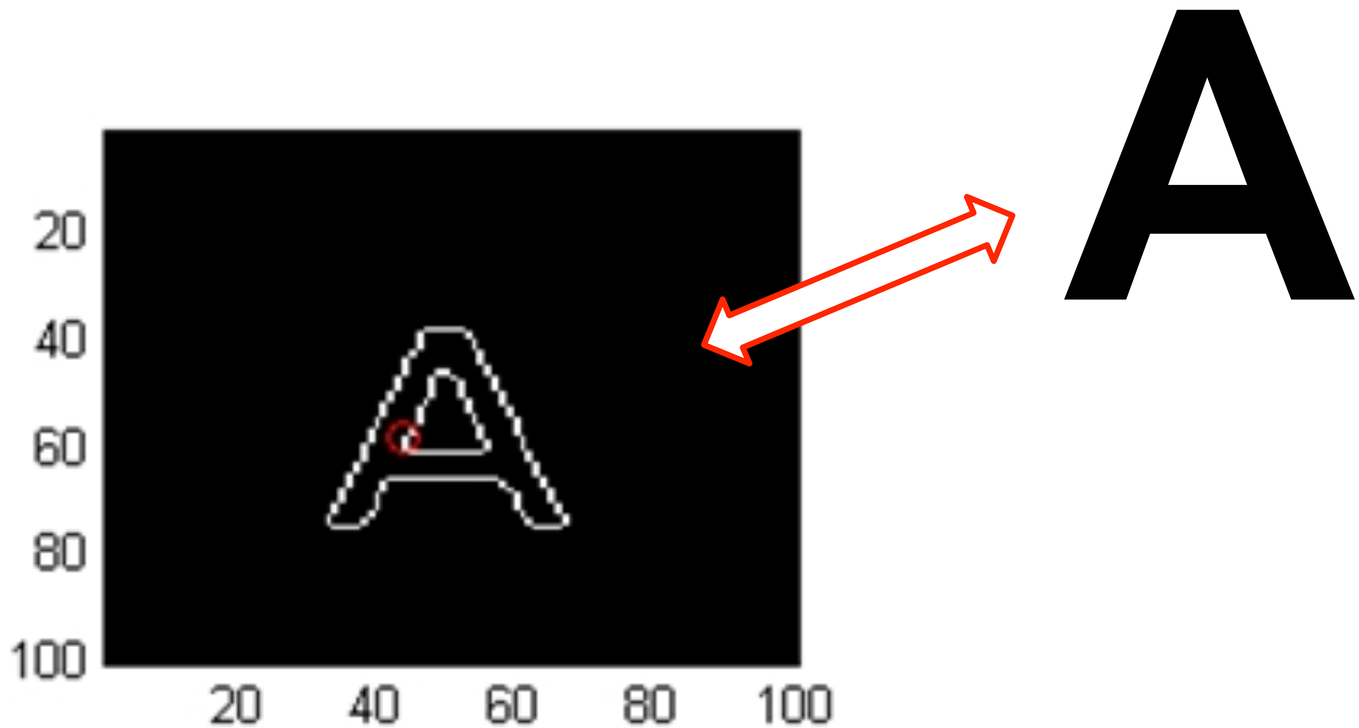
# Example: Estimating an homographic transformation



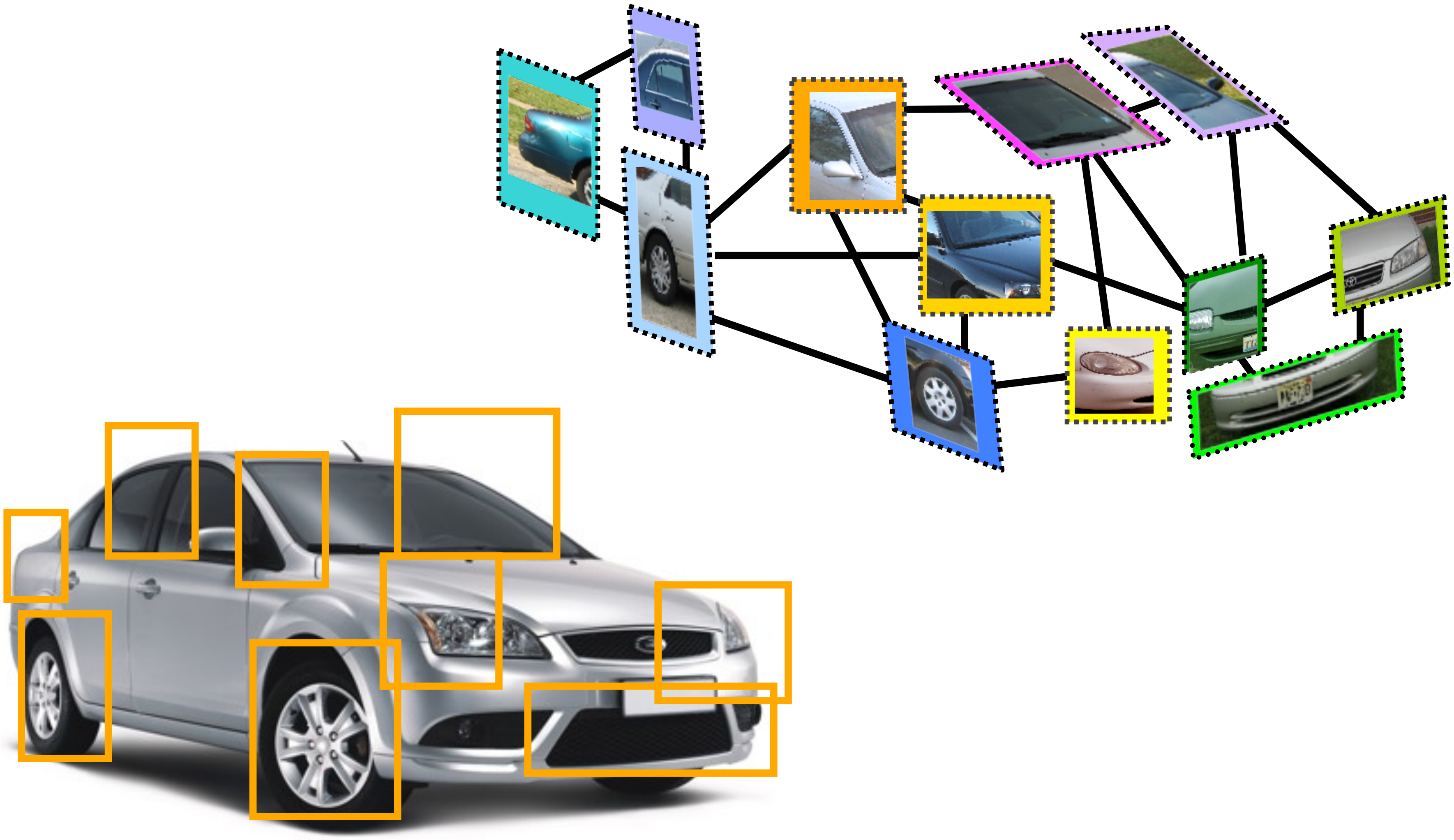
# Example: Estimating F



# Example: fitting a 2D shape template



# Example: fitting a 3D object model



# Fitting, matching and recognition are interconnected problems

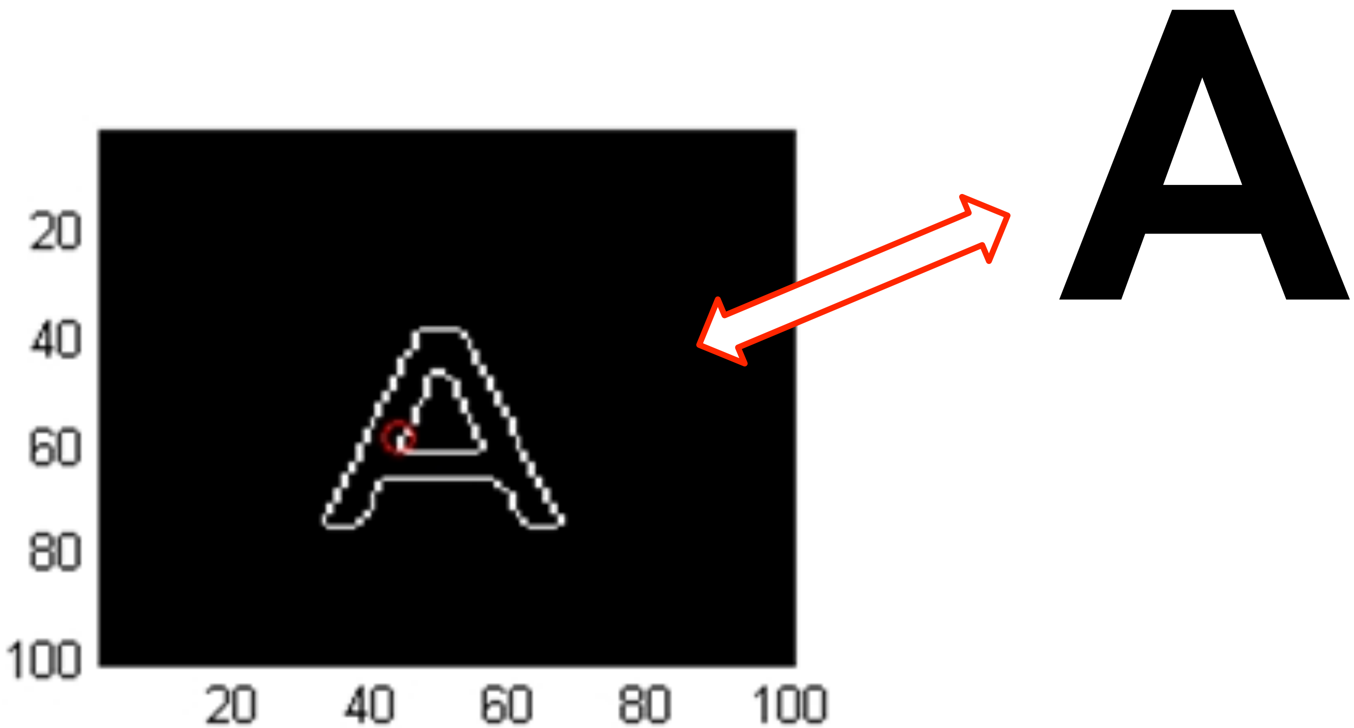
# Fitting

Critical issues:

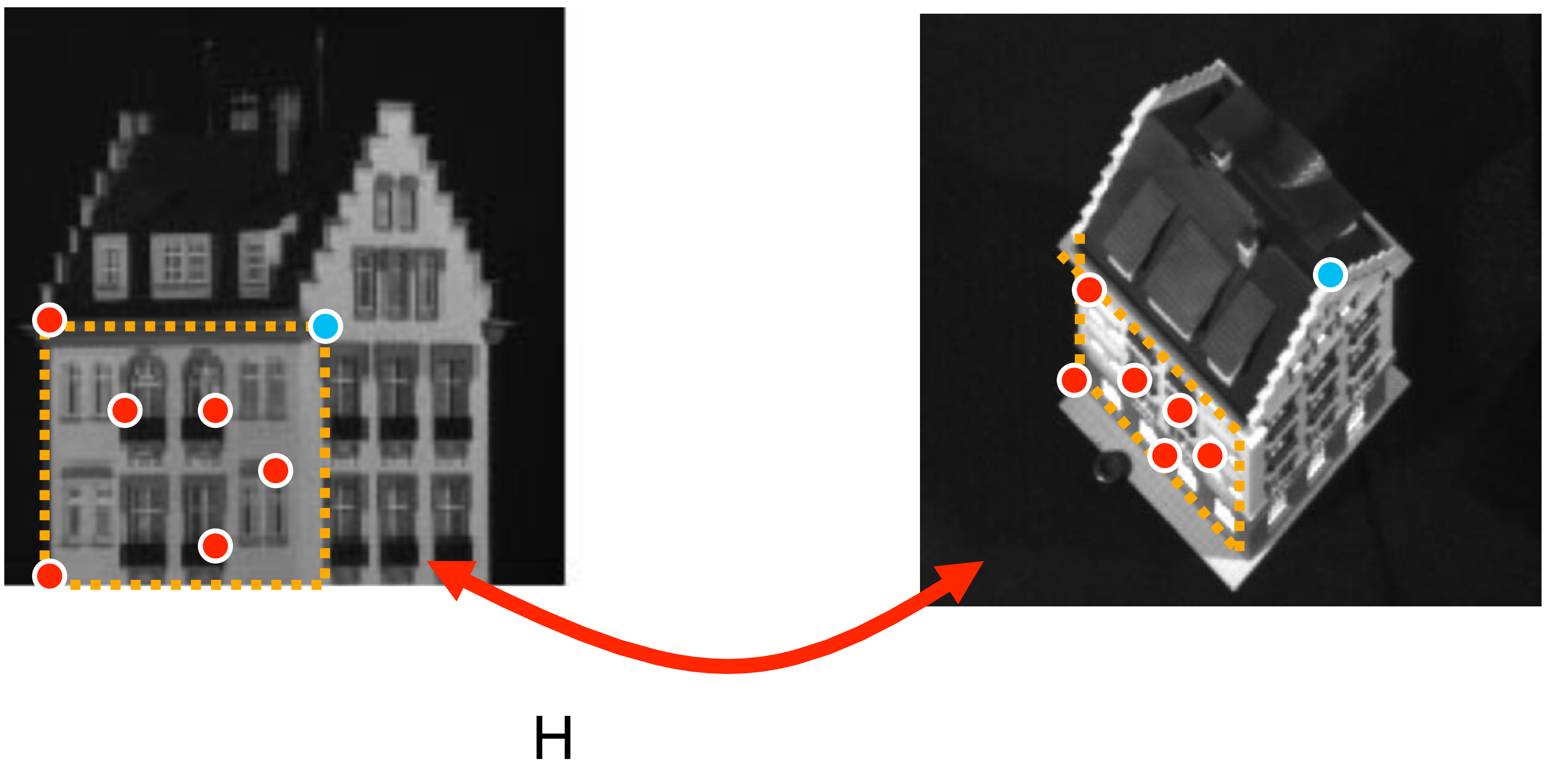
- noisy data
- outliers
- missing data

# Critical issues: noisy data

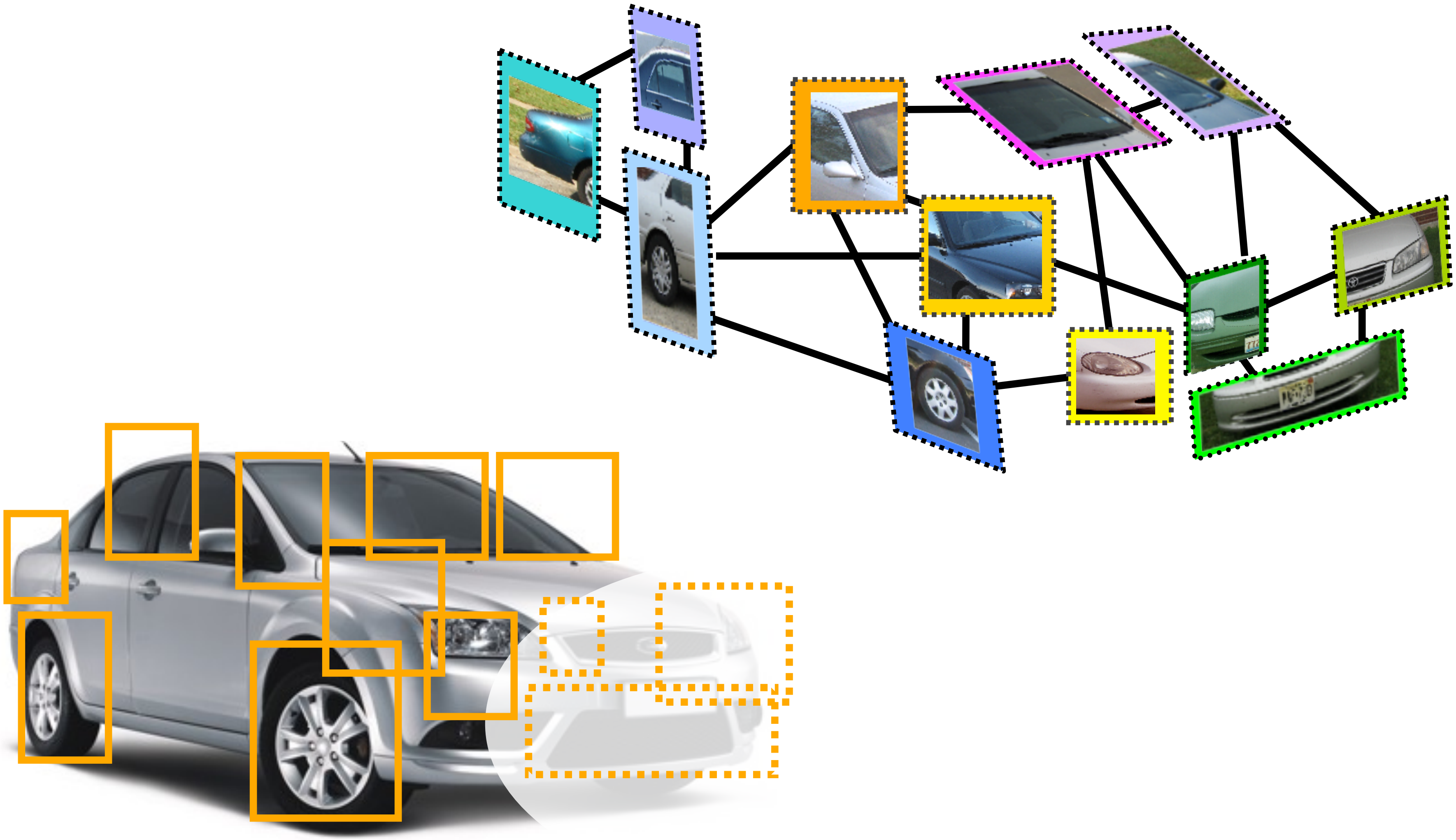
# Critical issues: noisy data (intra-class variability)



# Critical issues: outliers



# Critical issues: missing data (occlusions)



# Fitting

Goal: Choose a parametric model to fit a certain quantity from data

Techniques:

- Least square methods
- RANSAC
- Hough transform
- EM (Expectation Maximization) [not covered]

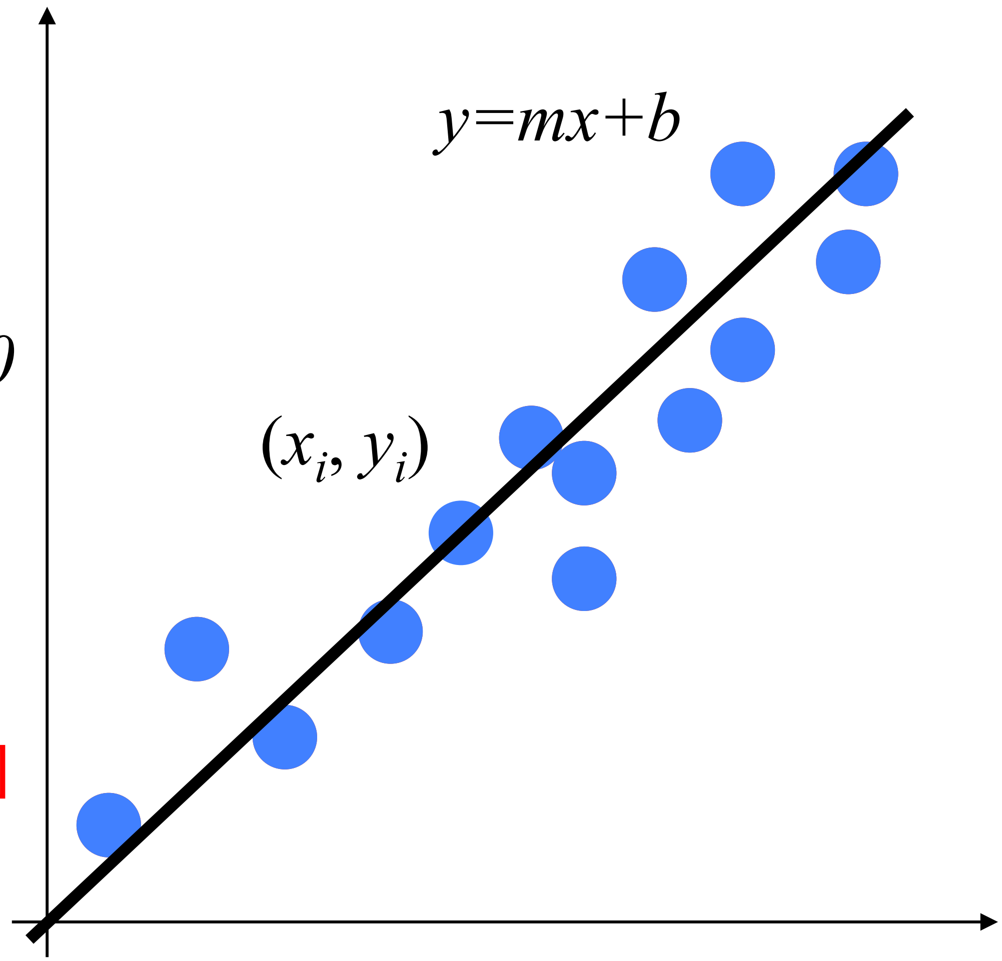
# Least squares methods
## - fitting a line -

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i - m\,x_i - b = 0$

[Eq. 1]

- Find $(m, b)$ to minimize

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



# Least squares methods
## - fitting a line -

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

$$E = \sum_{i=1}^{n} \left( y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right)^2 = \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right\|^2 = \|Y - Xh\|^2$$ [Eq. 3]

$$= (Y - Xh)^T (Y - Xh) = Y^T Y - 2(Xh)^T Y + (Xh)^T (Xh)$$ [Eq. 4]

Find $h = [m, b]^T$ that minimizes E

$$\frac{dE}{dh} = -2X^T Y + 2X^T X h = 0$$ [Eq. 5]

$$X^T X h = X^T Y$$ [Eq. 7]

*Normal equation*

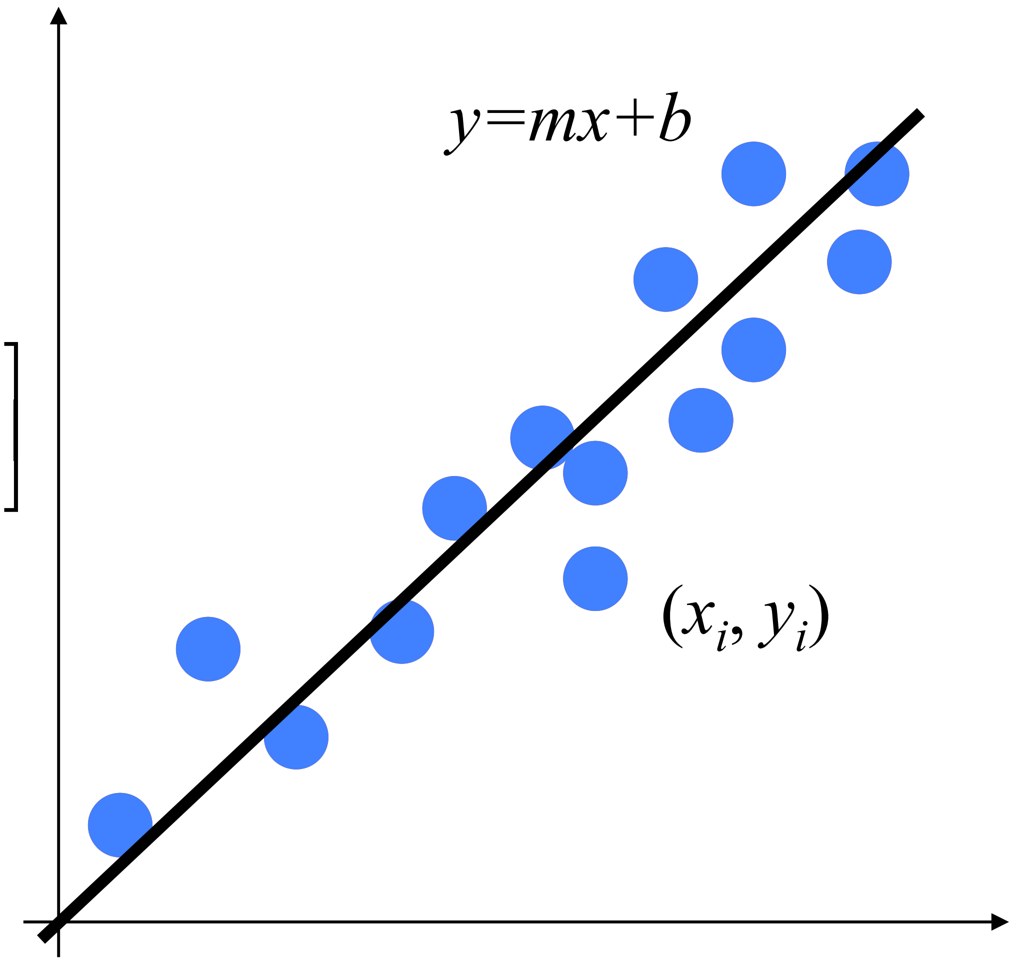
$$\boxed{h = \left( X^T X \right)^{-1} X^T Y}$$ [Eq. 6]

# Least squares methods
## - fitting a line -

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

$$\boxed{h = \left(X^T X\right)^{-1} X^T Y} \qquad h = \begin{bmatrix} m \\ b \end{bmatrix}$$

[Eq. 6]




## Limitations

- Fails completely for vertical lines

# Least squares methods

## - fitting a line -

- Distance between point $(x_n, y_n)$ and line $ax+by=d$

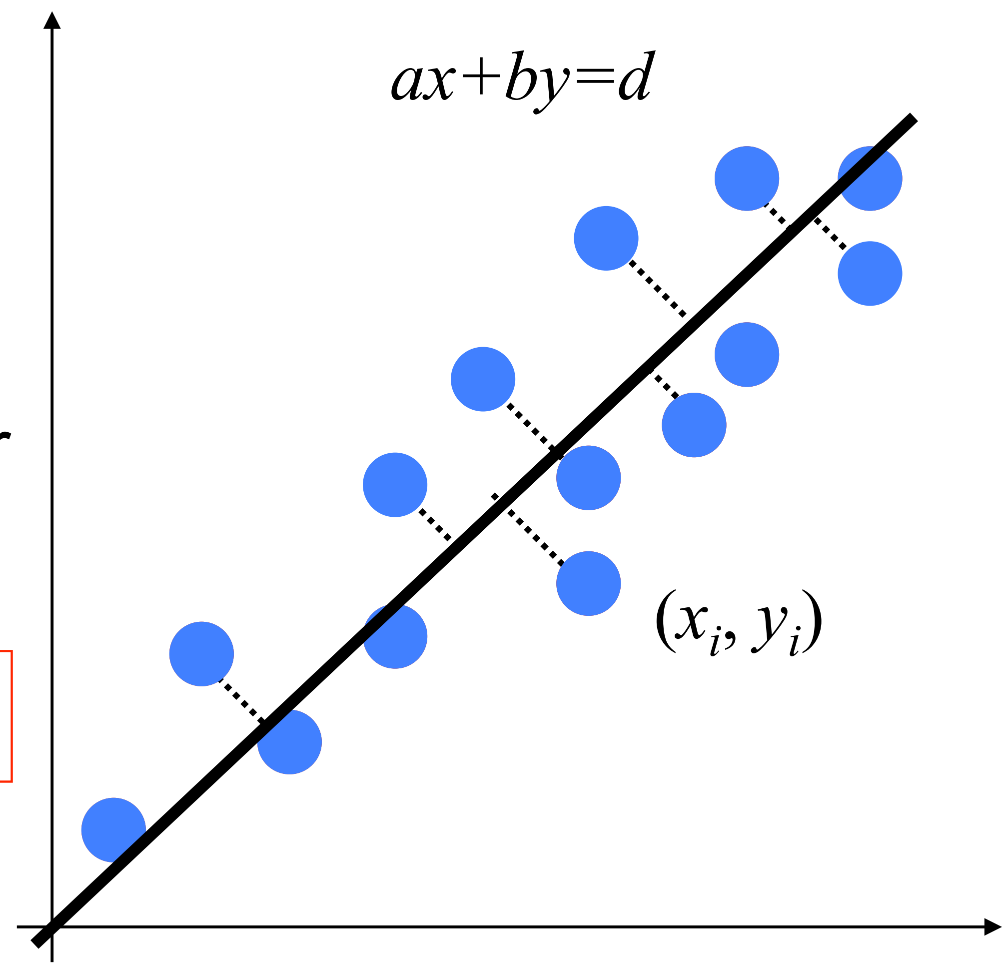
- Find $(a, b, d)$ to minimize the sum of squared perpendicular distances

[Eq. 8]

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

$$\underbrace{A}_{\text{data}}\;\underbrace{h}_{\text{model parameters}} = 0$$

[Eq. 9]



# Least squares methods

## - fitting a line -

$$A\,h = 0$$ A is rank deficient

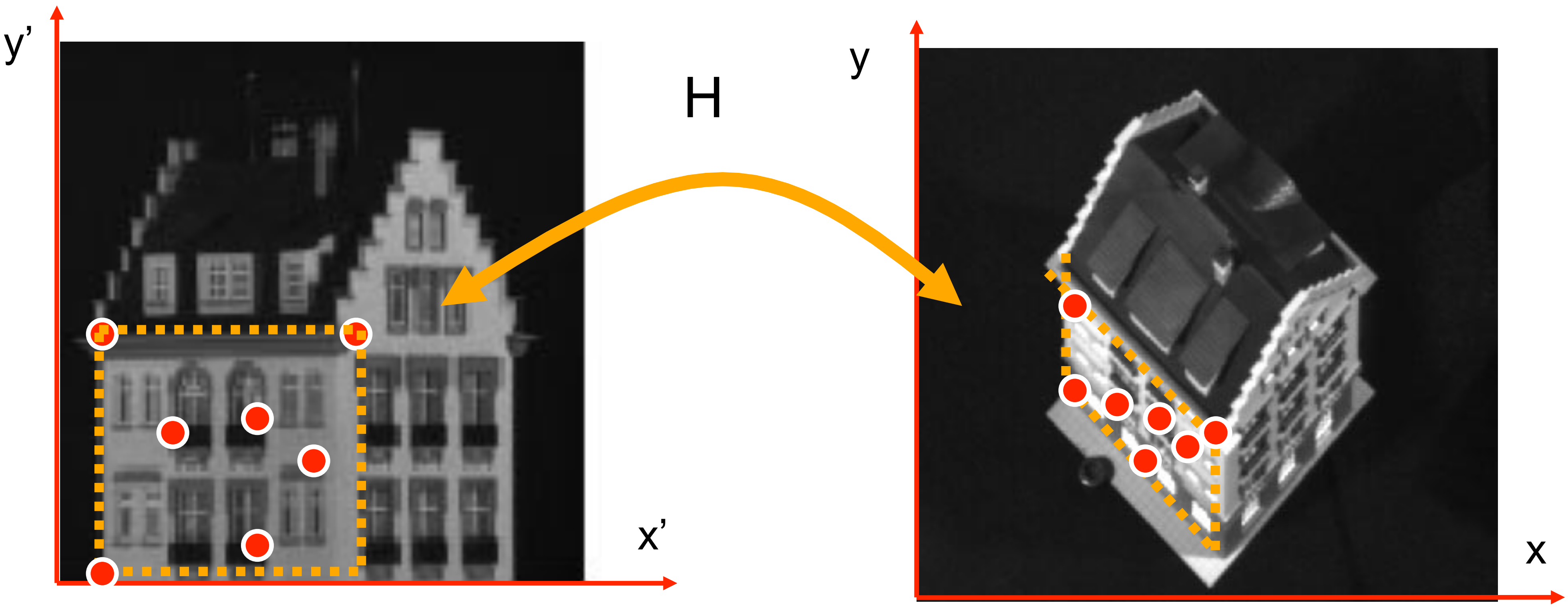
Minimize $\| A\,h \|$ subject to $\| h \| = 1$

$$A = UDV^T$$

h = last column of V

# Least squares methods

## - fitting an homography -

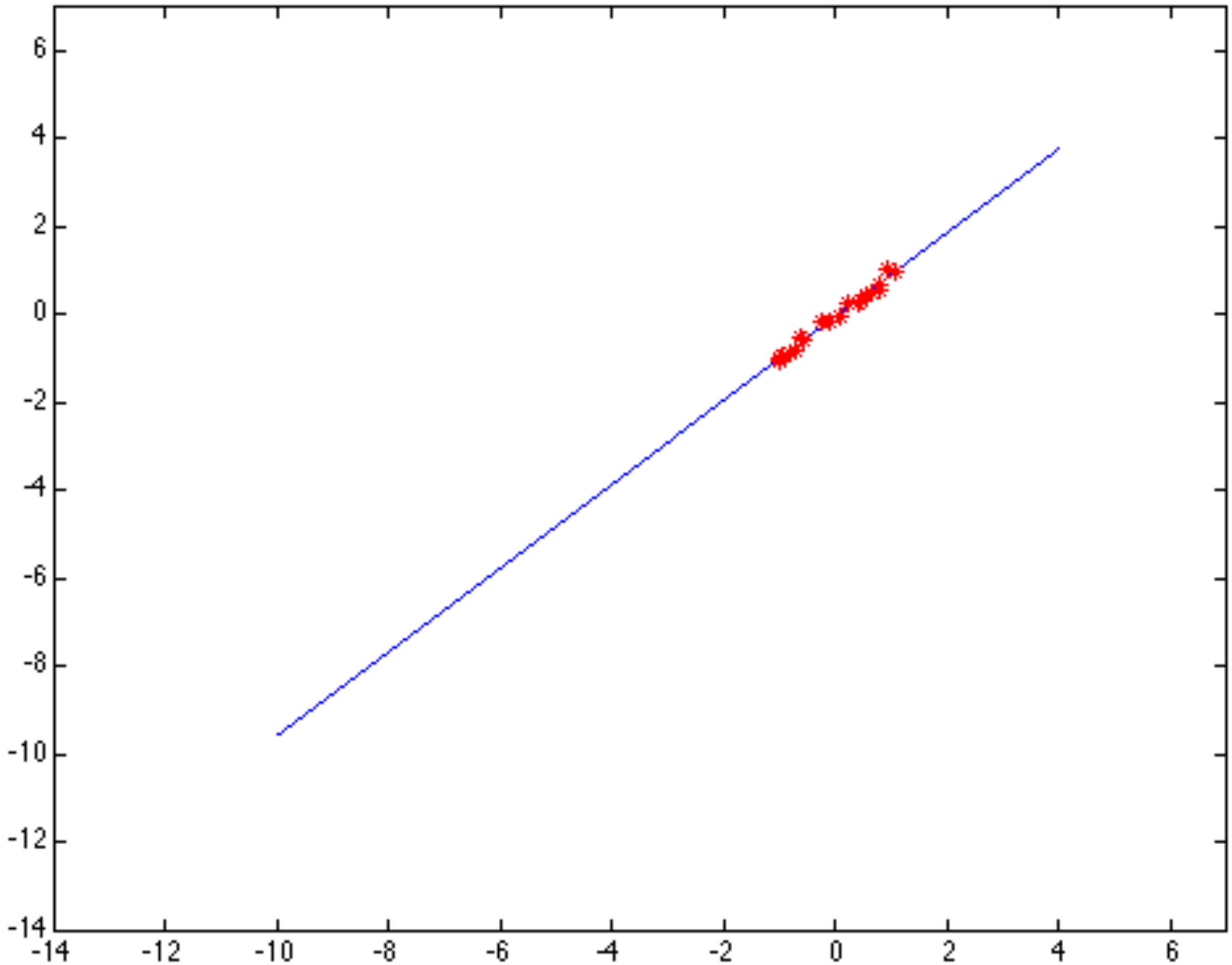



$$\boxed{A}\,\boxed{h} = 0 \qquad \text{[Eq. 10]}$$

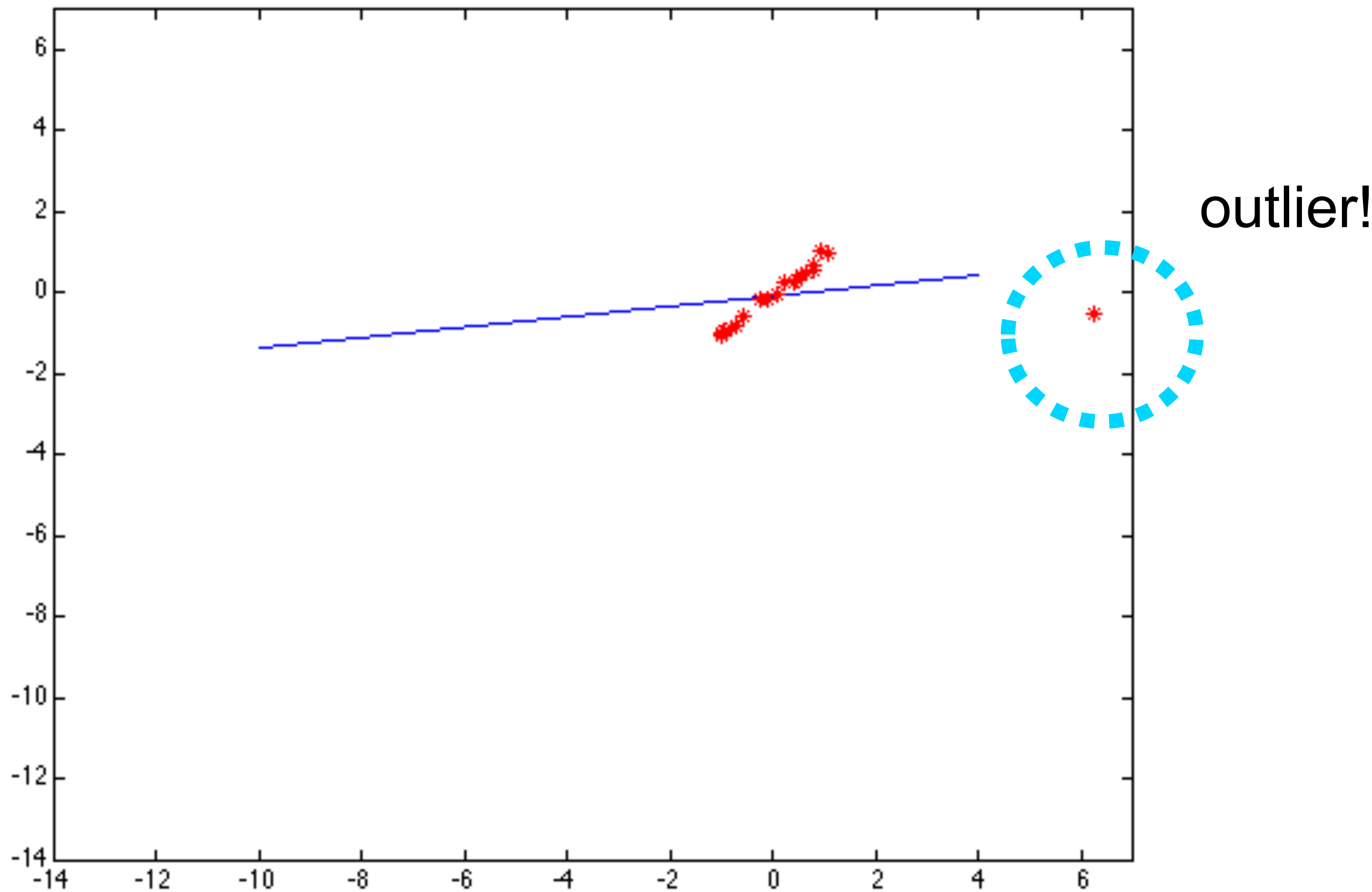
data model parameters

See HZ
- Sec 4.1 for details (DLT algorithm)
- Sec 4.1.2 (or APPENDIX)

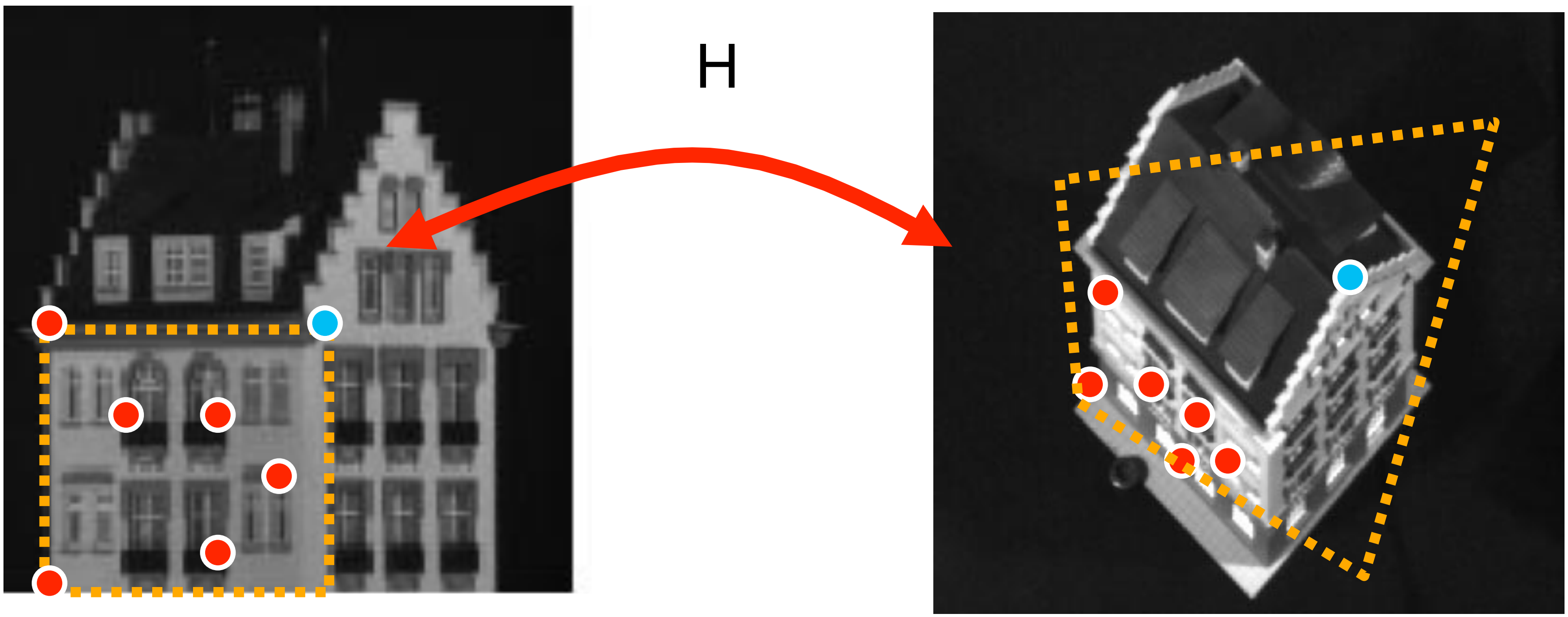
# Least squares: Robustness to noise



# Least squares: Robustness to noise



# Critical issues: outliers




CONCLUSION: Least square is not robust w.r.t. outliers

# Least squares: Robust estimators

Instead of minimizing $E = \sum_{i=1}^{n} (ax_i + by_i - d)^2$ [Eq. 8]

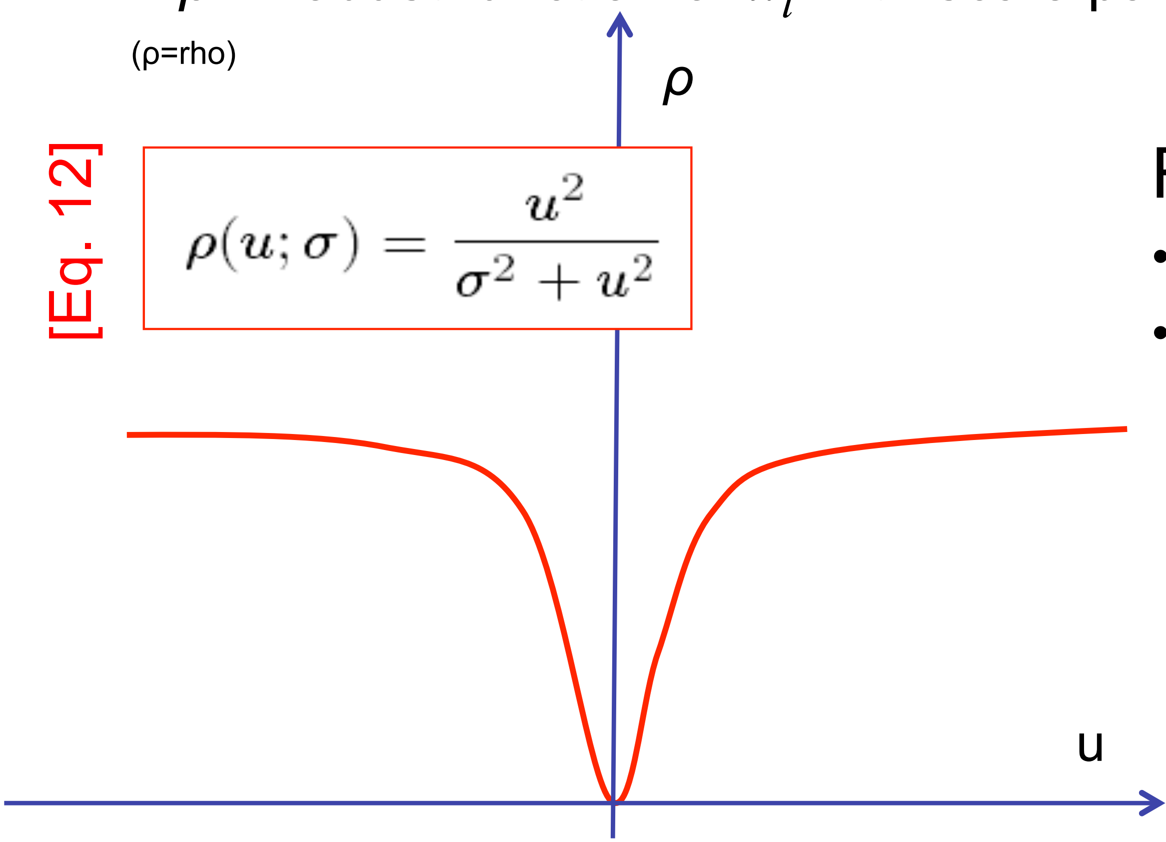
We minimize $\boxed{E = \sum_i \rho(u_i ; \sigma)}$ [Eq. 11] $\quad u_i = ax_i + by_i - d$

- $u_i$ = error (residual) of $i^{th}$ point w.r.t. model parameters $h$ = (a,b,d)
- $\rho$ = robust function of $u_i$ with scale parameter σ

(ρ=rho)

[Eq. 12]

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




Robust function $\rho$:

- When u is large, ρ saturates to 1
- When u is small, ρ is a function of $u^2$

In conclusion:

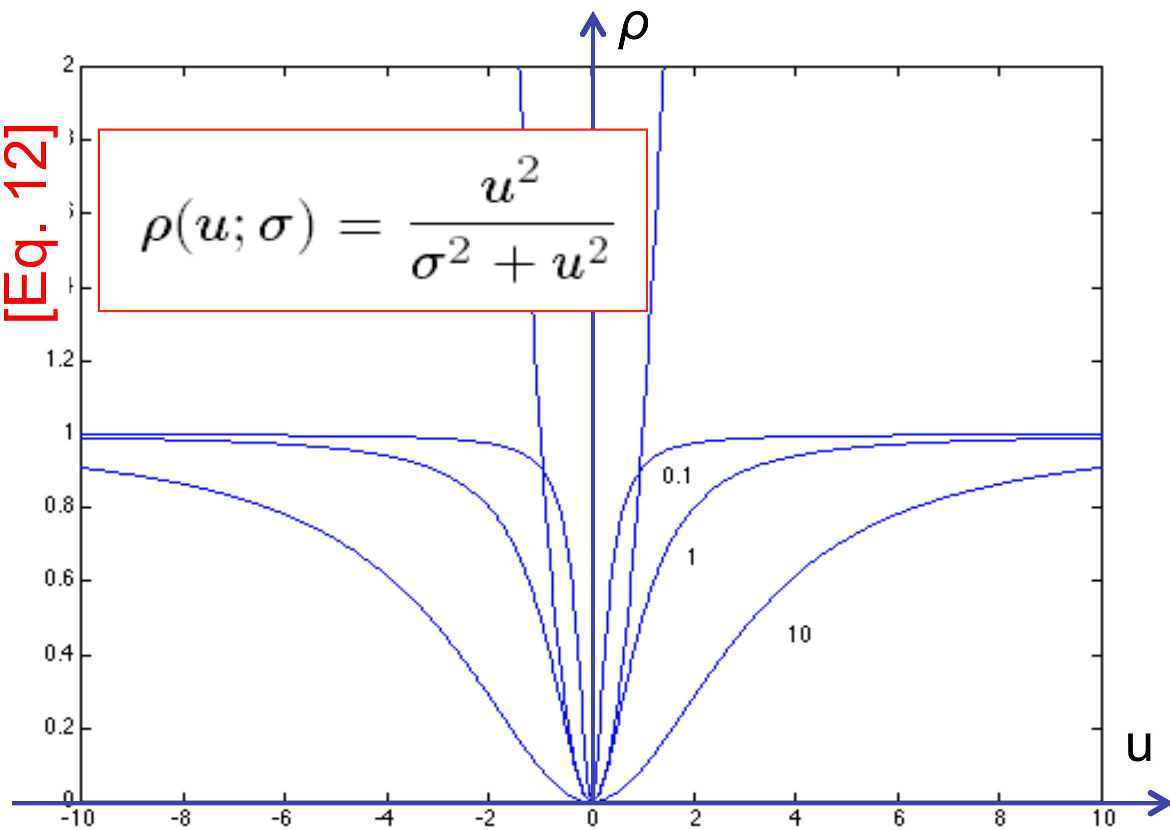
- Favors a configuration with small residuals
- Penalizes large residuals

# Least squares: Robust estimators

Instead of minimizing $E = \sum_{i=1}^{n} (ax_i + by_i - d)^2$ [Eq. 8]

We minimize $\boxed{E = \sum_i \rho(u_i ; \sigma)}$ [Eq. 11] $u_i = ax_i + by_i - d$

- $u_i$ = error (residual) of i^th point w.r.t. model parameters $h$ = (a,b,d)
- $\rho$ = robust function of $u_i$ with scale parameter σ

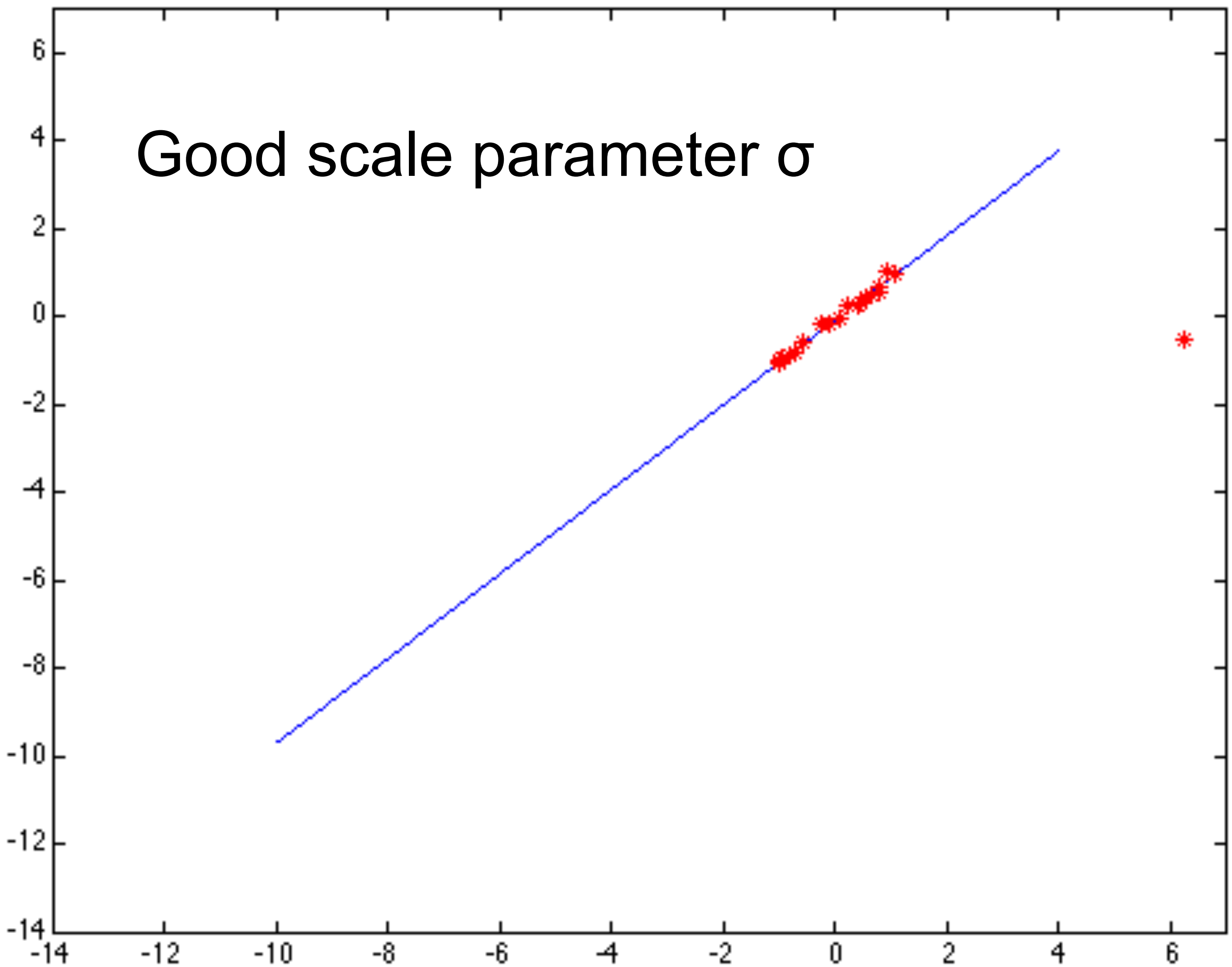



[Eq. 12] $\rho(u;\sigma) = \dfrac{u^2}{\sigma^2 + u^2}$

•Small sigma → highly penalize large residuals

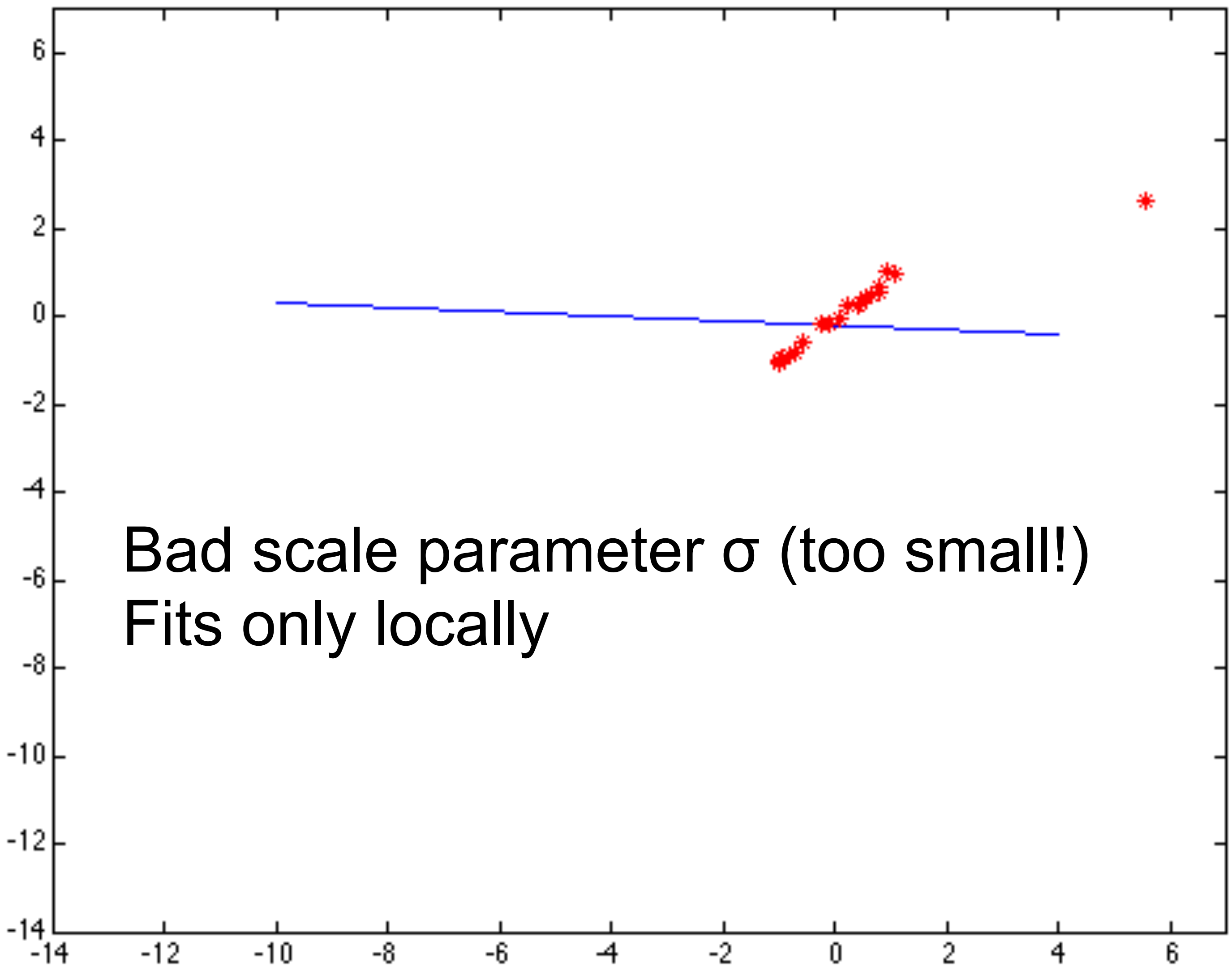
•Large sigma → mildly penalize large residual (like LSQR)

# Least squares: Robust estimators

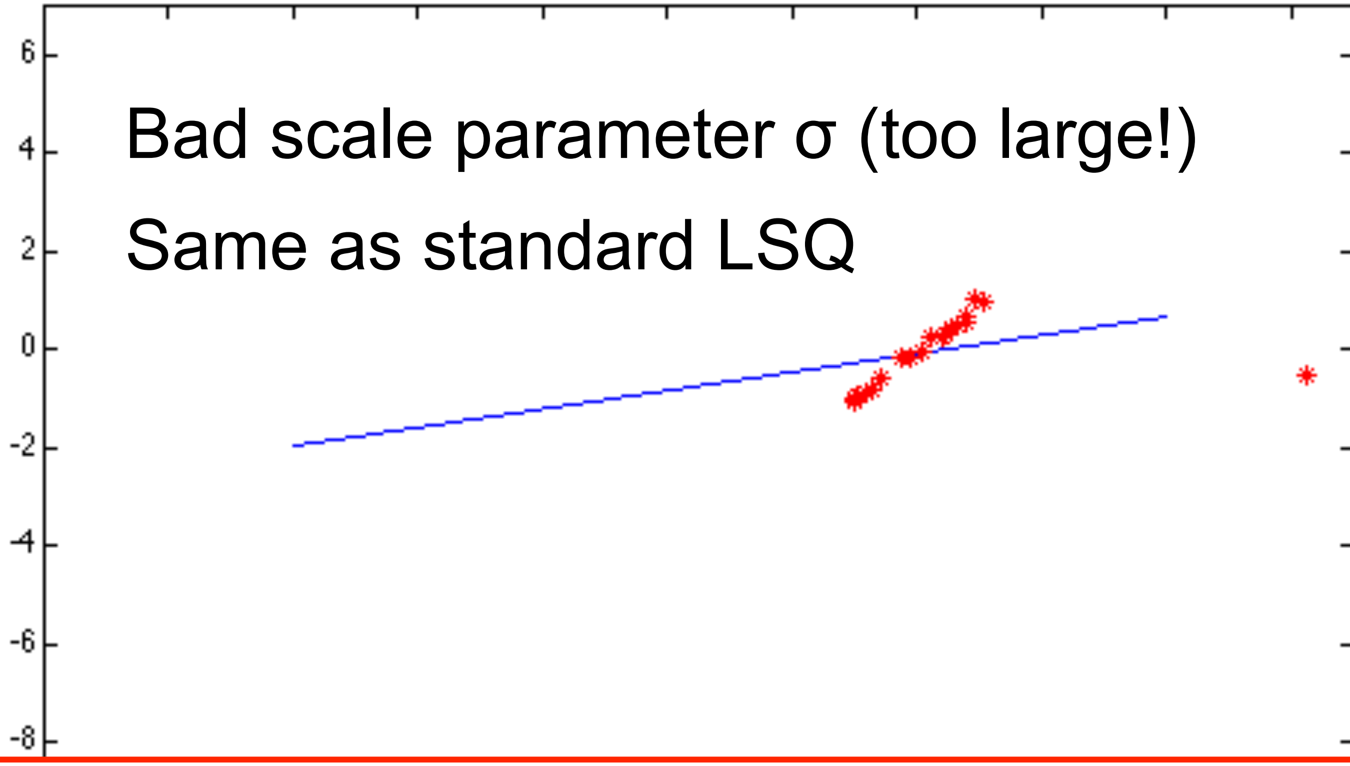



The effect of the outlier is eliminated

# Least squares: Robust estimators



# Least squares: Robust estimators




•CONCLUSION: Robust estimator useful if prior info about the distribution of points is known

•Robust fitting is a nonlinear optimization problem (iterative solution)

•Least squares solution provides good initial condition

# Fitting

Goal: Choose a parametric model to fit a certain quantity from data

## Techniques:

- Least square methods
- RANSAC
- Hough transform

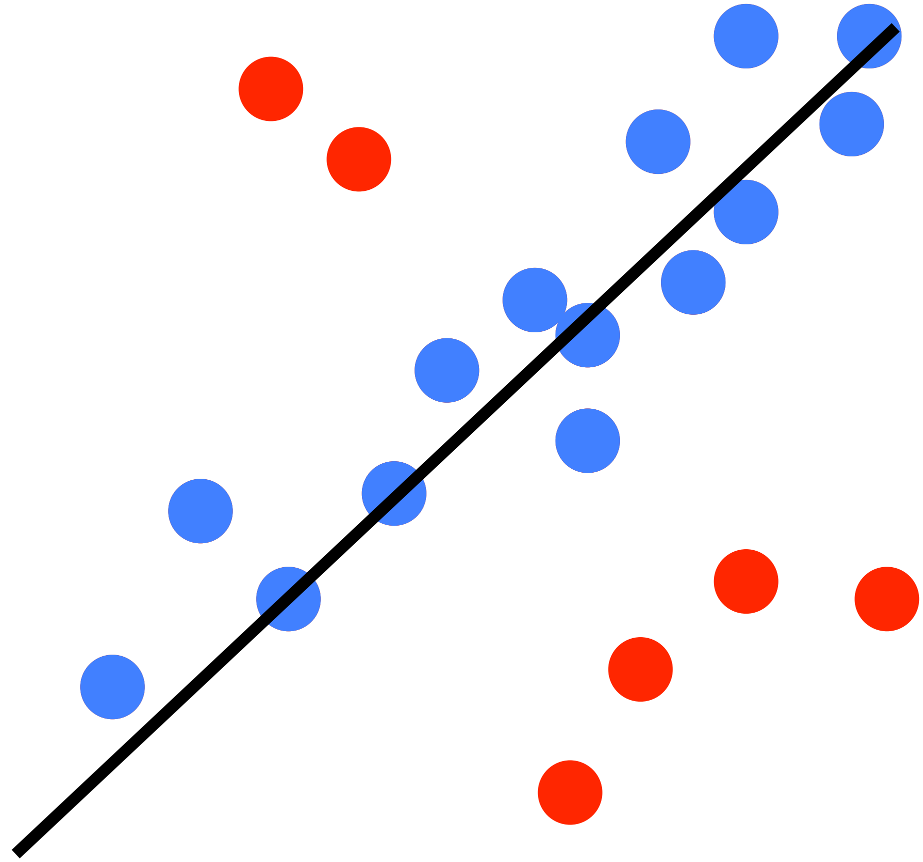
# Basic philosophy
## (voting scheme)

- Data elements are used to vote for one (or multiple) models

- Robust to outliers and missing data

- Assumption 1: Noisy data points will not vote consistently for any single model ("few" outliers)
- Assumption 2: There are enough data points to agree on a good model ("few" missing data)

# Example: Line fitting

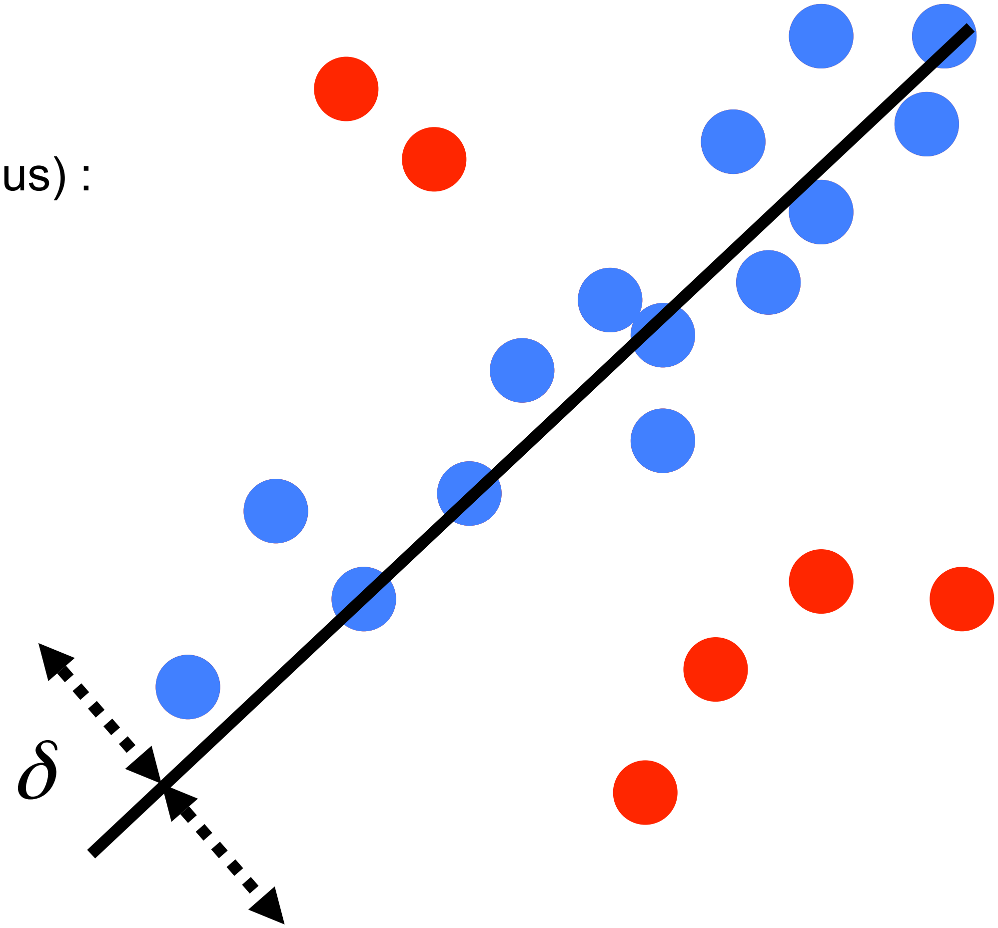


- Enough “good” data points supporting the line model in presence of noise
- “Few” outliers compared to the “good” data points – these few outliers won’t “consistently” for the line model

# RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.




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

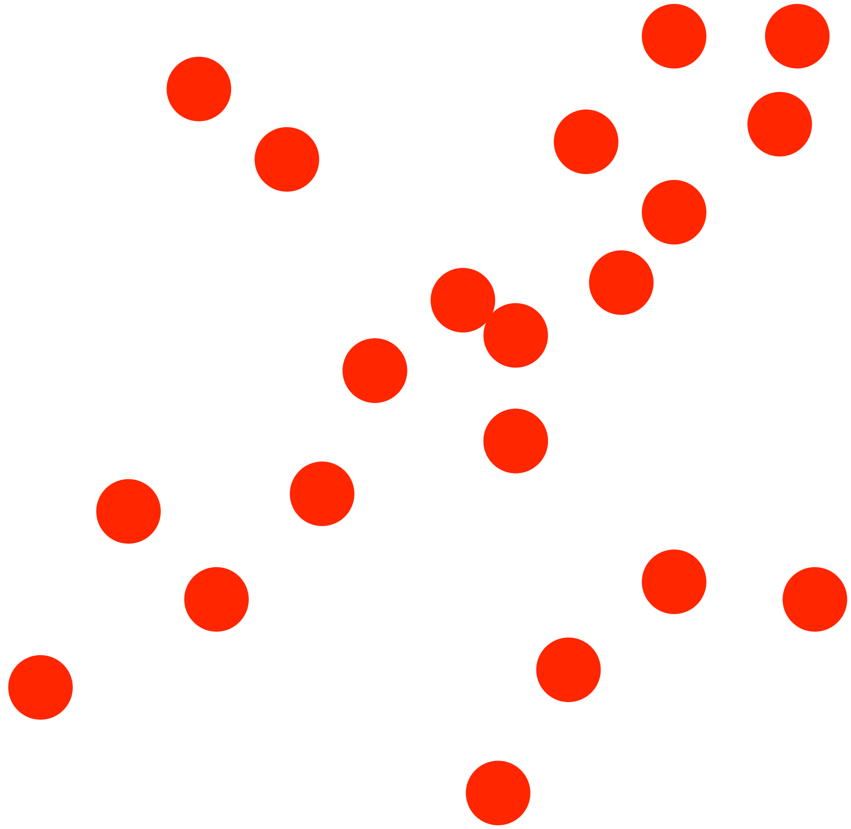
such that:

Model parameters

$$r(P, h) < \delta, \quad \forall P \in \mathbf{P} \qquad r(P, h) = residual$$

[Eq. 12]

# RANSAC



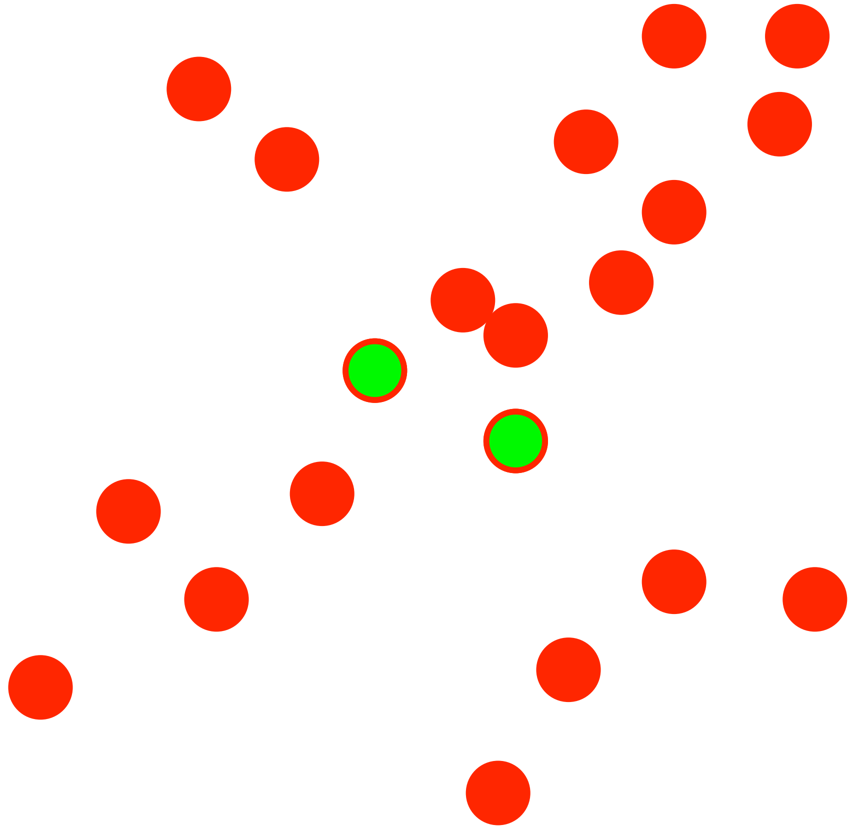
Sample set = set of points in 2D

## Algorithm:

1. Select random sample of minimum required size to fit model
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

# RANSAC



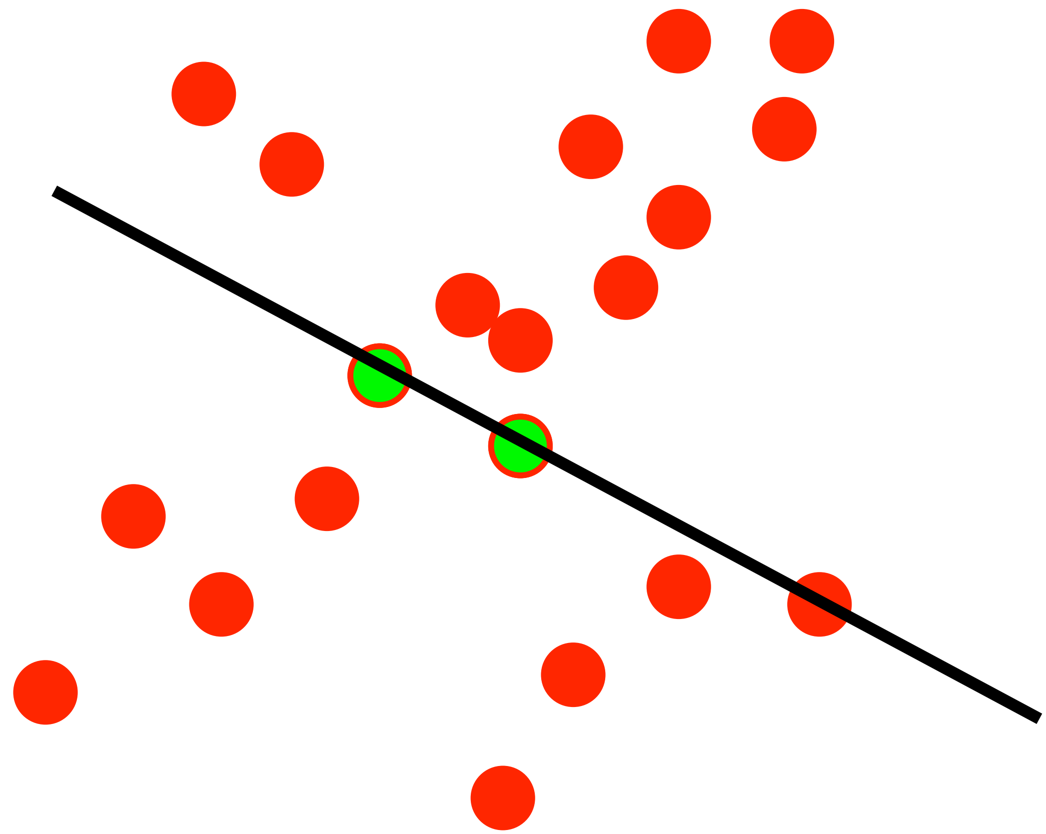
Sample set = set of points in 2D

## Algorithm:

1. Select random sample of minimum required size to fit model [?]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

# RANSAC



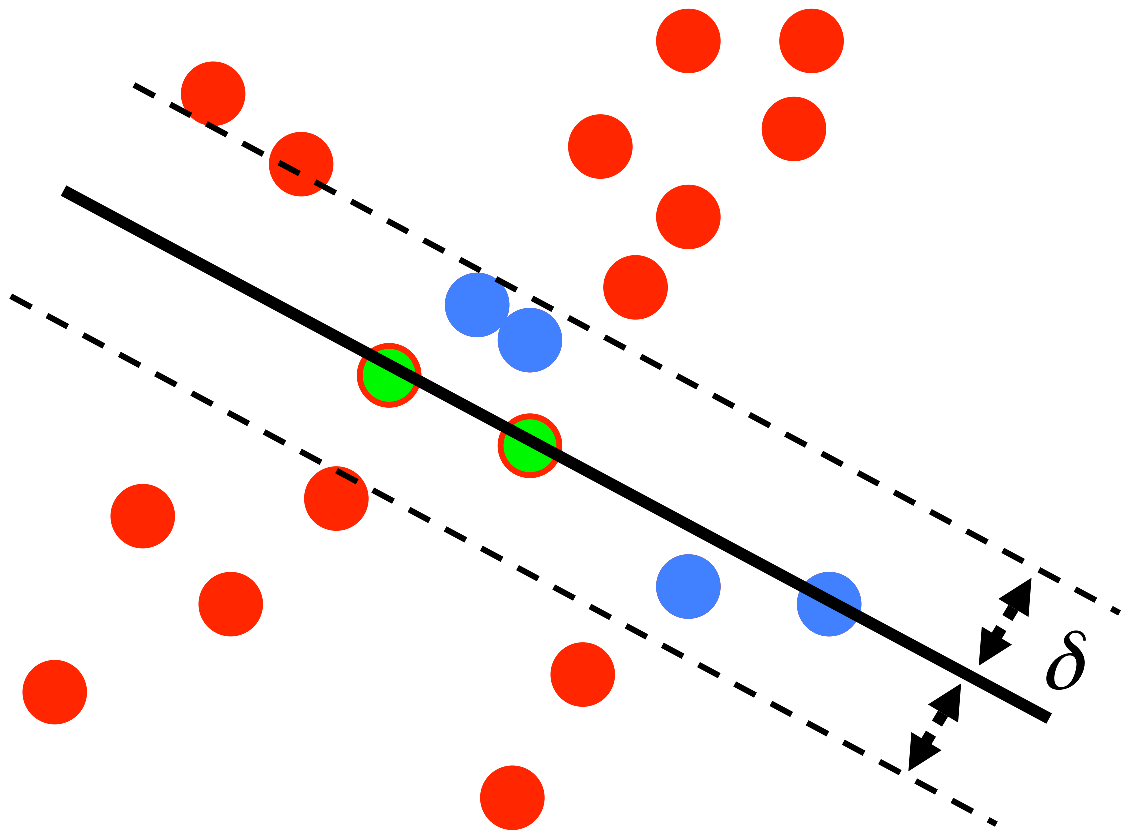
Sample set = set of points in 2D

## Algorithm:

1. Select random sample of minimum required size to fit model [?]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

# RANSAC




Sample set = set of points in 2D

$$|\mathbf{O}| = ? = 14$$

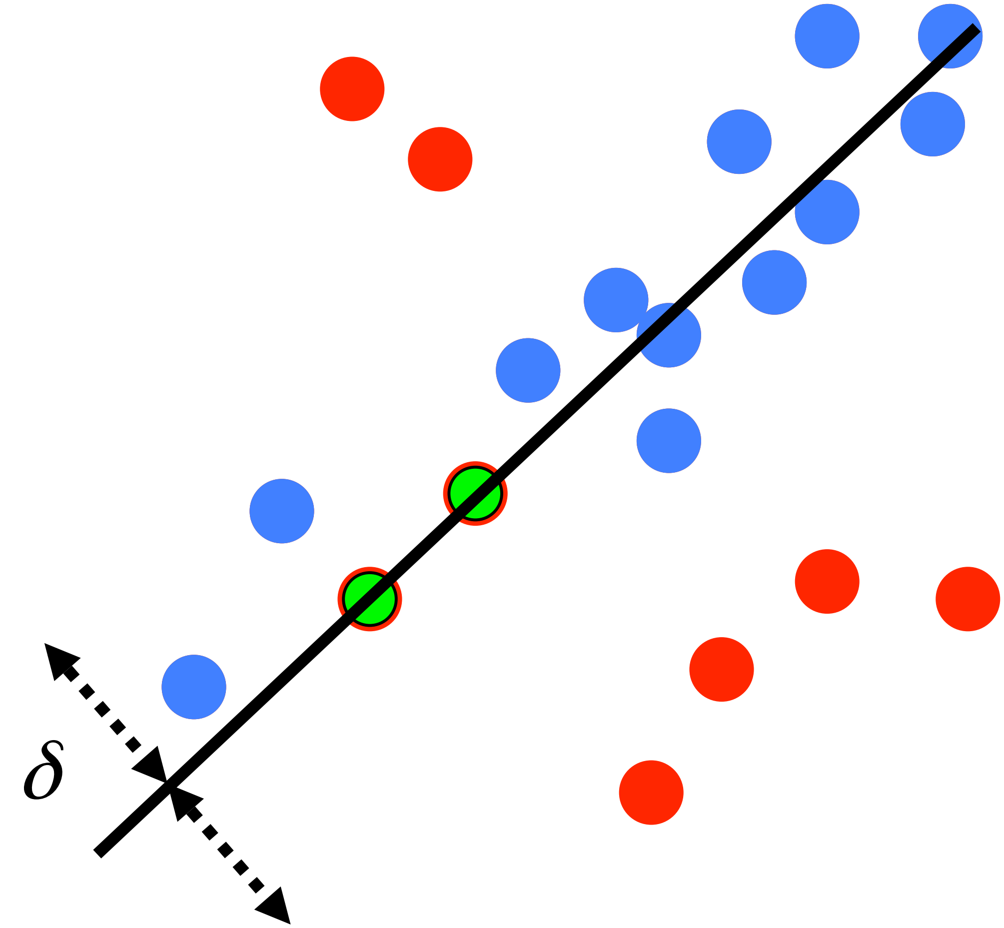
$$|\mathbf{P}| = ? = 6$$

Algorithm:

1. Select random sample of minimum required size to fit model [?]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

# RANSAC




$$|\mathbf{O}| = 6$$

$$|\mathbf{P}| = 14$$

Algorithm:

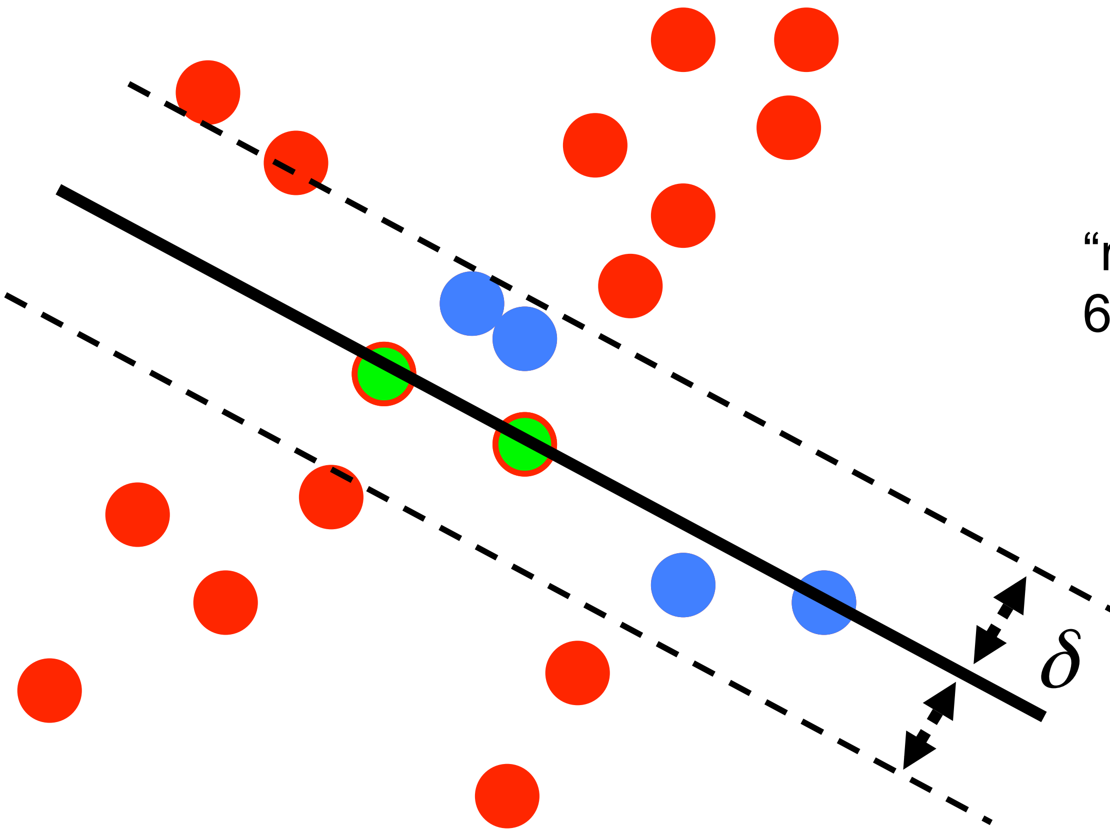
1. Select random sample of minimum required size to fit model [?]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

# How many samples?

- Number N of samples required to ensure, with a probability p, that at least one random sample produces an inlier set that is free from “real” outliers.
- Usually, $p$=0.99

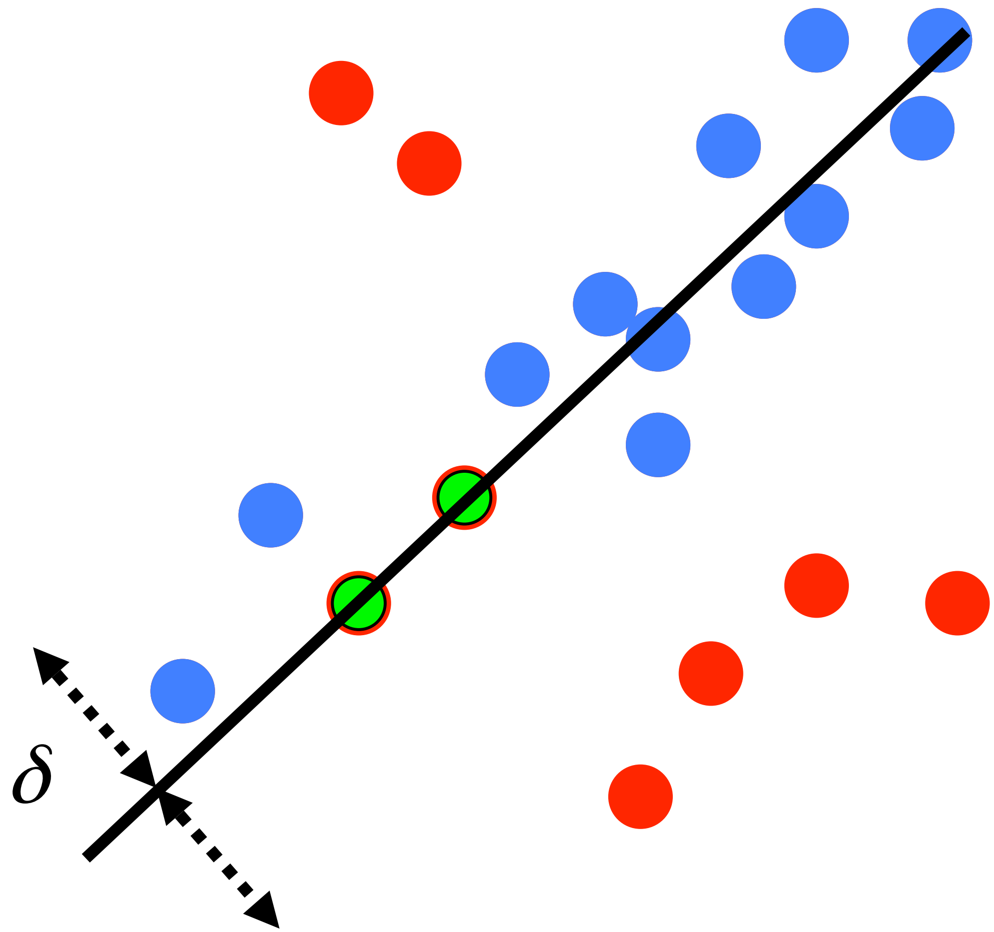
# Example




“real“ outlier ratio is 6/20 = 30%

- Here a random sample is given by two green points
- Estimated inlier set is given by the green+blue points
- How many “real” outliers we have here? 2

# Example




“real“ outlier ratio is
6/20 = 30%

- Random sample is given by two green points
- Estimated inlier set is given by the green+blue points
- How many “real” outliers we have here? 0

# How many samples?

- Number N of samples required to ensure, with a probability p, that at least one random sample produces an inlier set that is free from “real” outliers for a given s and e.
- E.g., *p*=0.99

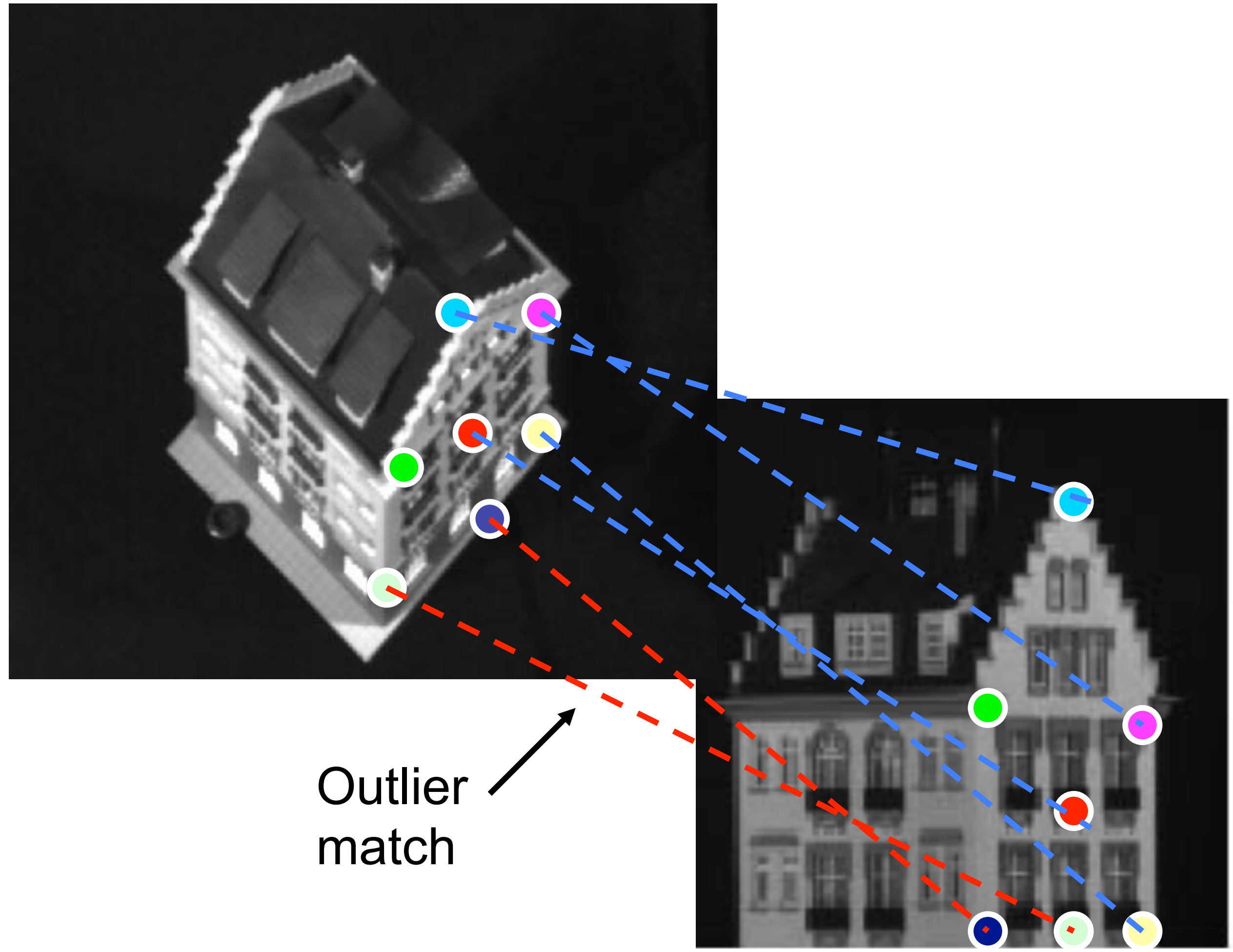
$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$ [Eq. 13]

| | proportion of outliers $e$ | | | | | | |
|---|---|---|---|---|---|---|---|
| s | 5% | 10% | 20% | 25% | 30% | 40% | 50% |
| 2 | 2 | 3 | 5 | 6 | $\boxed{7}$ | 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 |

*e* = outlier ratio
s = minimum number needed to fit the model

Note: this table assumes “negligible” measurement noise

# Estimating H by RANSAC




•H → 8 DOF
•Need 4 correspondences

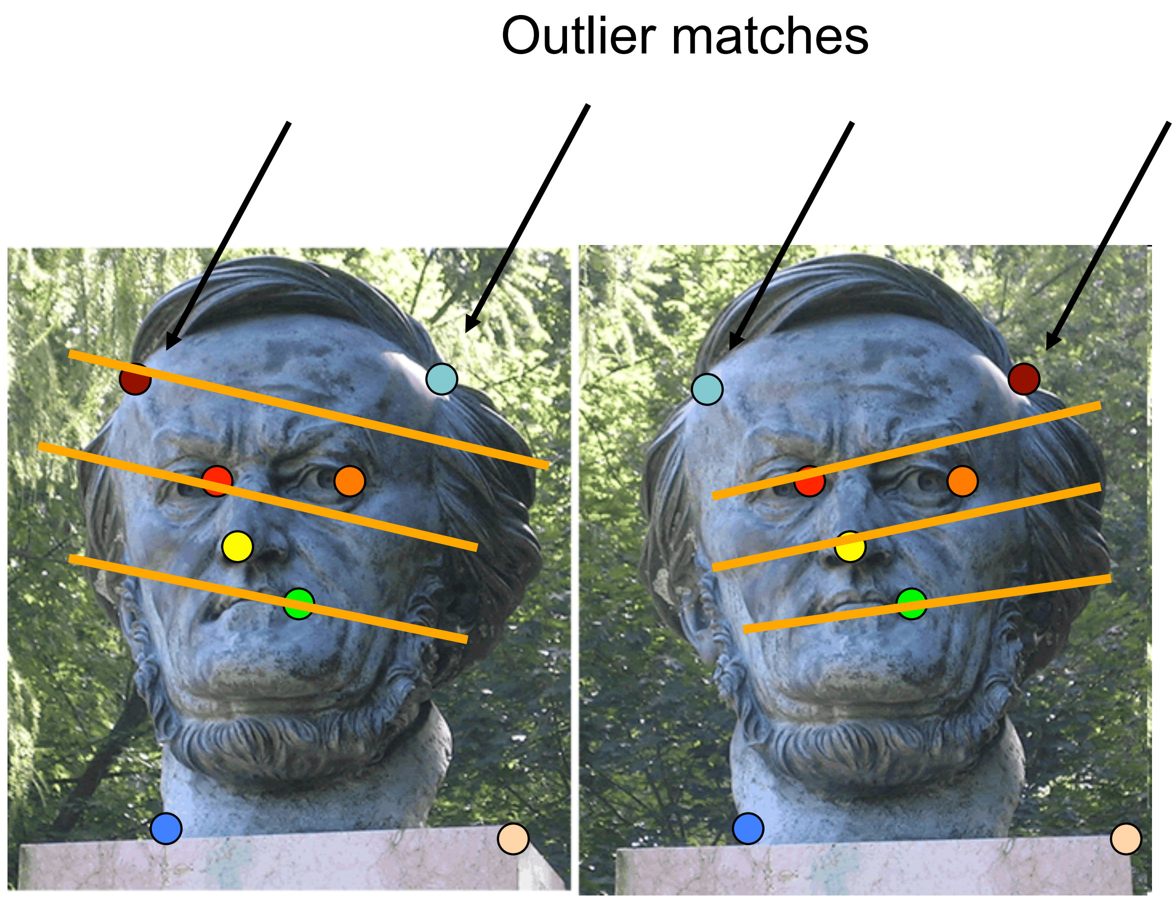
Sample set = set of matches between 2 images

## Algorithm:

1. Select a random sample of minimum required size [?]
2. Compute a putative model from these
3. Compute the set of inliers to this model from whole sample space

Repeat 1-3 until model with the most inliers over all samples is found

# Estimating F by RANSAC




- F → 7 DOF
- Need 7 (8) correspondences

Sample set = set of matches between 2 images

## Algorithm:

1. Select a random sample of minimum required size [?]
2. Compute a putative model from these
3. Compute the set of inliers to this model from whole sample space

Repeat 1-3 until model with the most inliers over all samples is found

# RANSAC - conclusions

Good:

- Simple and easily implementable
- Successful in different contexts

Bad:

- Many parameters to tune
- Trade-off accuracy-vs-time
- Cannot be used if ratio inliers/outliers is too small

# Fitting

Goal: Choose a parametric model to fit a certain quantity from data

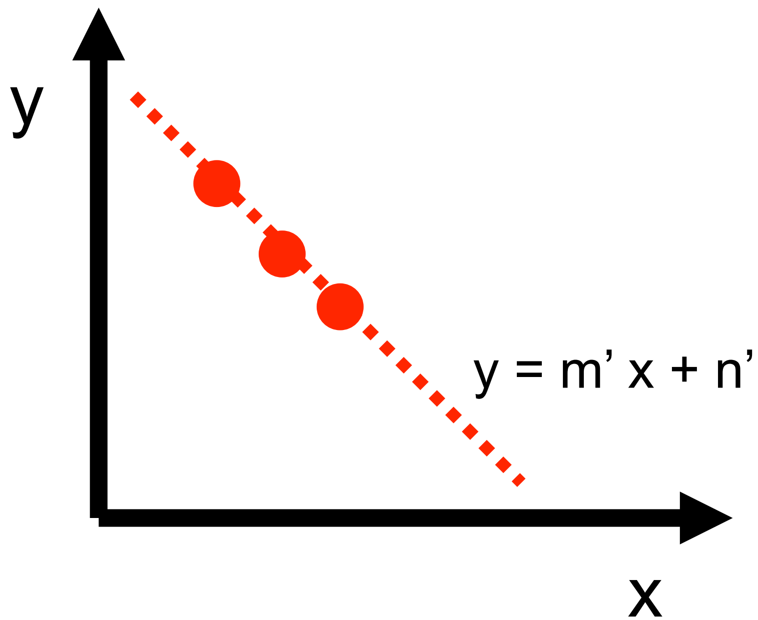
## Techniques:

- Least square methods
- RANSAC
- Hough transform

# Hough transform

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

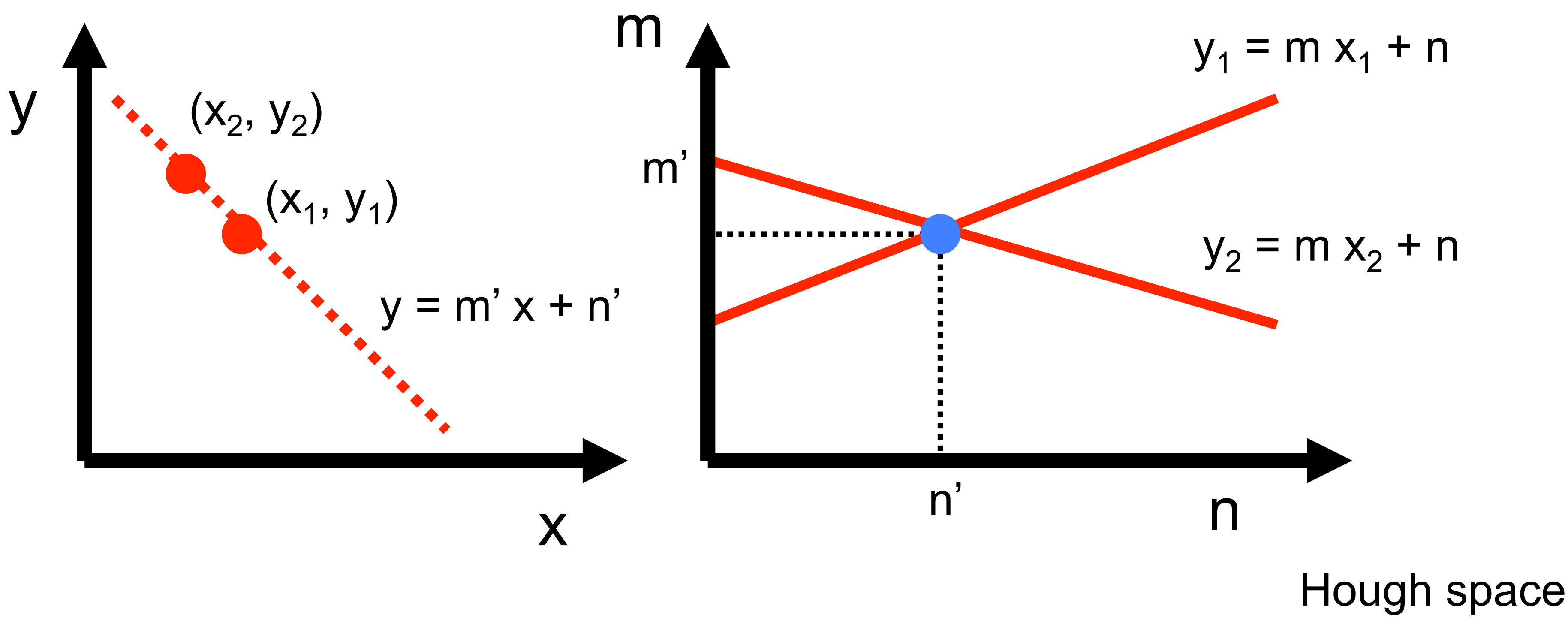
Given a set of points, find the curve or line that explains the data points best



# Hough transform

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

Given a set of points, find the curve or line that explains the data points best




$$y = m\,x + n$$

# Hough transform

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

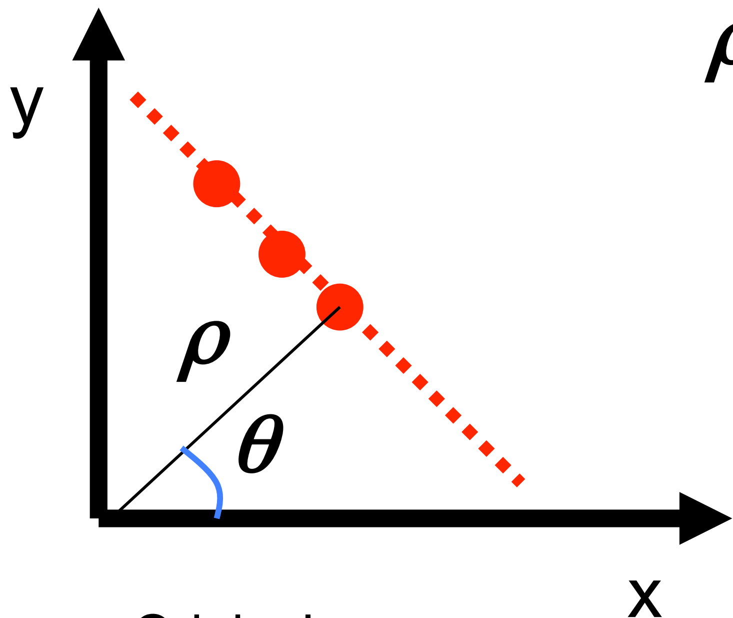
Any Issue? The parameter space [m,n] is unbounded…

# Hough transform

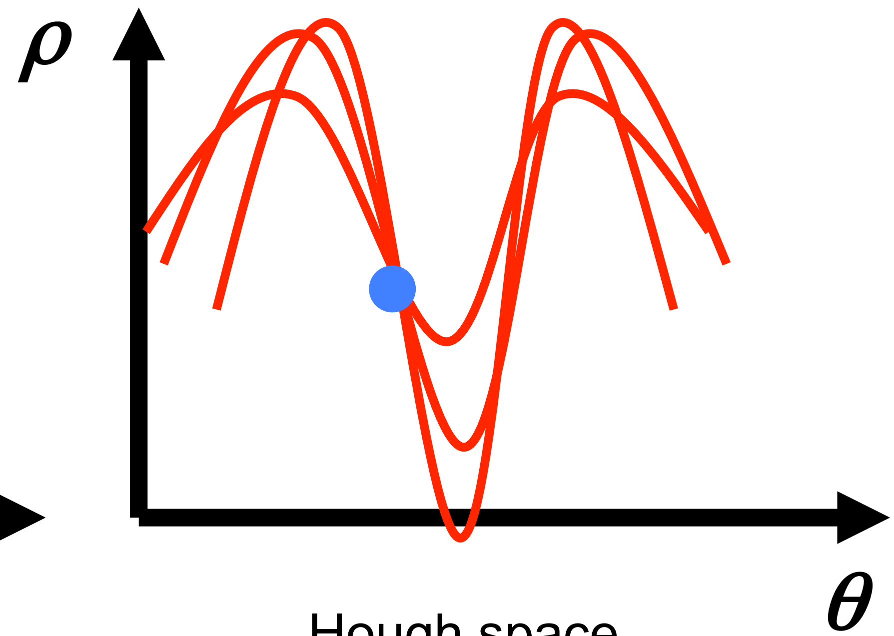
P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Any Issue?  The parameter space [m,n] is unbounded…

•Use a polar representation for the parameter space



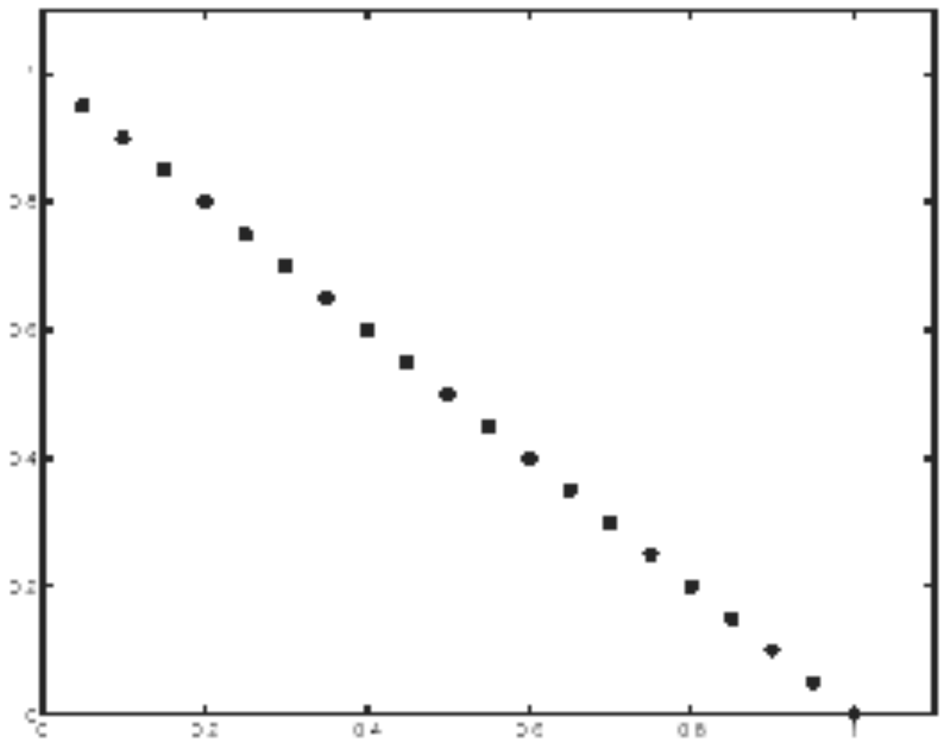

Original space



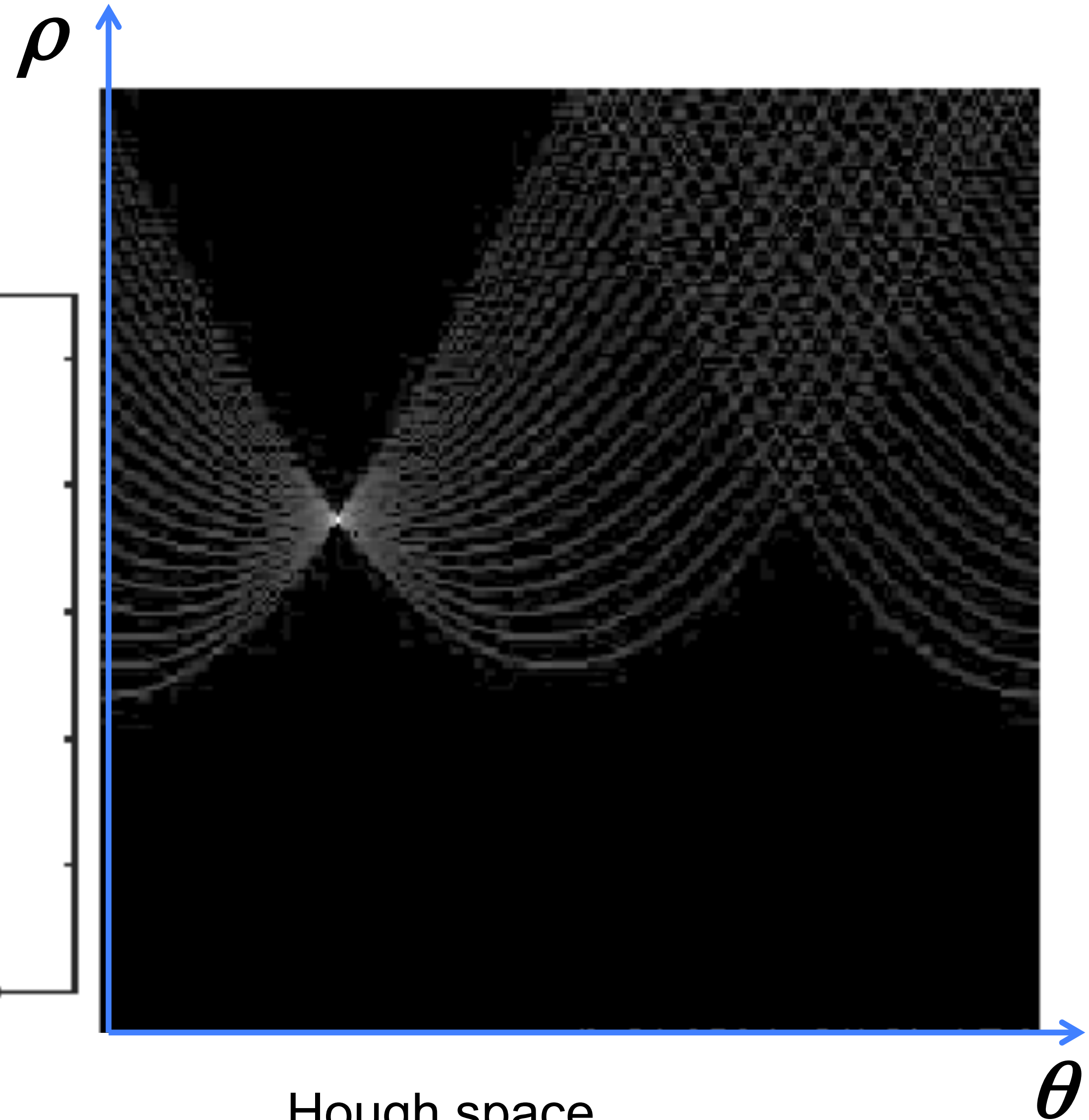

Hough space

$$x\cos\theta + y\sin\theta = \rho \qquad \text{[Eq. 13]}$$

# Hough transform - experiments



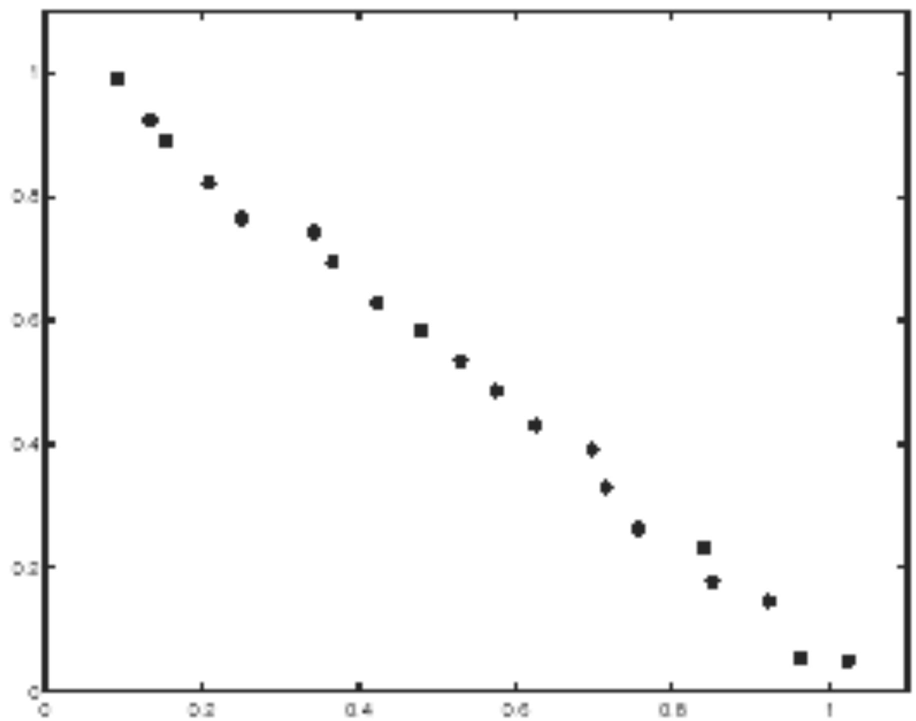
Original space



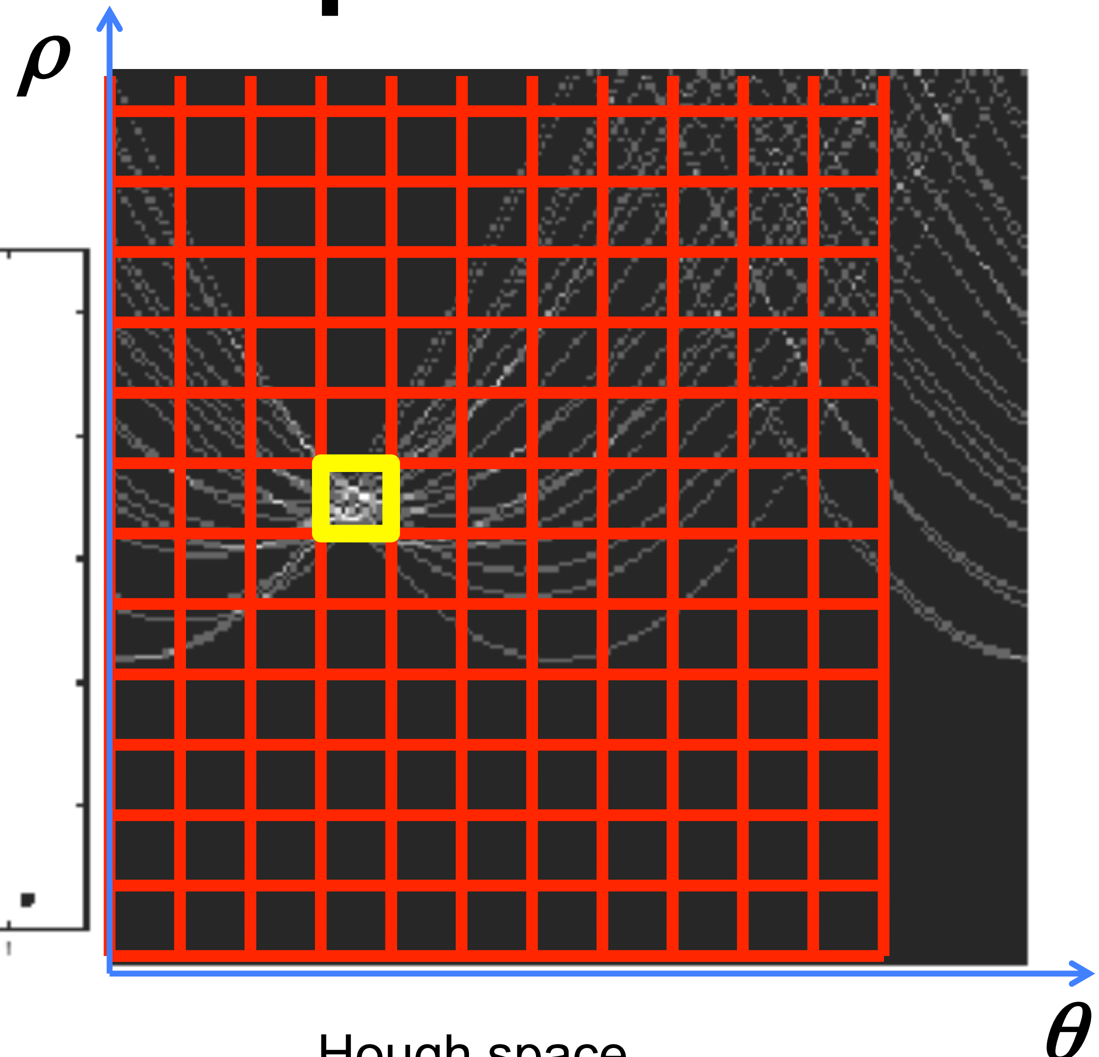

Hough space

# Hough transform - experiments

Noisy data



Original space



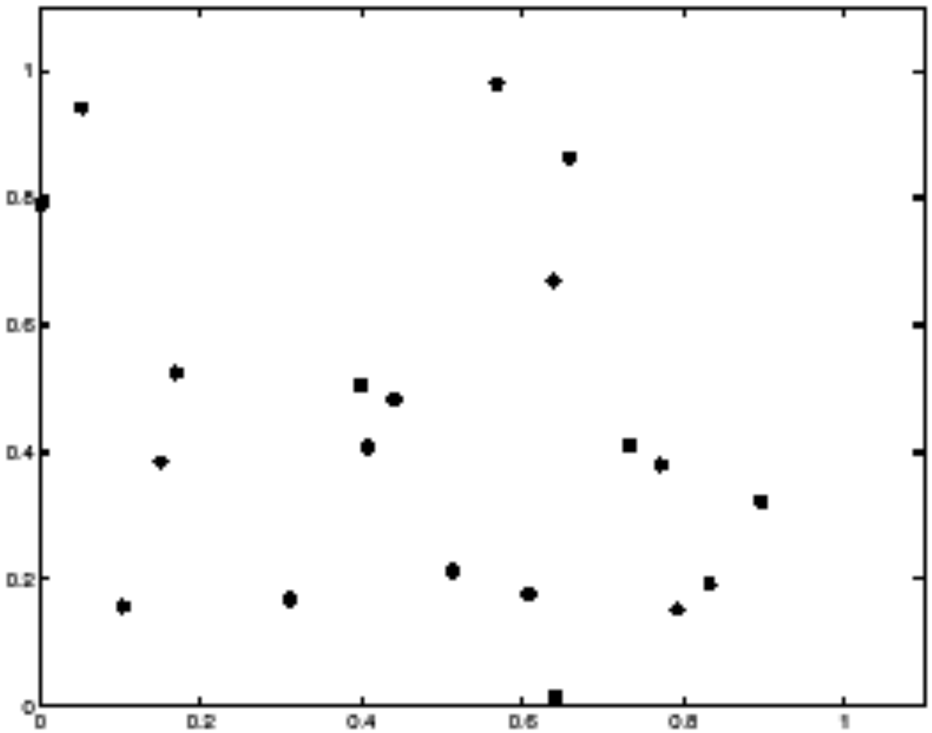

Hough space

How to compute the intersection point?

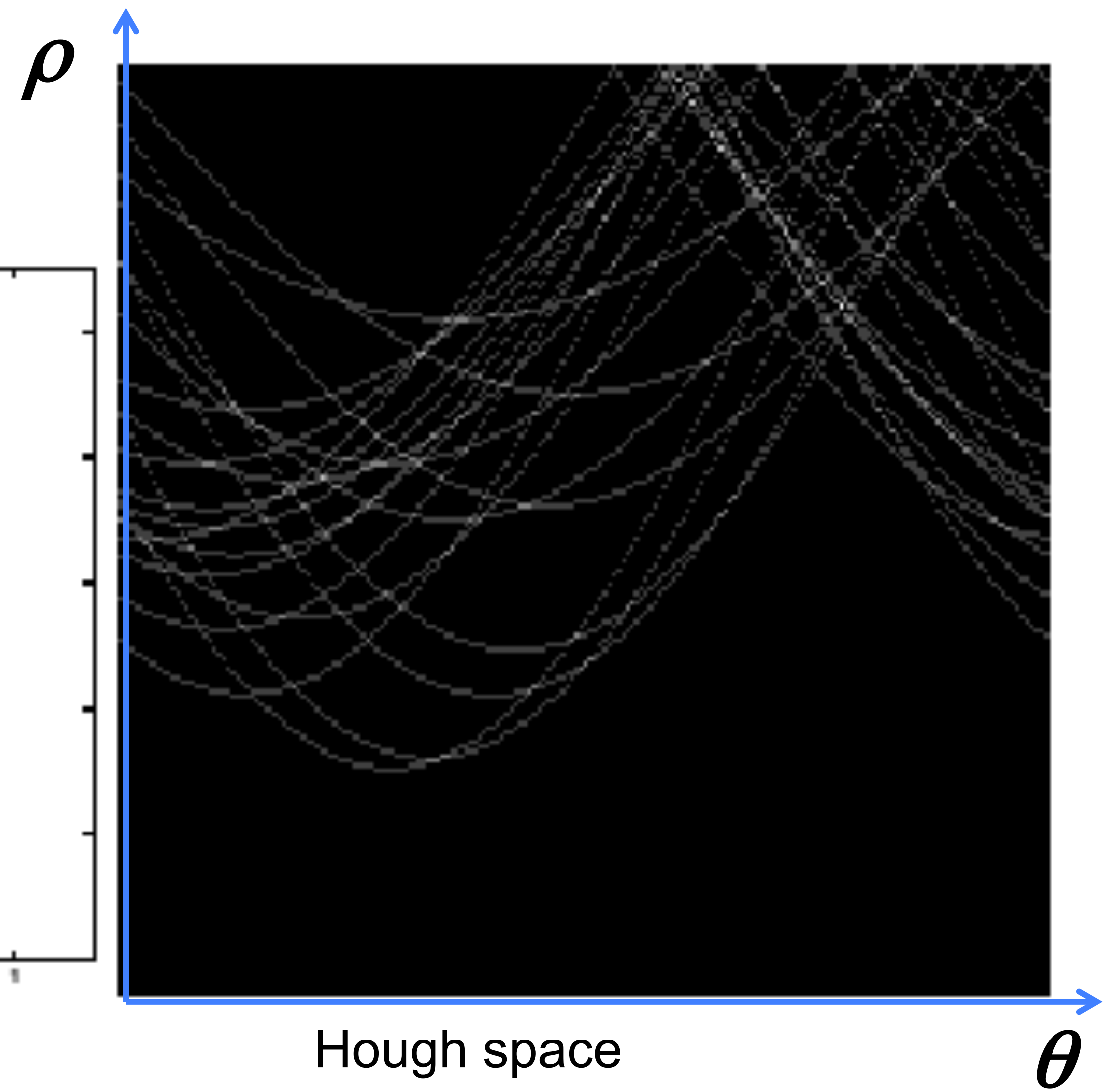
IDEA: introduce a grid a count intersection points in each cell

Issue: Grid size needs to be adjusted…

# Hough transform - experiments




Original space




Hough space

Issue: spurious peaks due to uniform noise

# Hough transform - conclusions

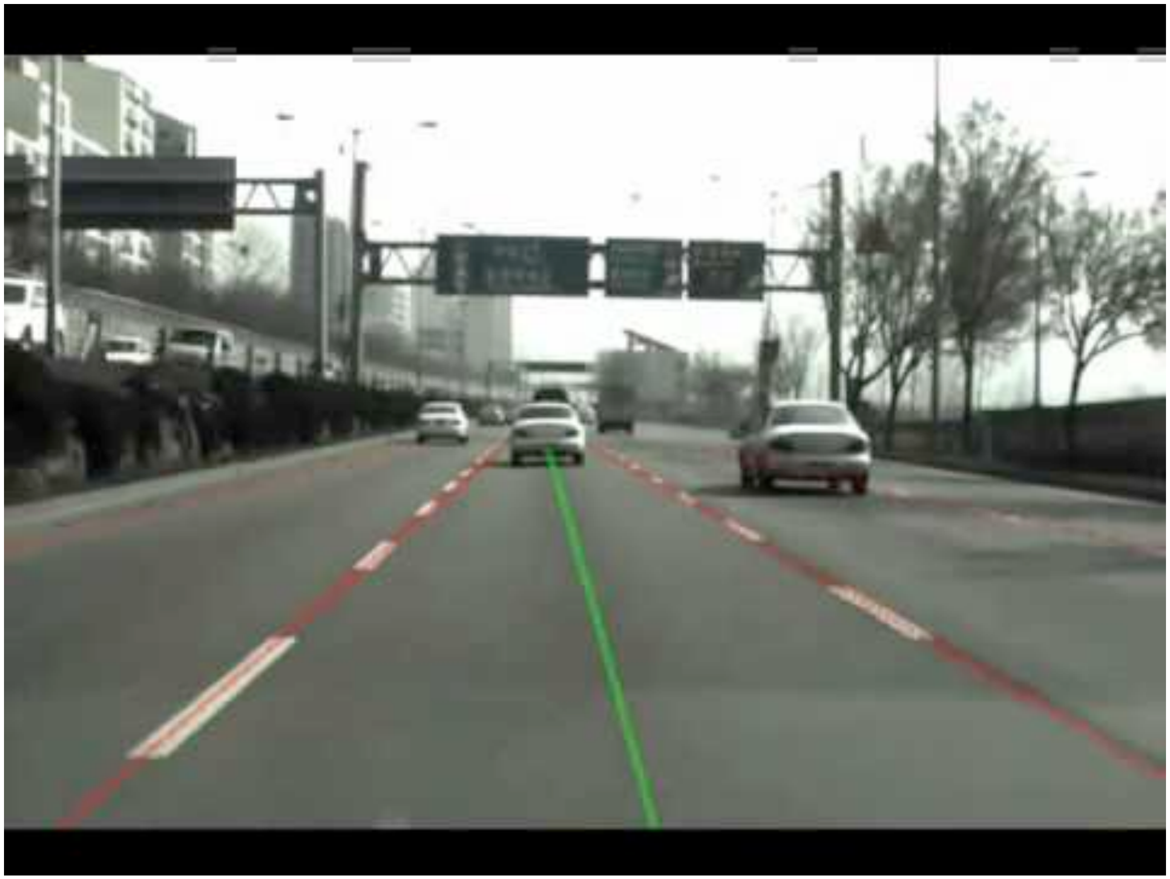
## Good:

- All points are processed independently, so can cope with occlusion/outliers
- Some robustness to noise: noise points unlikely to contribute consistently to any single cell

## Bad:

- Spurious peaks due to uniform noise
- Trade-off noise-grid size (hard to find sweet point)

# Applications – lane detection



Courtesy of Minchae Lee

# Applications – computing vanishing points

# Generalized Hough transform

[more on forthcoming lectures]

D. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, Pattern Recognition 13(2), 1981

- Parameterize a shape by measuring the location of its parts and shape centroid

- Given a set of measurements, cast a vote in the Hough (parameter) space

- Used in object recognition! (the implicit shape model)

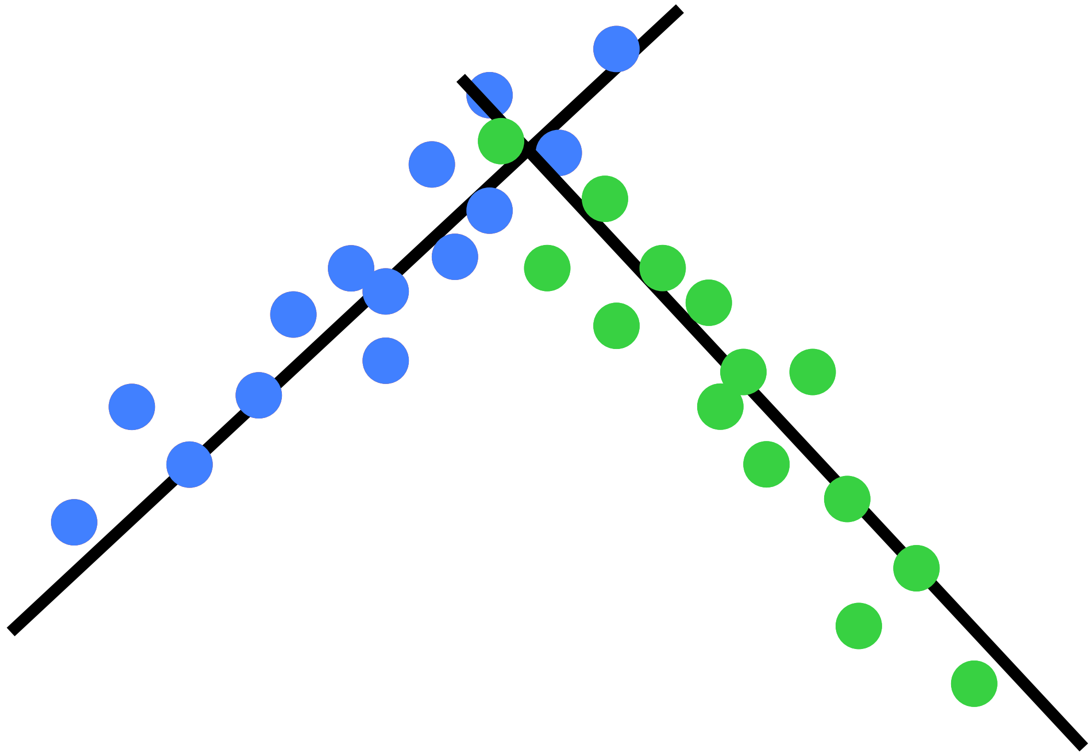
B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

# Lecture 9

# Fitting and Matching

- Problem formulation
- Least square methods
- RANSAC
- Hough transforms
- Multi-model fitting
- Fitting helps matching!

# Fitting multiple models



- Incremental fitting
- E.M. (probabilistic fitting)
- Hough transform

# Incremental line fitting

Scan data point sequentially (using locality constraints)

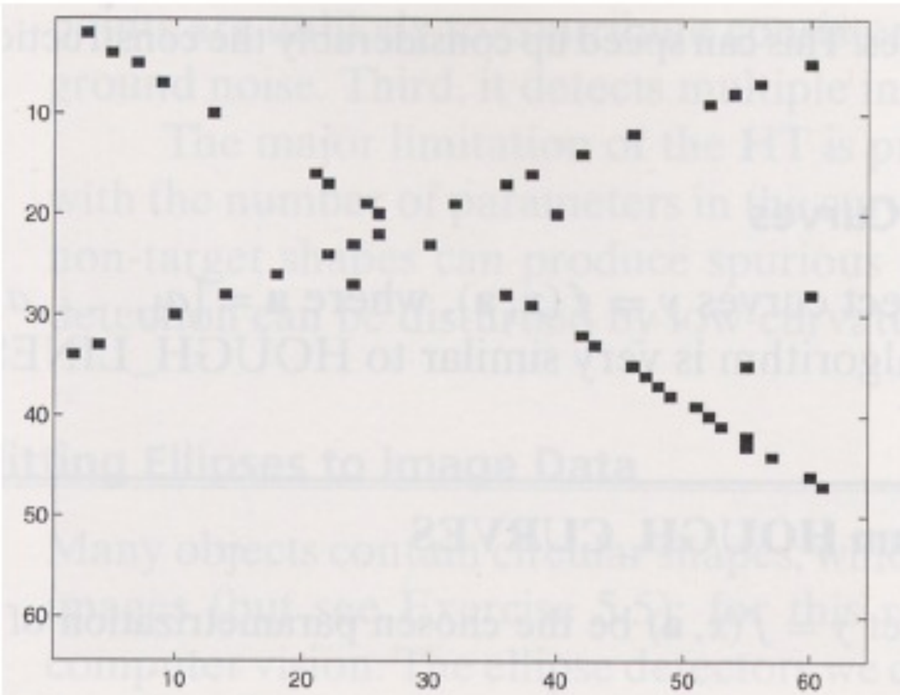
Perform following loop:

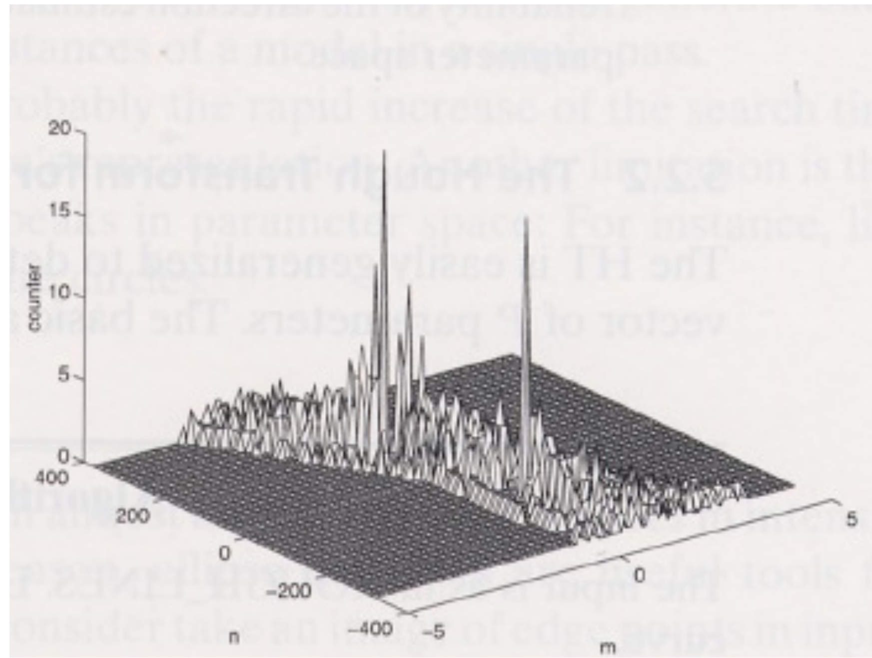
1. Select N point and fit line to N points
2. Compute residual $R_N$
3. Add a new point, re-fit line and re-compute $R_{N+1}$
4. Continue while line fitting residual is small enough,

➢ When residual exceeds a threshold, start fitting new model (line)

# Hough transform

Courtesy of unknown



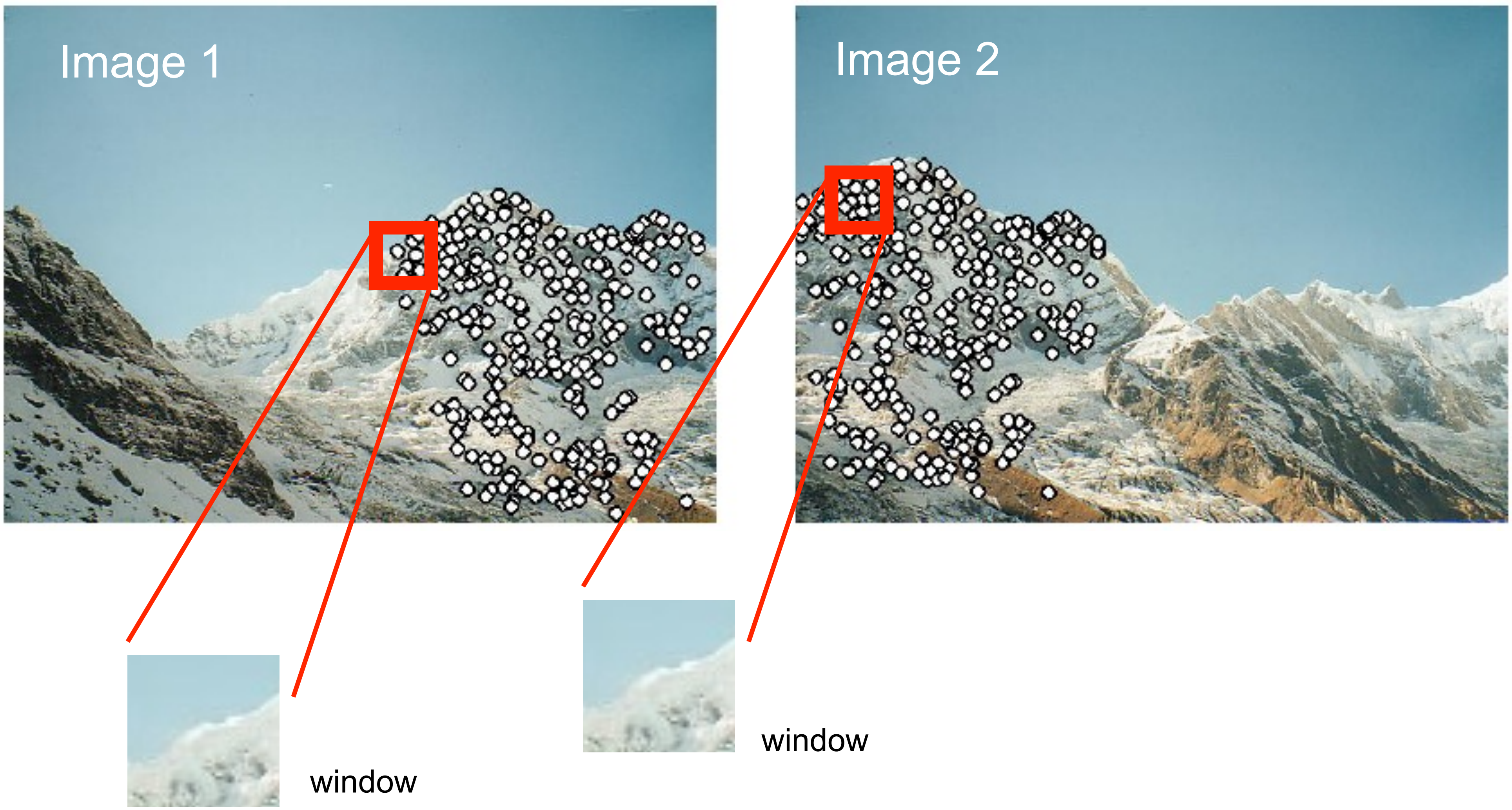




Same cons and pros as before…

# Lecture 9
# Fitting and Matching

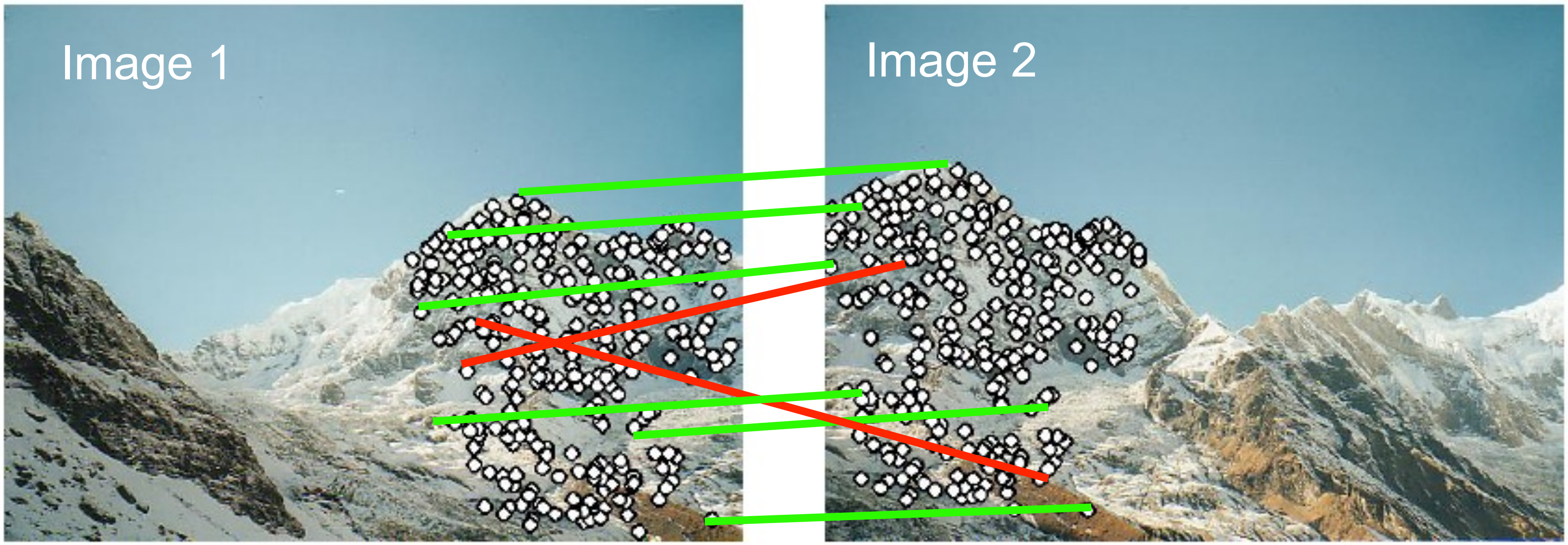
- Problem formulation
- Least square methods
- RANSAC
- Hough transforms
- Multi-model fitting
- Fitting helps matching!

# Fitting helps matching!




Features are matched (for instance, based on correlation)

# Fitting helps matching!




Matches based on appearance only

Green: good matches

Red: bad matches

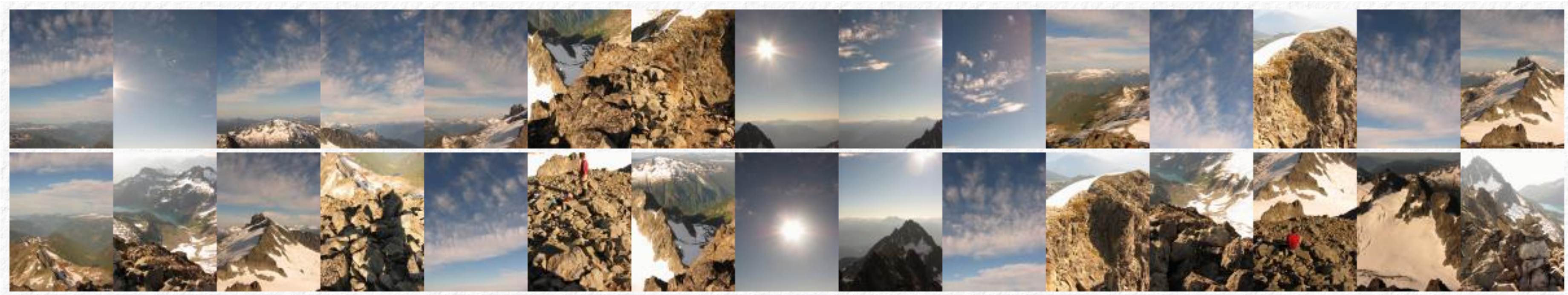
Idea:

- Fitting an homography H (by RANSAC) mapping features from images 1 to 2
- Bad matches will be labeled as outliers (hence rejected)!

# Fitting helps matching!

# Recognising Panoramas

M. Brown and D. G. Lowe. Recognising Panoramas. In Proceedings of the *9th International Conference on Computer Vision -- ICCV2003*



# Next lecture:
# Feature detectors and descriptors

# Least squares methods
## - fitting a line -

$$Ax = b$$

- More equations than unknowns

- Look for solution which minimizes $||Ax-b|| = (Ax-b)^T(Ax-b)$
- Solve $\frac{\partial (Ax-b)^T (Ax-b)}{\partial x_i} = 0$

- LS solution

$$x = (A^T A)^{-1} A^T b$$

# Least squares methods

## - fitting a line -

**Solving** $x = (A^t A)^{-1} A^t b$

$A^+ = (A^t A)^{-1} A^t$ = pseudo-inverse of A

$A = U \sum V^t$ = SVD decomposition of A

$A^{-1} = V \sum^{-1} U$

$A^+ = V \sum^+ U$

with $\sum^+$ equal to $\sum^{-1}$ for all nonzero singular values and zero otherwise

# Least squares methods

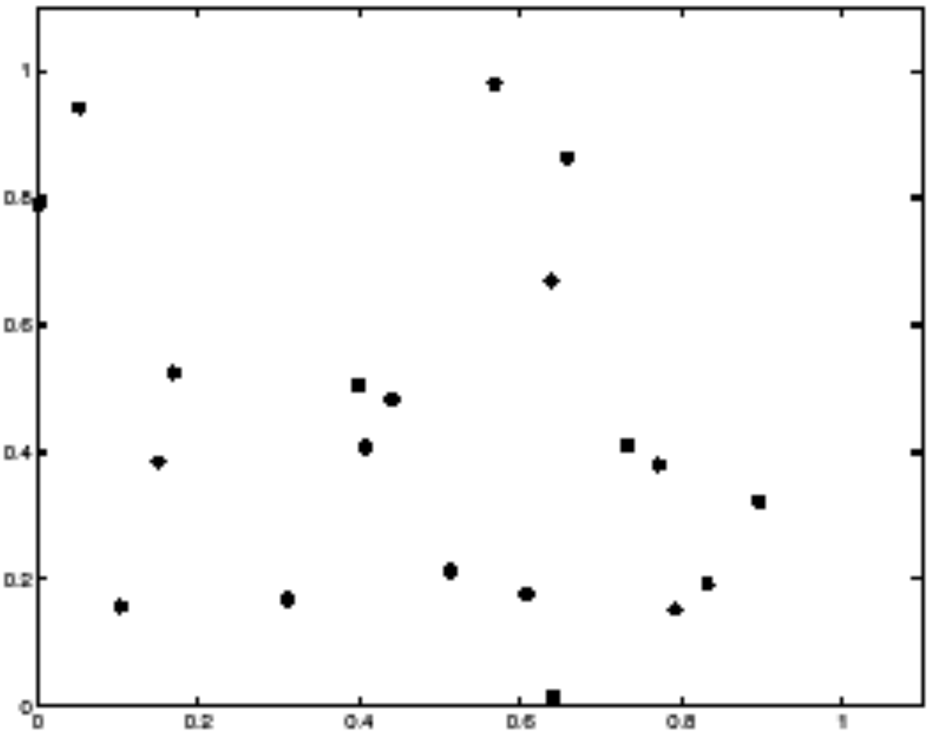
## - fitting an homography -

$$
\begin{aligned}
h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' - x' &= 0 \\
h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' - y' &= 0
\end{aligned}
$$

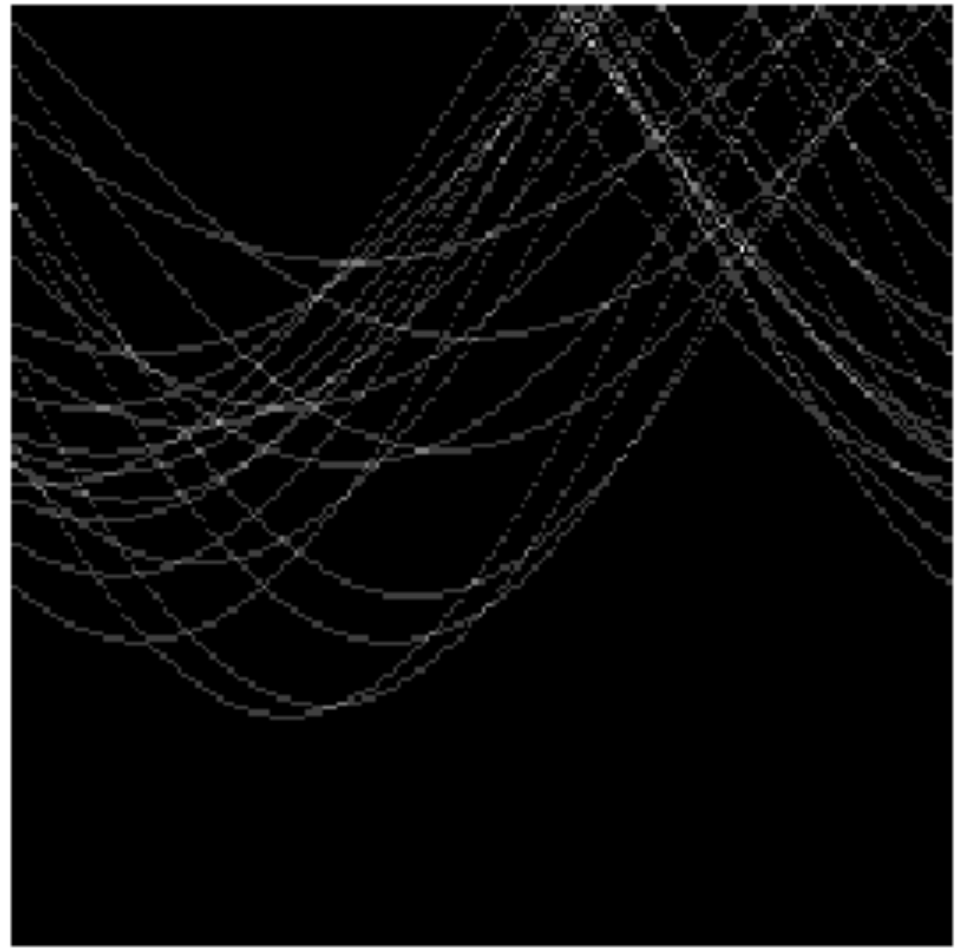
From n>=4 corresponding points: $\boxed{A\,h = 0}$

$$
\begin{pmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 & -x'_2 \\
0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 & -y'_2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_n & y_n & 1 & 0 & 0 & 0 & -x_nx'_n & -y_nx'_n & -x'_n \\
0 & 0 & 0 & x_n & y_n & 1 & -x_ny'_n & -y_ny'_n & -y'_n
\end{pmatrix}
\begin{bmatrix} h_{1,1} \\ h_{1,2} \\ \vdots \\ h_{3,3} \end{bmatrix} = 0
$$

# Hough transform - experiments



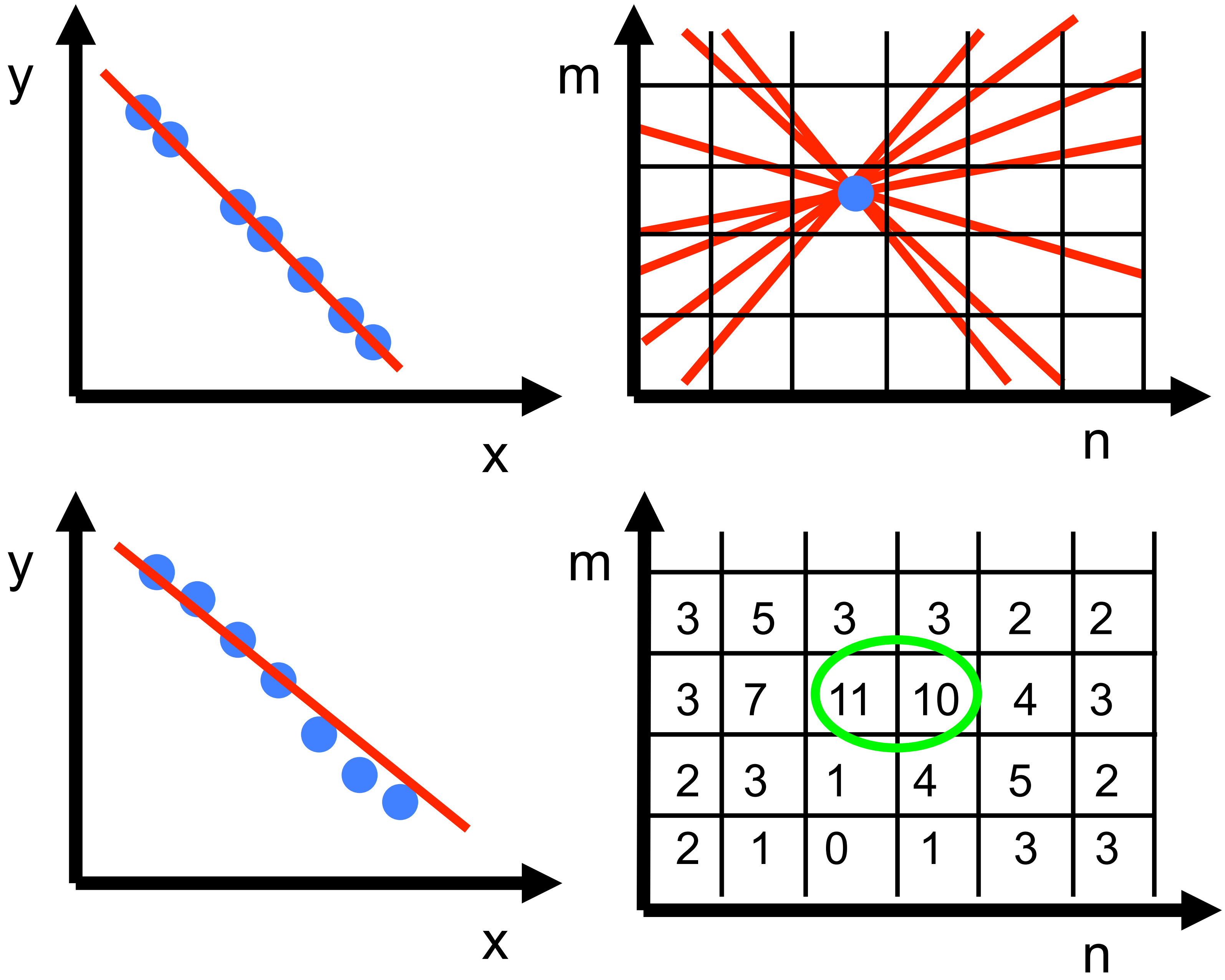

features



votes

Issue: spurious peaks due to uniform noise

# Hough transform



# Fitting helps matching!

**Images courtesy of Brandon Lloyd**