# Mathematical Principles in Visual Computing: Robust Estimation

Prof. Friedrich Fraundorfer

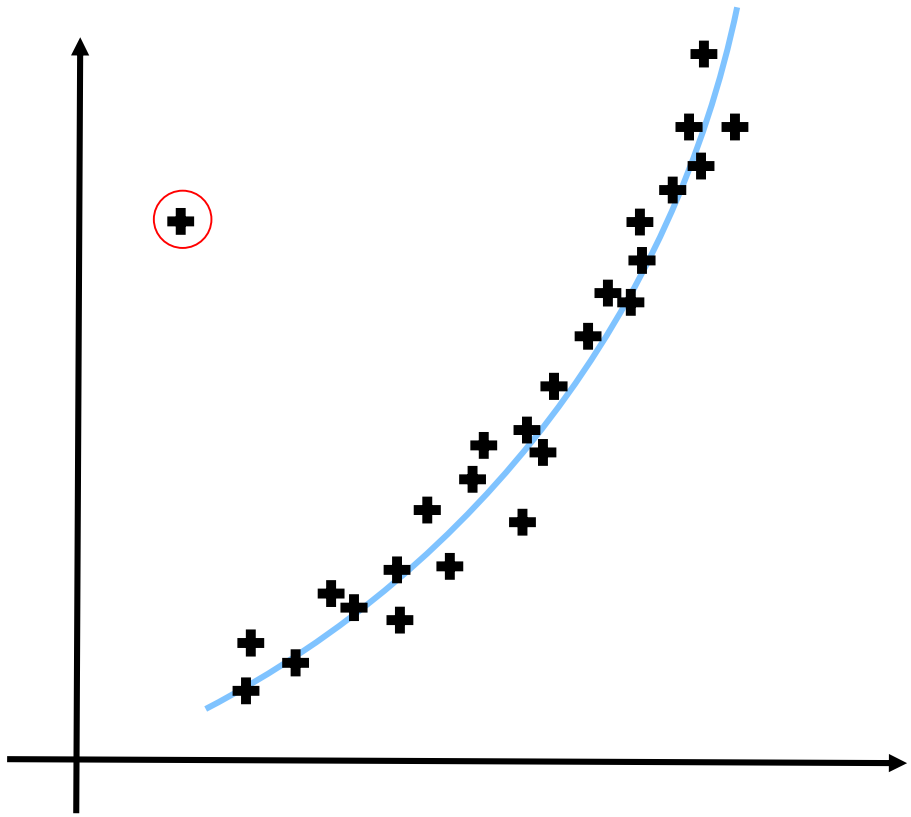
SS 2022

# Outline

- Robust estimation
  - Ransac
  - M-estimators (robust cost functions)
  - Iteratively reweighted least squares
- Rotation averaging

# Robust estimation in computer vision

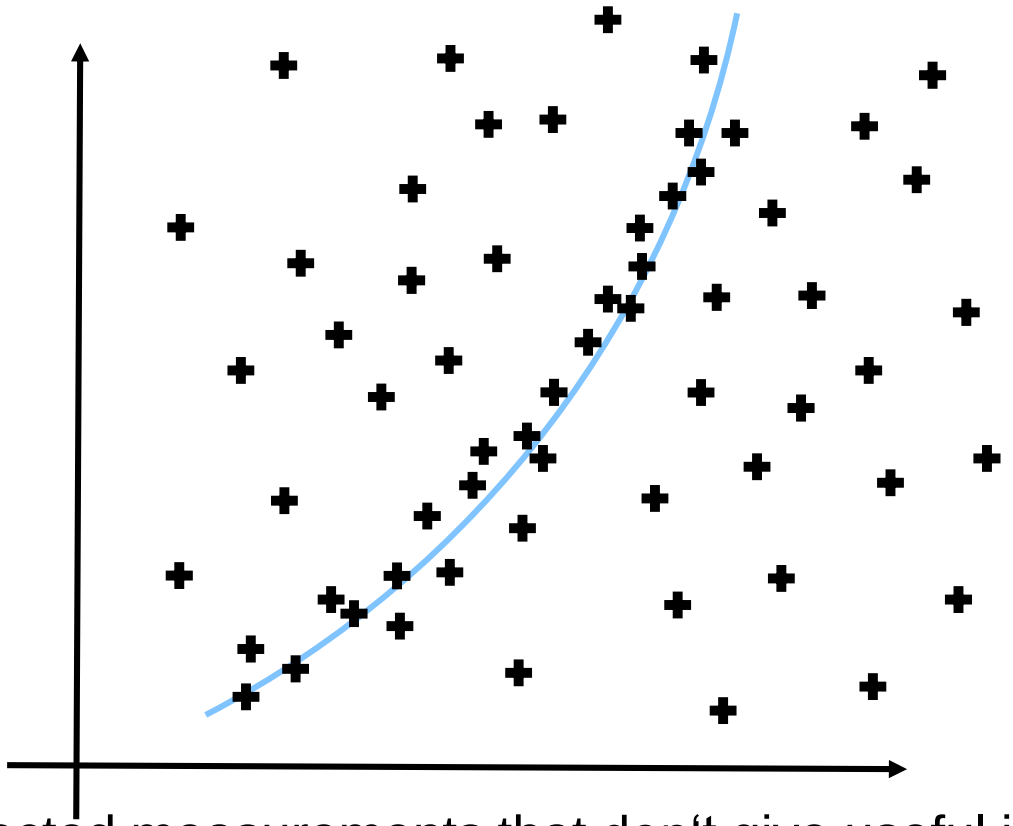
- Outliers in general estimation



- Rare unexpected measurements that don‘t fit the model

# Robust estimation in computer vision

- Outliers in computer vision



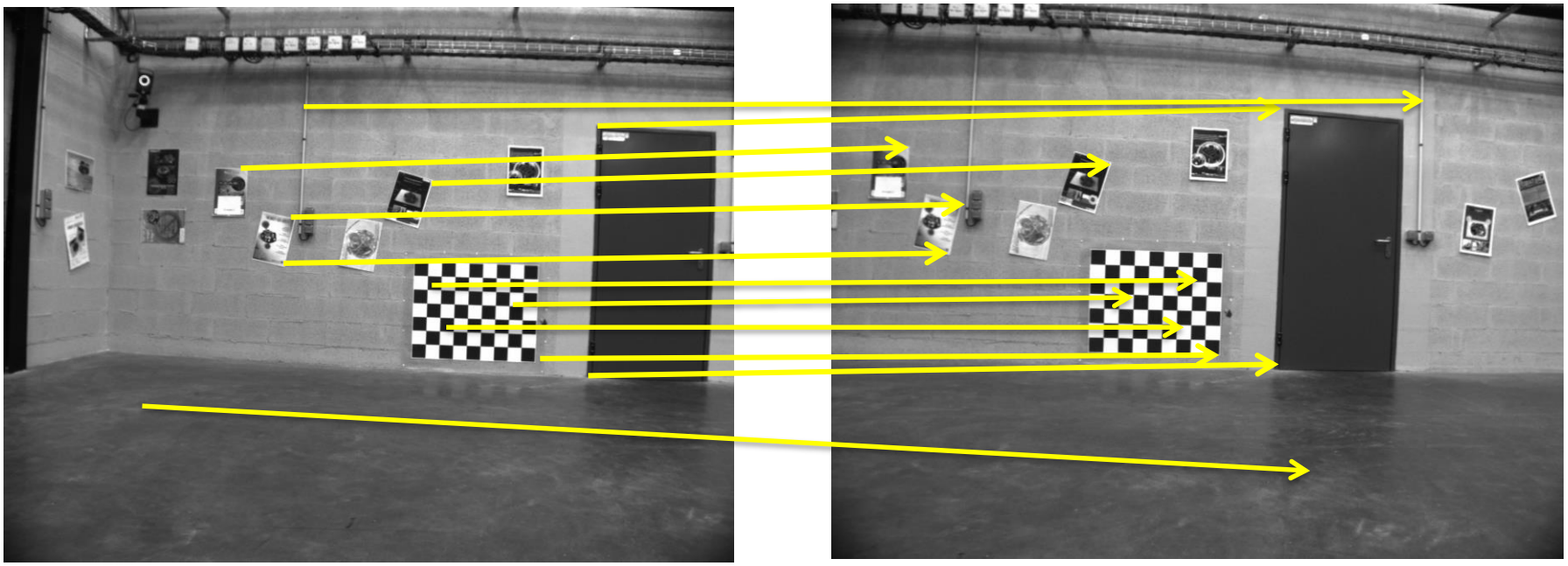
- Frequent expected measurements that don‘t give useful information

# Robust estimation in computer vision

- Multiple areas in computer vision require robust estimation techniques
  - Camera pose estimation
  - Triangulation
  - 2D Registration
  - 3D Registration (also computer graphics)
  - 3D Model fitting (computer graphics)
  - Line detection
  - Rotation averaging
  - Bundle adjustment
- Techniques:
  - Ransac
  - Iteratively reweighted least squares (robust loss functions)

# Camera motion estimation

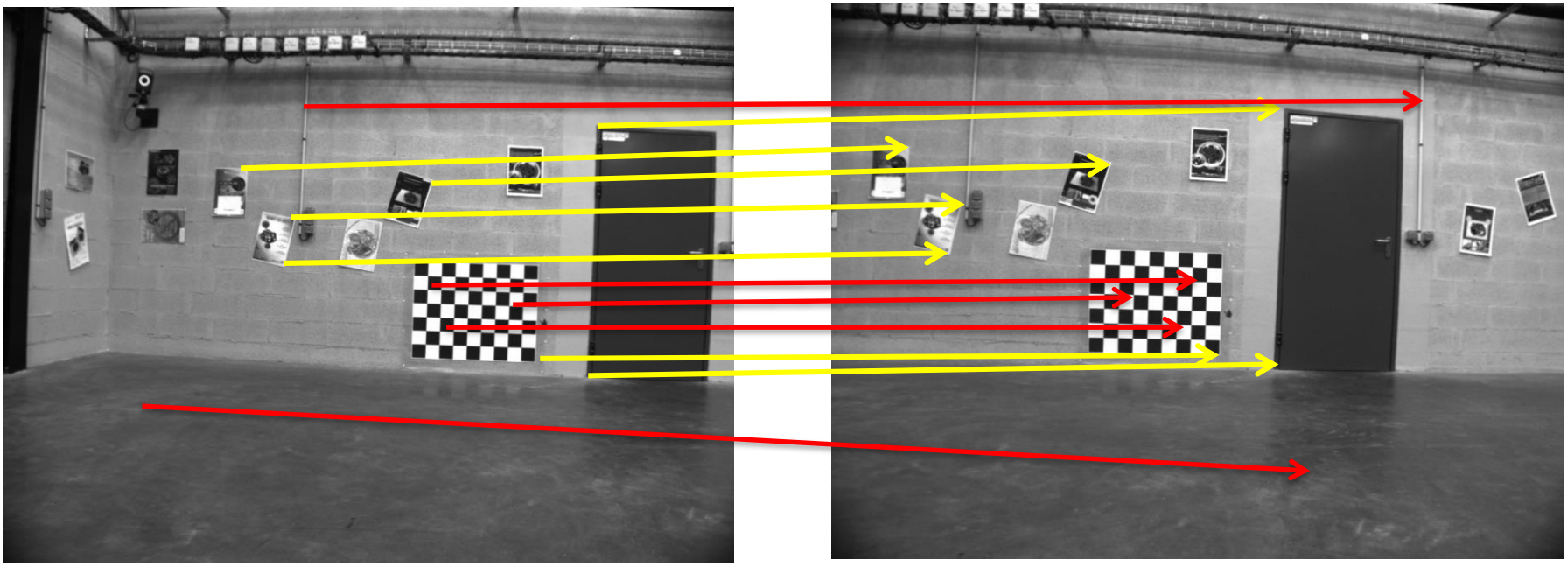
- Needs to be robust against mismatches



Yellow: automatically generated image matches
(contain mis-matches)

# Camera motion estimation

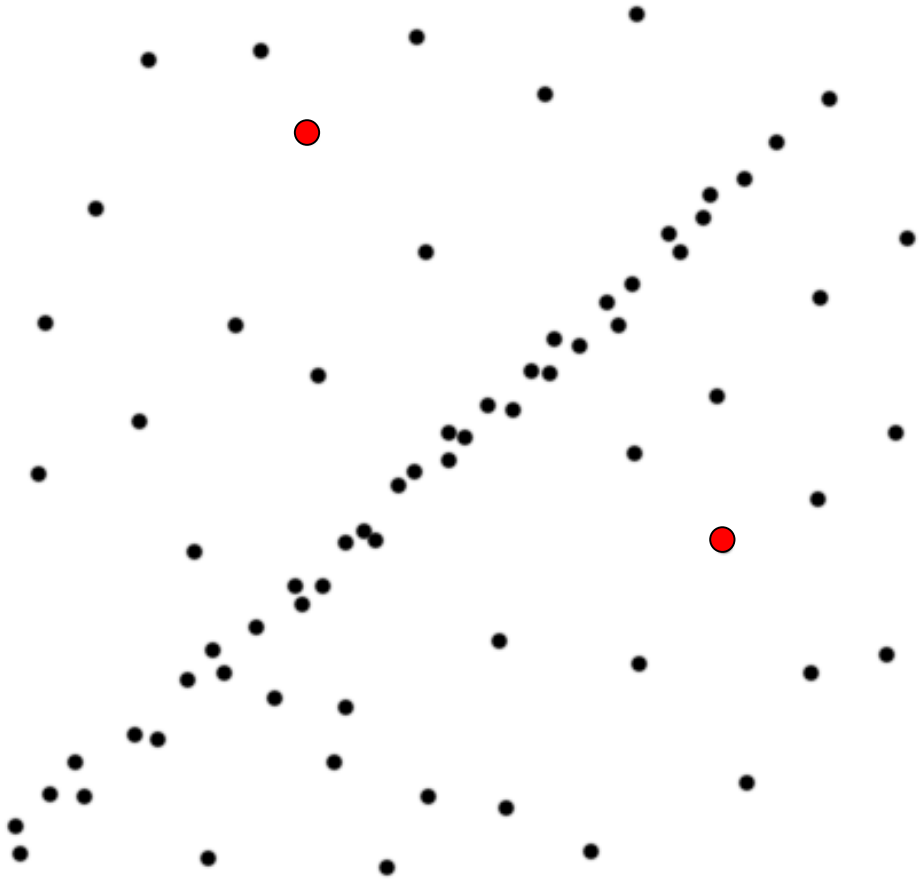
- All feature matches need to follow the same motion



Yellow: correct matches
Red: incorrect matches.

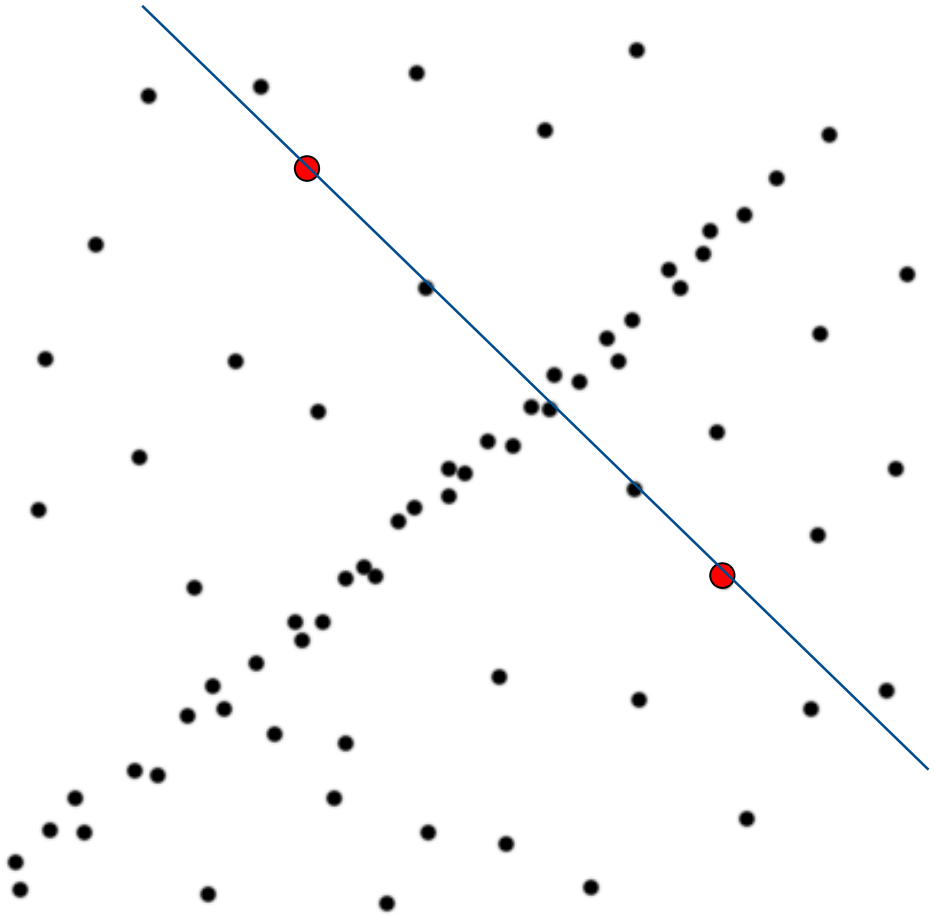
# RANSAC Example: Line Extraction

# RANSAC Example: Line Extraction



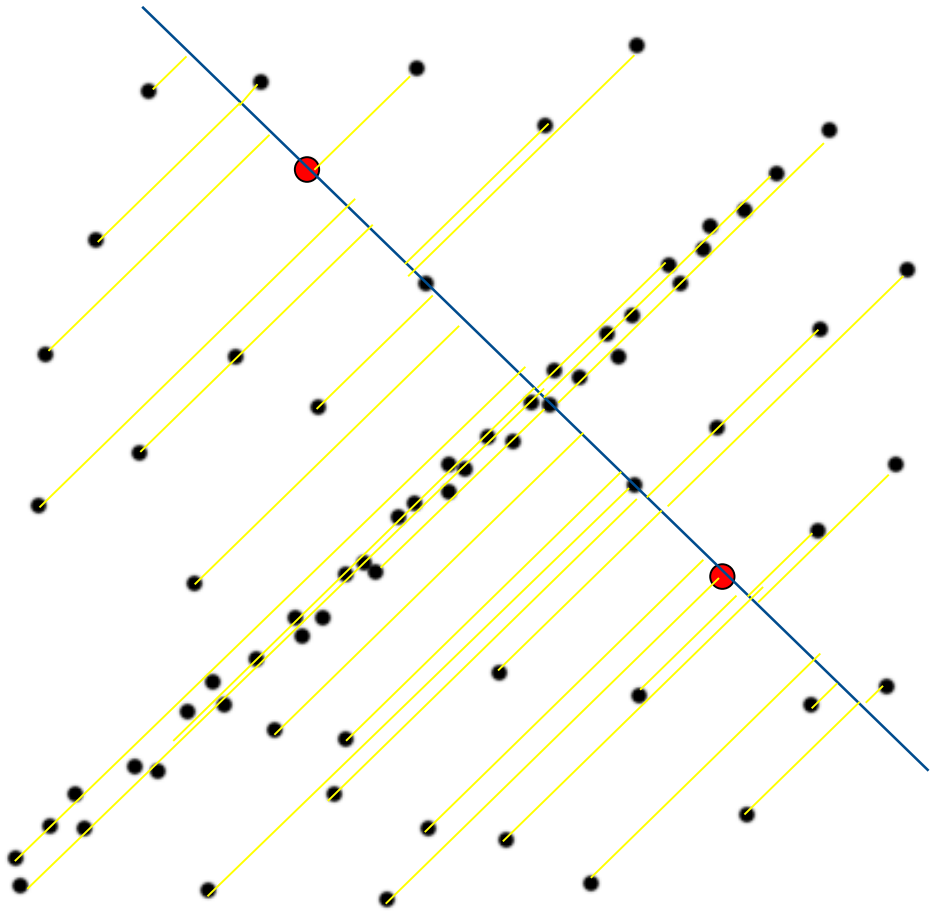
- Random sampling of 2 points

# RANSAC Example: Line Extraction



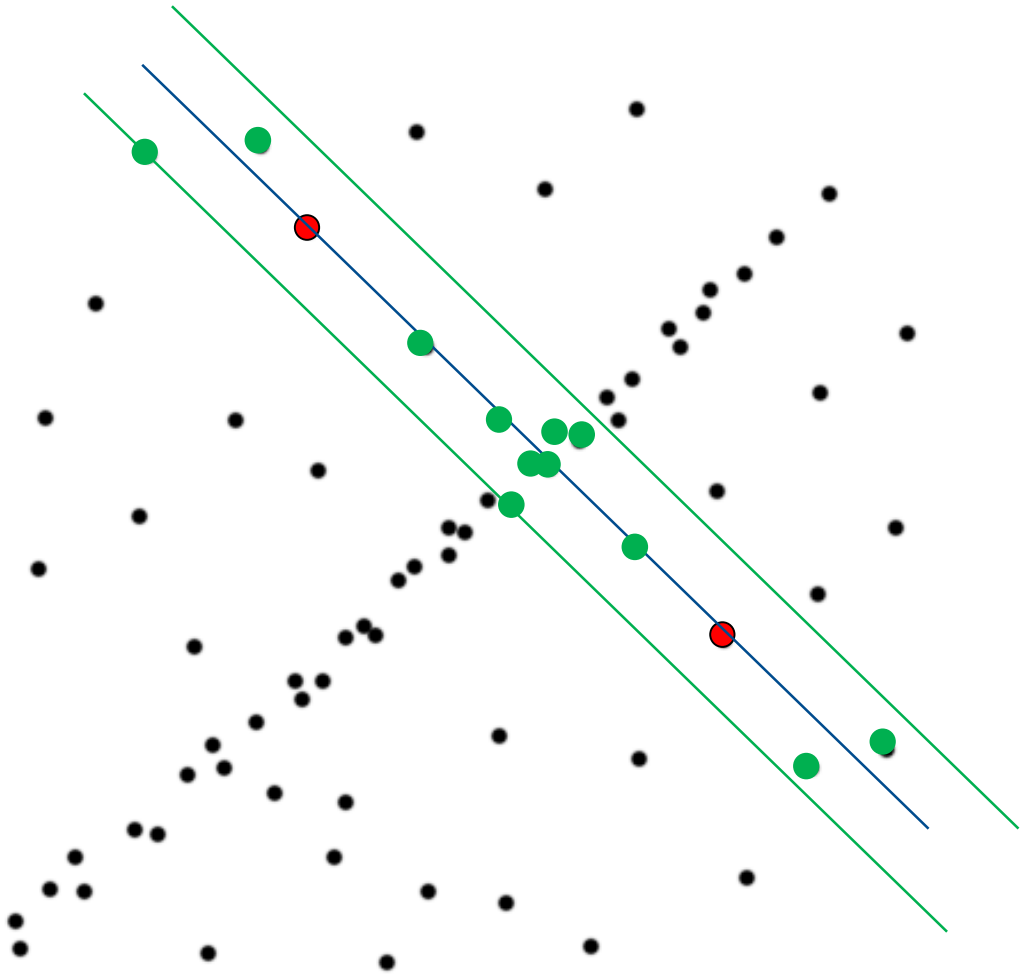
- Random sampling of 2 points
- Calculate line model from this 2 data points

# RANSAC Example: Line Extraction



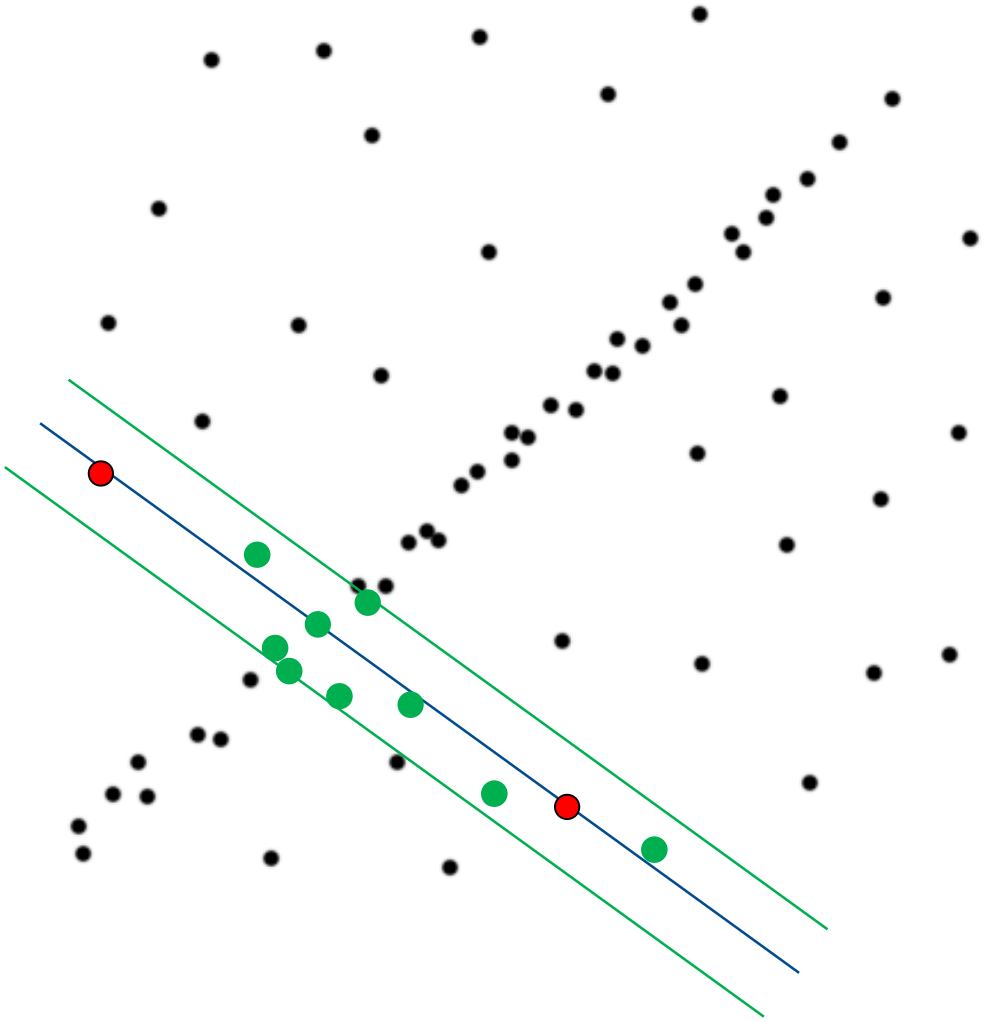
- Random sampling of 2 points
- Calculate line model from this 2 data points
- Calculate residual error for each data point (normal distance to line)

# RANSAC Example: Line Extraction



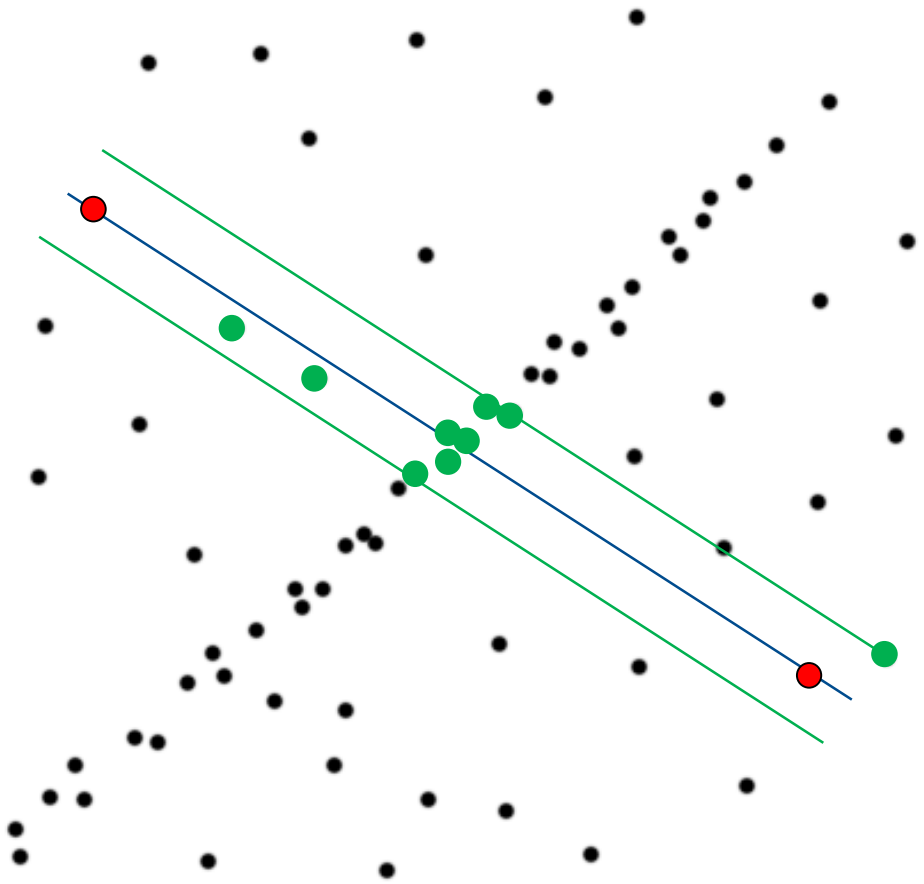
- Random sampling of 2 points
- Calculate line model from this 2 data points
- Calculate residual error for each data point (normal distance to line)
- Select data points that support current hypothesis

# RANSAC Example: Line Extraction



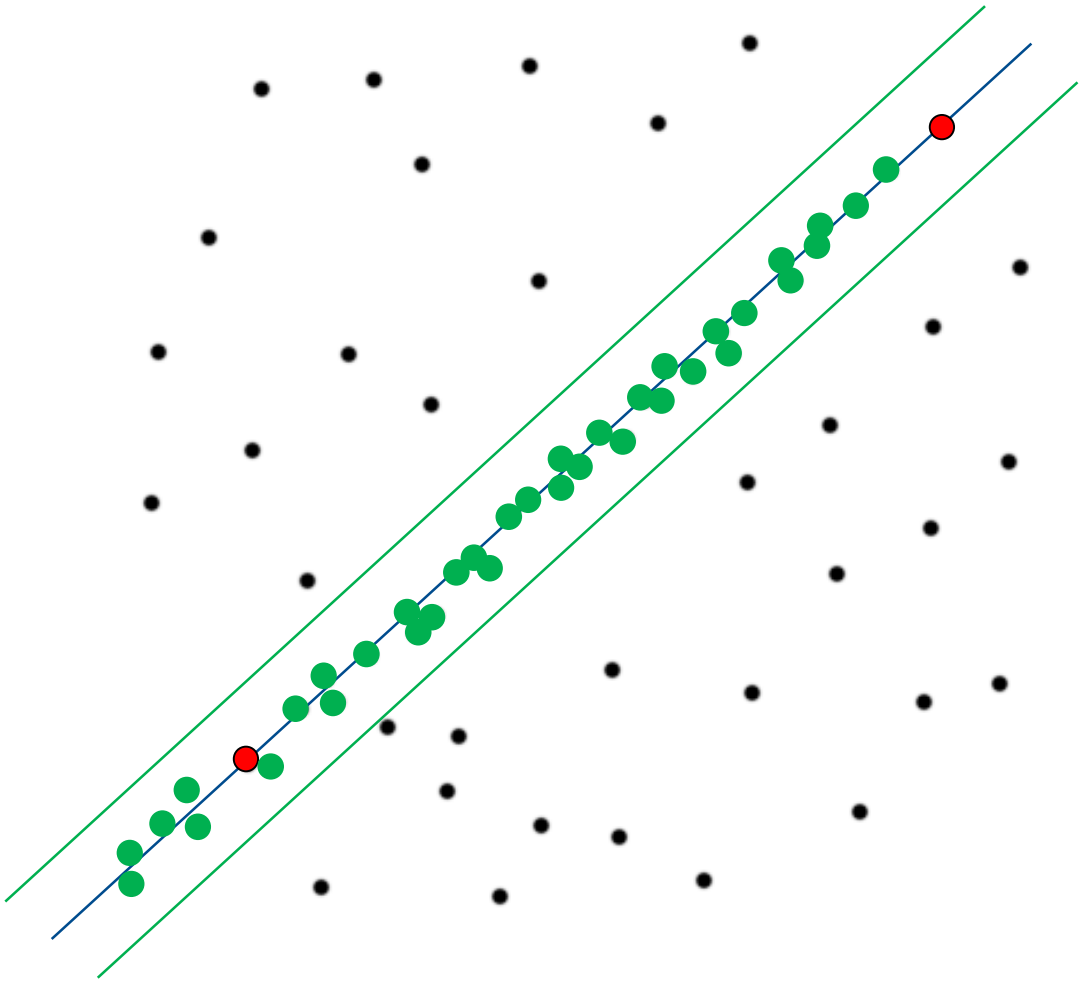
- Random sampling of 2 points
- Calculate line model from this 2 data points
- Calculate residual error for each data point (normal distance to line)
- Select data points that support current hypothesis
- Repeat sampling

# RANSAC Example: Line Extraction



- Random sampling of 2 points
- Calculate line model from this 2 data points
- Calculate residual error for each data point (normal distance to line)
- Select data points that support current hypothesis
- Repeat sampling

# RANSAC Example: Line Extraction



- Random sampling of 2 points
- Calculate line model from this 2 data points
- Calculate residual error for each data point (normal distance to line)
- Select data points that support current hypothesis
- Repeat sampling
- Until a hypothesis with a maximum number of inliers has been found

# RANSAC Algorithm

1. Initial: Let A be a set of N data points
2. Repeat
    1. Randomly select a sample of s data points from A
    2. Fit a model to these points
    3. Compute the distance of all other points to this model
    4. Construct the inlier set (i.e. count the number of data points whose distance from the model < threshold d
    5. Store these inliers
3. Until maximum number of iterations reached
4. The model with the maximum number of inliers is chosen as the solution to the problem
5. Re-estimate the model using all the inliers

# How many iterations of RANSAC ?

- The number of iterations N which is necessary to guarantee that at least a single correct solution is found can be computed by

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

- s is the number of data points from which a model can be minimally computed
- $\varepsilon$ is the percentage of outliers in the data (can only be guessed)
- p is the requested probability of success

- Example: p = 0.99, s = 5, $\varepsilon$ = 0.5 -> N = 145

# How many iterations of RANSAC ?

- RANSAC is an iterative method and it is non deterministic. It returns different solutions on different runs.

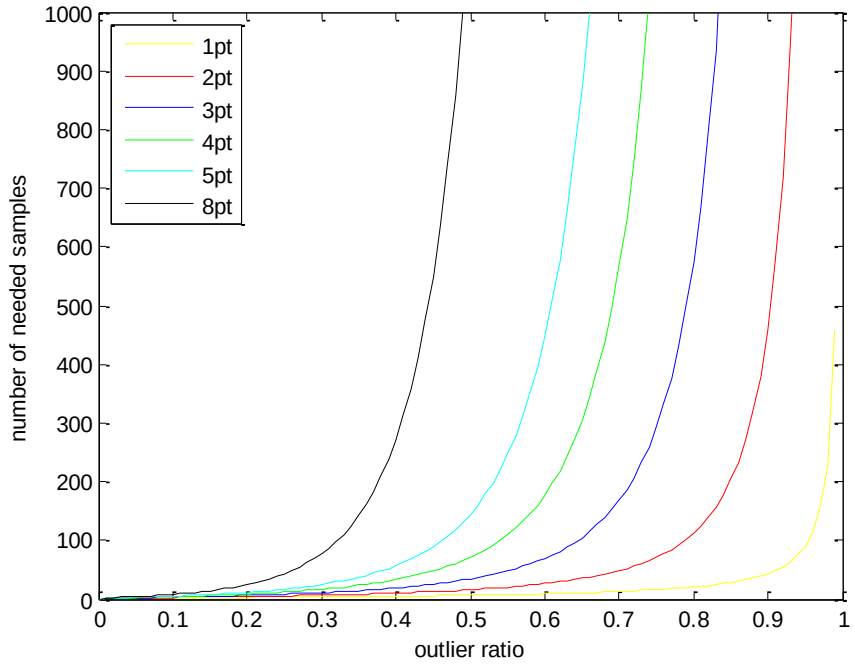
- For reasons of reliability, in many practical implementations N is usually multiplied by a factor of 10

- More advanced implementations of RANSAC estimate the fraction of inliers adaptively, for every iteration, and use it to update N.

# RANSAC iterations

- N is exponential in the number of data points s necessary to estimate the model
- Therefore, it is very important to use a minimal parameterization of the model

| Number of points (s) p=0.99, e=0.5 | 8 | 7 | 6 | 5 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| Number of iterations (N) | 1177 | 587 | 292 | 145 | 71 | 16 | 7 |

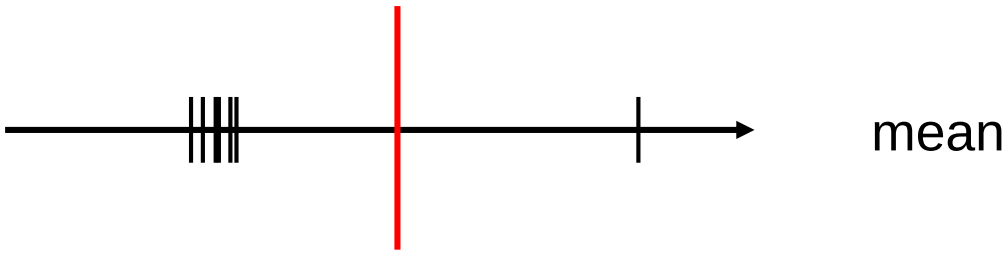


# Non-Robust estimation

- Standard (non-robust) model estimation with least-squares
- One-step closed form solution

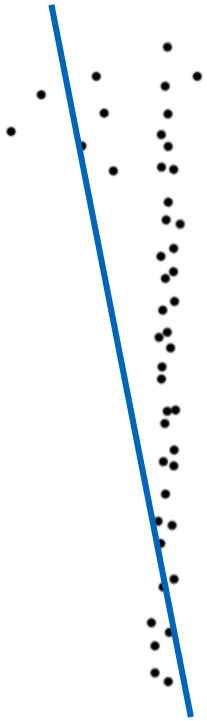
$$Ax = b \rightarrow \min_{x} \|Ax - b\|$$
$$x = (A^T A)^{-1} A^T b$$

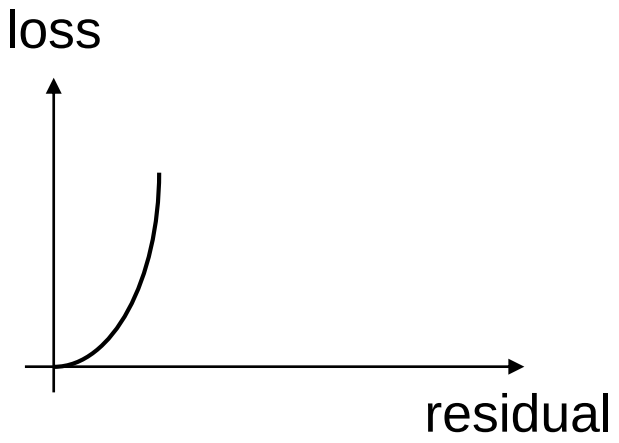
- Assumption: Residuals follow a Gaussian distribution!
- Not true for data that contains outliers



# Non-Robust estimation

- Least squares means quadratic loss function



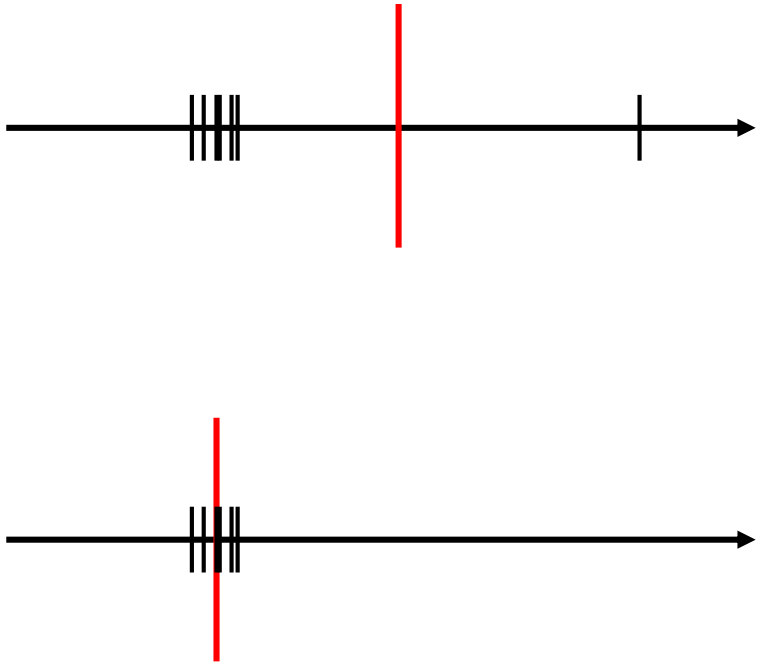



$$\min_x \sum_i r_i^2(x)$$

- Robust estimation can be achieved by different loss functions than quadratic

# Trimmed Least Squares

- A simple approach to robust least squares fitting is to first do an ordinary least squares fit, then identify the k data points with the largest residuals, omit these, perform the fit on the remaining data.



# Robust estimation

- Iteratively reweighted least squares (IRLS)
  - Estimates weights in every iteration to down-weigh outliers

$$\min_x W\|Ax - b\|$$

- Non-linear estimation with robust loss function (M-estimators)
  - A robust loss function is used that down-weighs the influence of outliers

$$\min_x \sum_i \rho_i(\|f_i(x_i)\|^2)$$

$$\min_x \|A(x) - b\|$$

# Non-linear estimation

$$\min_x \sum_i r_i^2$$

- Gauss-Newton method optimizes x by iteratively computing updates to x

$$x_{k+1}=x_k + d_k$$

- To compute updates the Jacobian of the error function has to be calculated

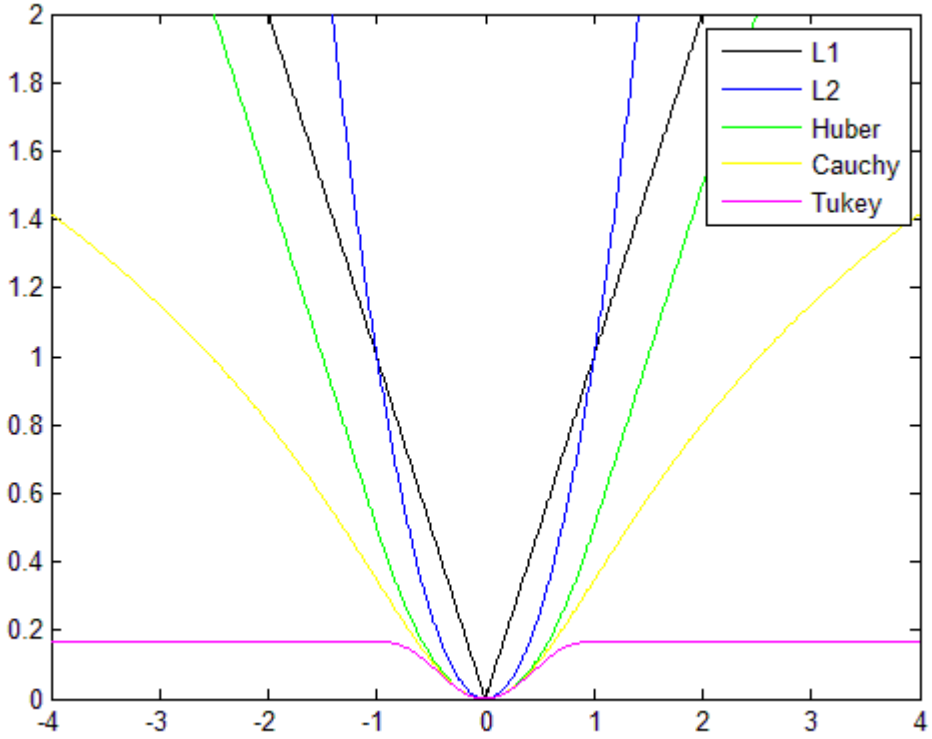
$$J_k^T J_k \; d_k \; = -J_k^T r_k$$

$$d_k \; = -(J_k^T J_k \;)^{-1} J_k^T r_k$$

# Non-linear estimation with robust loss function

$$\min_x \sum_i \rho_i(\|f_i(x_i)\|^2)$$

- Non-linear optimization (e.g. Levenberg-Marquard)
- Iteration necessary
- No explicit weight computation necessary
- Loss function should be differentiable
- Jacobian needs to be calculated



# Weighted Least-Squares

- Not all residuals are equally important. Weights define the importance.
- Weights are collected in diagonal weight matrix

$$\min_x W\|Ax - b\|$$
$$x = (A^T W A)^{-1} A^T W b$$

- Still one-step closed form solution.
- But, how to find the weights?

- Online estimation of the weights from robust cost function -> IRLS

# Iteratively reweighted least squares (IRLS)

1. Write the problem to solve as a weighted least squares optimization

$$C(x, w) = \sum_i w_i f_i(x)$$ cost function

2. Solve iteratively: At each step define weights

$$w_i^t = w_i(x^t)$$

and define update step

$$x^{t+1} = \underset{x}{\operatorname{argmin}}\, C(x, w^t)$$

$$= \underset{x}{\operatorname{argmin}} \sum_i w_i^t f_i(x) \qquad x = (A^T W A)^{-1} A^T W b$$

3. Hope that it converges to what you want

# How to choose the weights

- Weighted least squares minimizes the following cost

$$C(x, w) = \sum_i w_i f_i(x)$$

- We wish to minimize the cost with robustifier ρ

$$C_\rho(x) = \sum_i \rho(f_i(x))$$

- Minima of both cost functions need to be the same

$$\nabla C(x, w) = 0 \; if \; and \; only \; if \nabla C_\rho(x) = 0$$

$$\nabla w_i f_i(x) = \nabla \rho(f_i(x))$$
$$w_i \nabla f_i(x) = \rho'(f_i(x)) \nabla f_i(x)$$
$$w_i = \rho'(f_i(x))$$ required weights

# Example L1

- L1 norm means absolute distance

$$f_i(x) = d(x, y_i)^2$$
$$\rho(s) = \sqrt{s}$$

$$C_\rho(x) = \sum_i \rho(f_i(x)) = \sum_i |d(x, y_i)|$$ sum of absolute distances

- Weight computation:

$$w_i = \rho'\big(f_i(x)\big)$$
$$= \tfrac{1}{2} f(x)^{-1/2}$$
$$= \frac{1}{2} d(x, y_i)^{-1}$$

# Example Lq

- Lq norm means norm with power of q (q=1 .. L1, q = 2 .. L2)

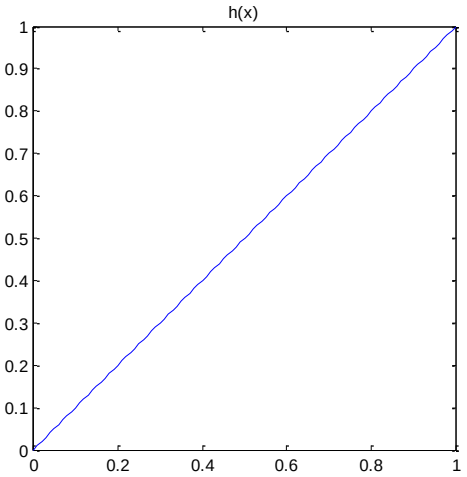
$$f_i(x) = d(x, y_i)^2$$
$$\rho(s) = s^{q/2}$$

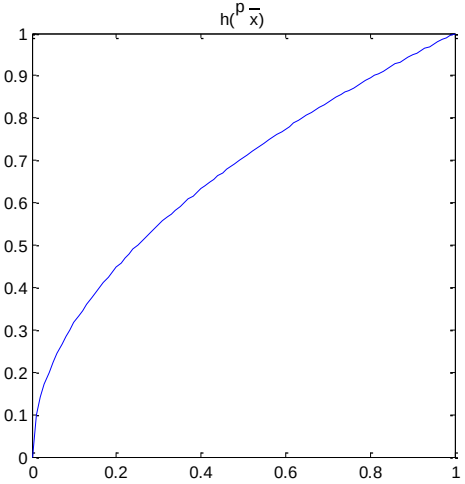
$$C_\rho(x) = \sum_i \rho(f_i(x)) = \sum_i d(x, y_i)^q$$

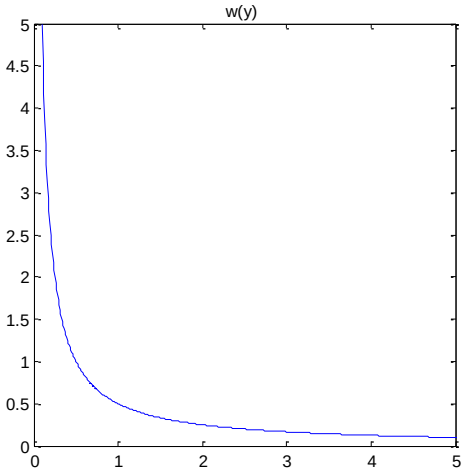
- Weight computation:

$$w_i = \rho'\big(f_i(x)\big)$$
$$= \frac{q}{2} f(x)^{(q-2)/2}$$
$$= \frac{q}{2} d(x, y_i)^{q-2}$$

# L1



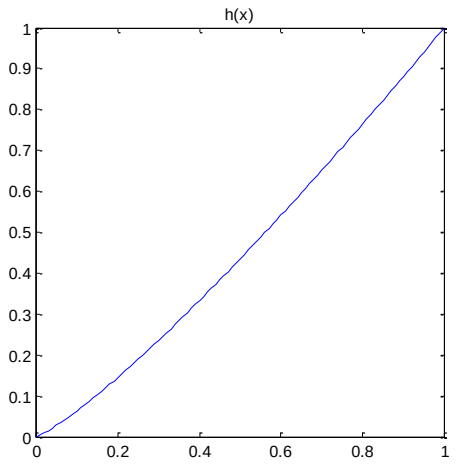



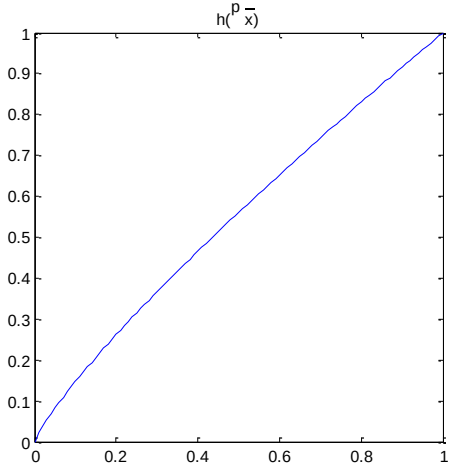

Advantage: Robust

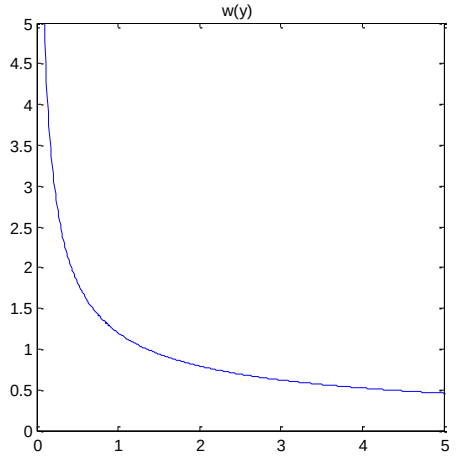
Disadvantage:

- Weights not defined at 0
- Can stop at non-minimum

# Lq



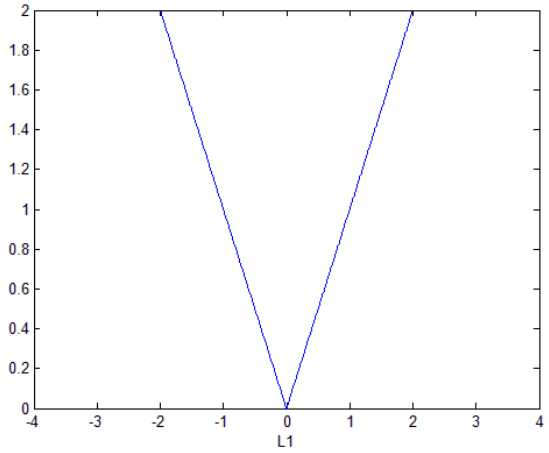

Advantage:

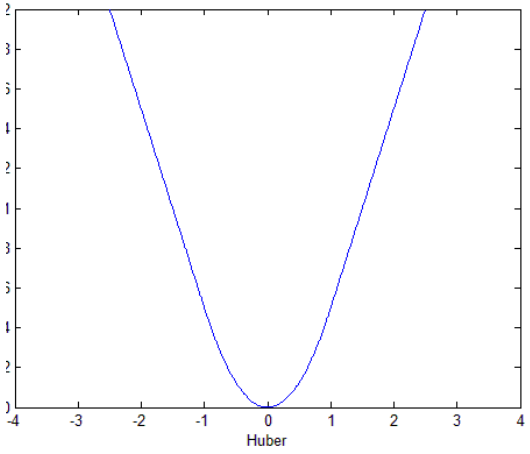
- Robust
- Cost function differentiable everywhere

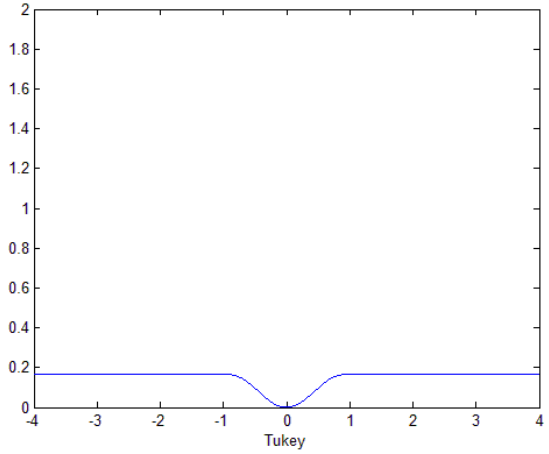
Disadvantage:

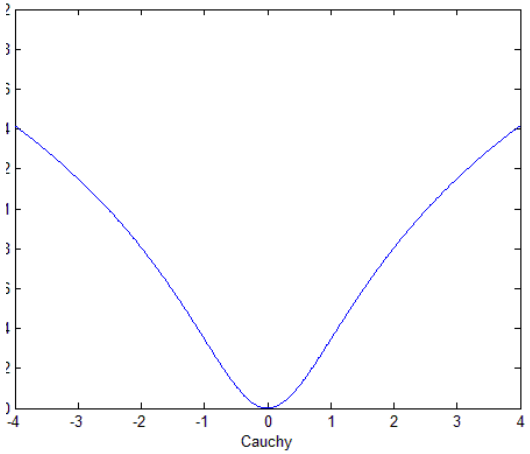
- Weights not defined at 0
- Can stop at non-minimum

# More robust loss functions



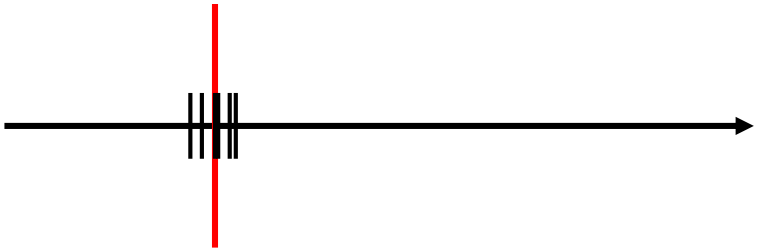




| loss function | p(x) |
|---|---|
| $L_1$ | $\lvert x\rvert$ |
| $Huber \begin{cases} if\ \lvert x\rvert \le k \\ if\ \lvert x\rvert > k \end{cases}$ | $\begin{cases} x^2/2 \\ k(\lvert x\rvert - \frac{k}{2}) \end{cases}$ |
| $Tukey \begin{cases} if\ \lvert x\rvert \le k \\ if\ \lvert x\rvert > k \end{cases}$ | $\begin{cases} k^2/6(1-\left(1-\left(\frac{x}{c}\right)^2\right)^3 \\ k^2/6 \end{cases}$ |
| $Cauchy$ | $\frac{k^2}{2}\log(1+(x/k)^2)$ |

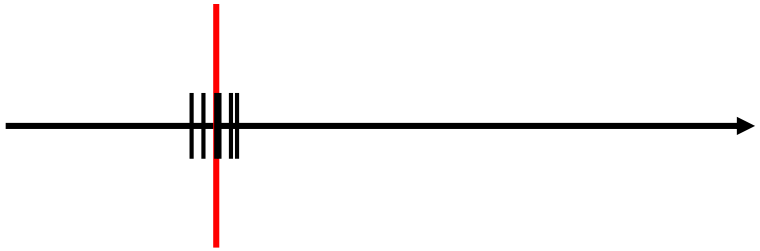
# Breakdown point

- Rate of measurement that can ruin the solution

Least-squares estimator
Mean



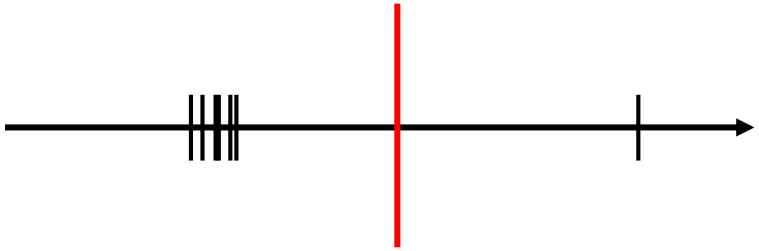
Robust estimator
Median



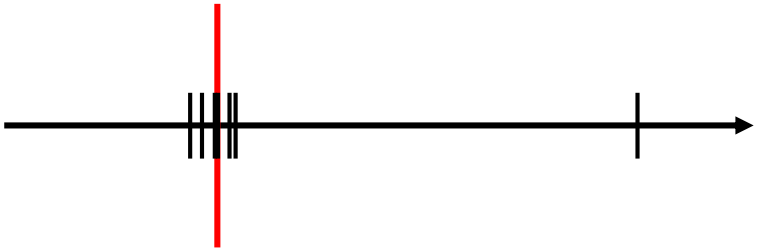
- No problem when there are no outliers

# Breakdown point

Least-squares estimator
Mean
Breakdown: 0%



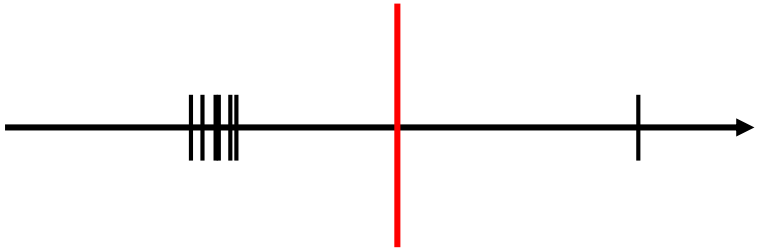
Robust estimator
Median



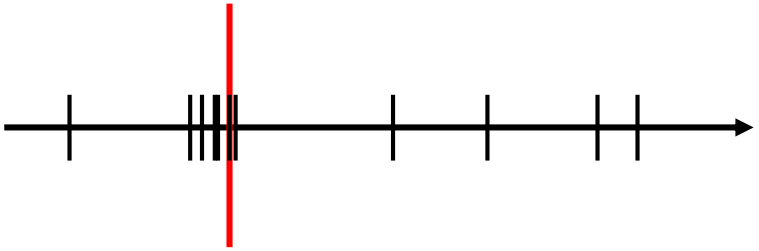
- LS breaks down when there is a single outlier

# Breakdown point

Least-squares estimator
Mean
Breakdown: 0%

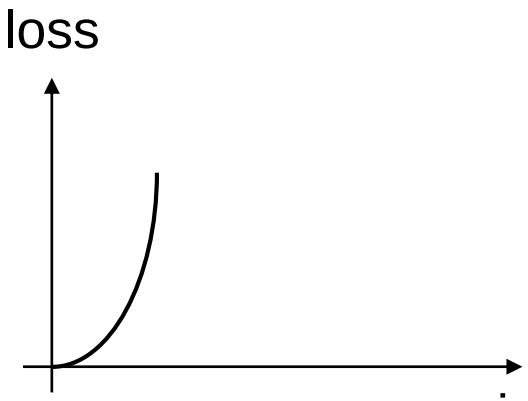


Robust estimator
Median
Breakdown: 50%



# Effects of different loss functions

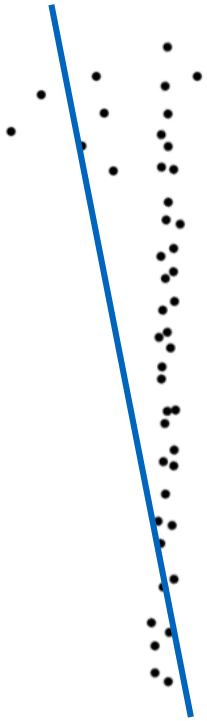
L2 solution



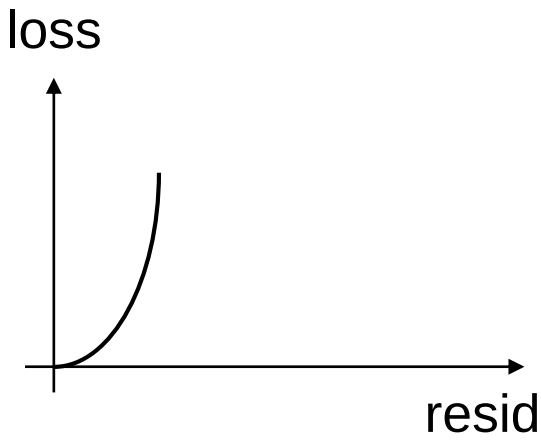

$$\min_x \sum_i r_i^2(x)$$

- L2 (quadratic) loss works fine for outlier-free data

# Effects of different loss functions



L2 solution

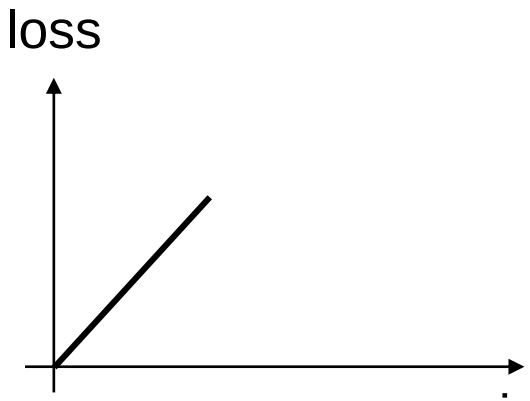



$$\min_x \sum_i r_i^2(x)$$

- Outliers lead to wrong estimate

# Effects of different loss functions

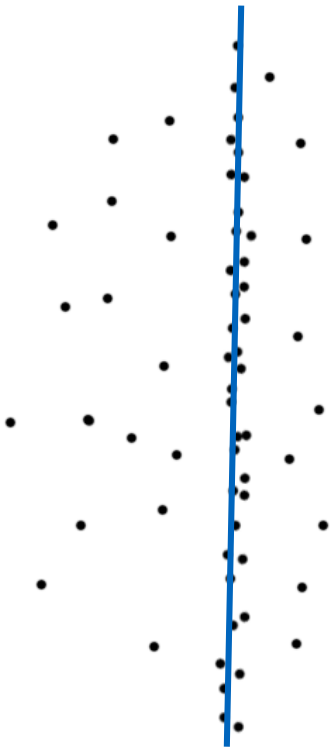
L1 solution



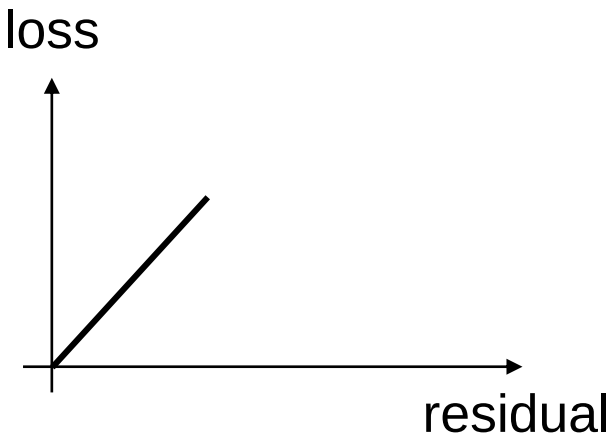

$$\min_x \sum_i |r_i(x)|$$

- L1 loss leads to correct solution (estimation robust to outliers)

# Effects of different loss functions



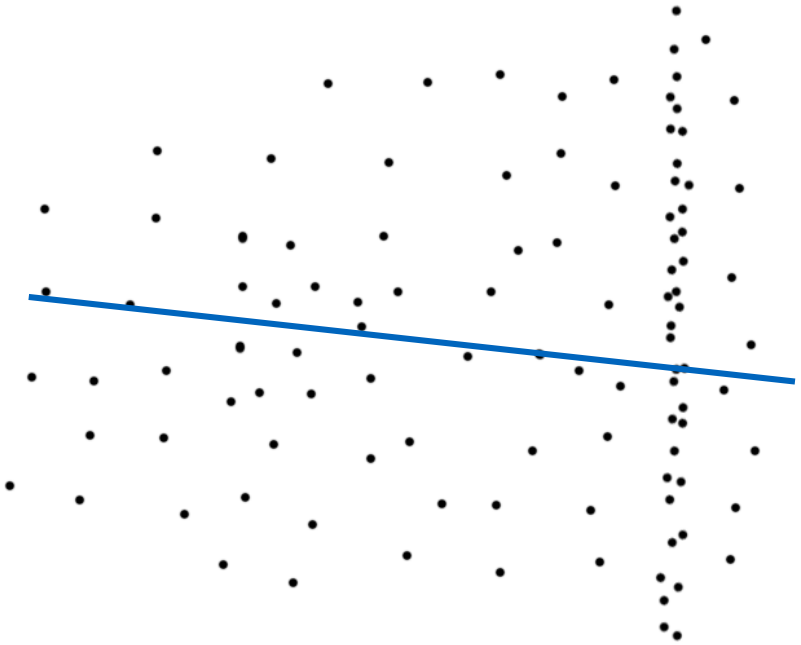
L1 solution



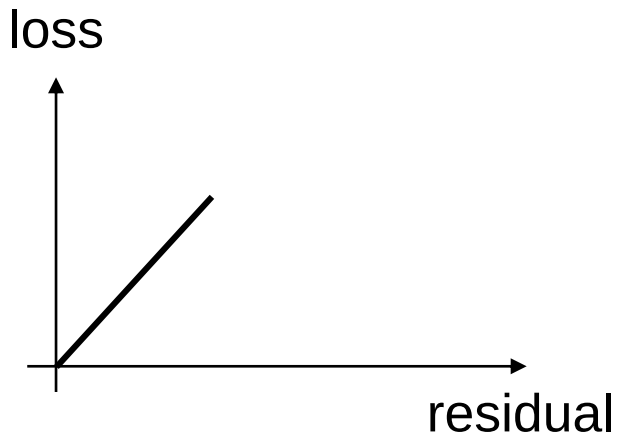

$$\min_x \sum_i |r_i(x)|$$

- Does work for larger number of outliers as well

# Effects of different loss functions



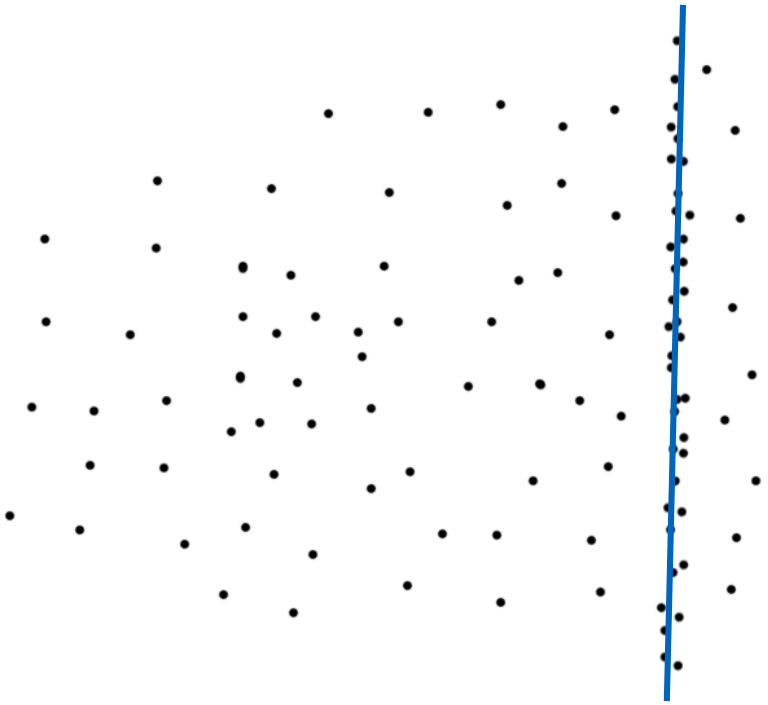
L1 solution

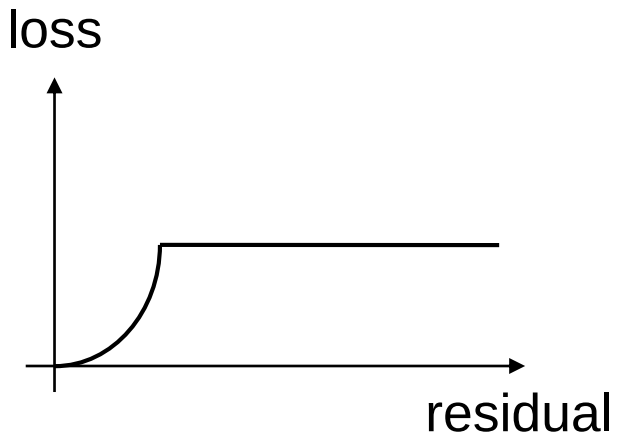



$$\min_x \sum_i |r_i(x)|$$

- Breaks down eventually (> 50% outliers)

# Effects of different loss functions






$$\min_x \sum_i \min(r_i^2(x), t^2)$$

- However, can be solved with different cost function (e.g. Truncated L2)
- However, not easy to predict this behaviour

# Rotation averaging

- Given rotations Ri element of SO(3) we are looking for the Lq mean defined as:

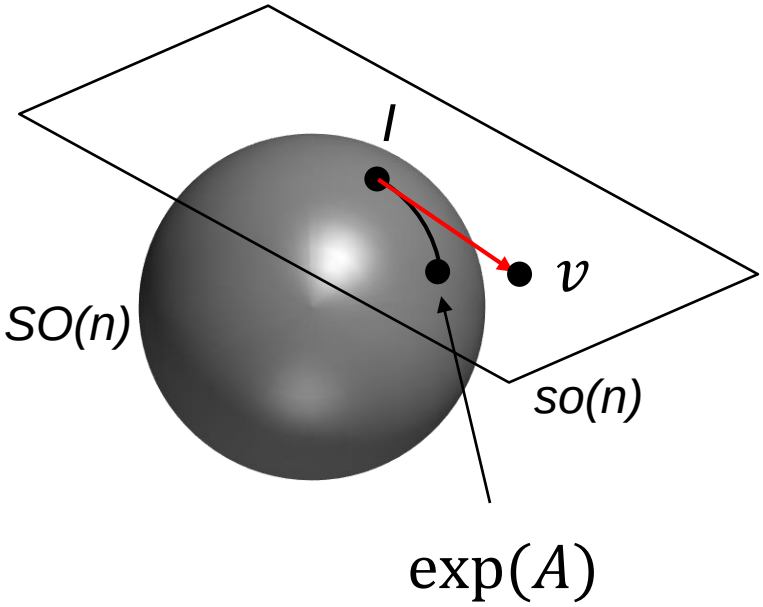
$$S^* = \underset{S \in SO(3)}{arg\min} \sum_i d(S, R_i)^q$$

- q=2: Standard least-squares L2 averaging. Not robust to outliers
- q=1: L1 averaging. Robust to outliers

- Problem: How to measure the distance d(S,Ri)?
  - Angular distance $d_{ang}(R,S)$
  - Quaternion distance min(||r-s||,||r+s||)
  - Chordal distance $||R-S||_F$ (Frobenius norm)
- Best: Optimization on the manifold

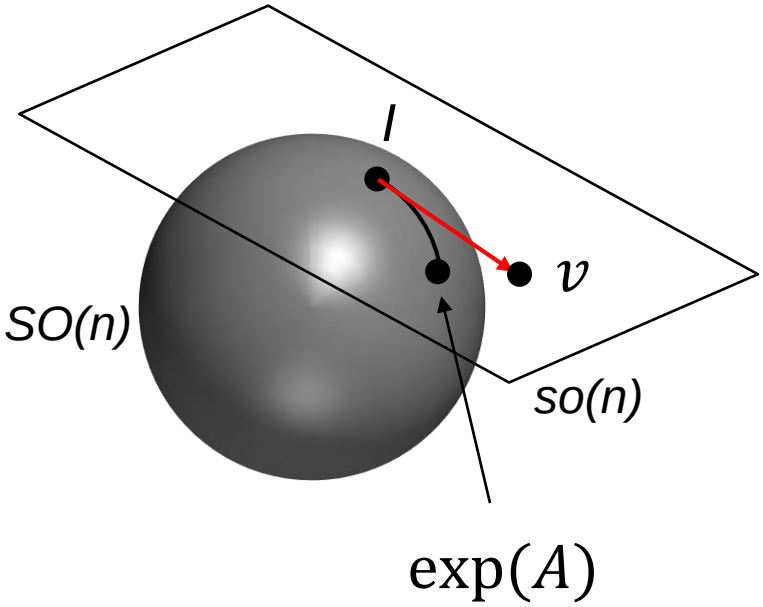
# Iteratively reweighted least squares (IRLS) for rotation averaging

- Averaging step is linear least squares
- For robust estimation IRLS can be used
- However, optimization needs to be done on the manifold (use of tangent space)
- Idea: Map back and forth from the manifold to the tangent space using the exponential and logarithm maps



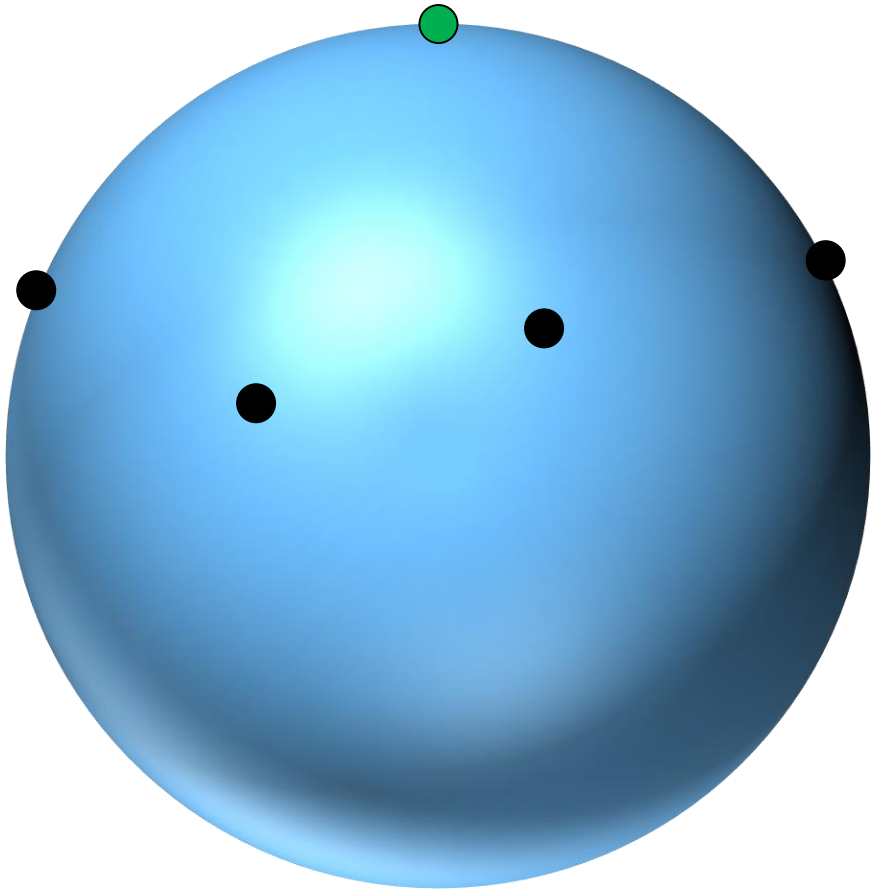
# Iteratively reweighted least squares (IRLS) for rotation averaging

1. Apply the logarithm map to the rotations Ri to map them onto the tangent space vi using some initial estimate for the average R*
2. Compute weights for robust loss function
3. Solve weighted linear least squares problem to get average in tangent space v*
4. Use exponential map to compute R* from v*
5. Repeat from 1 until convergence



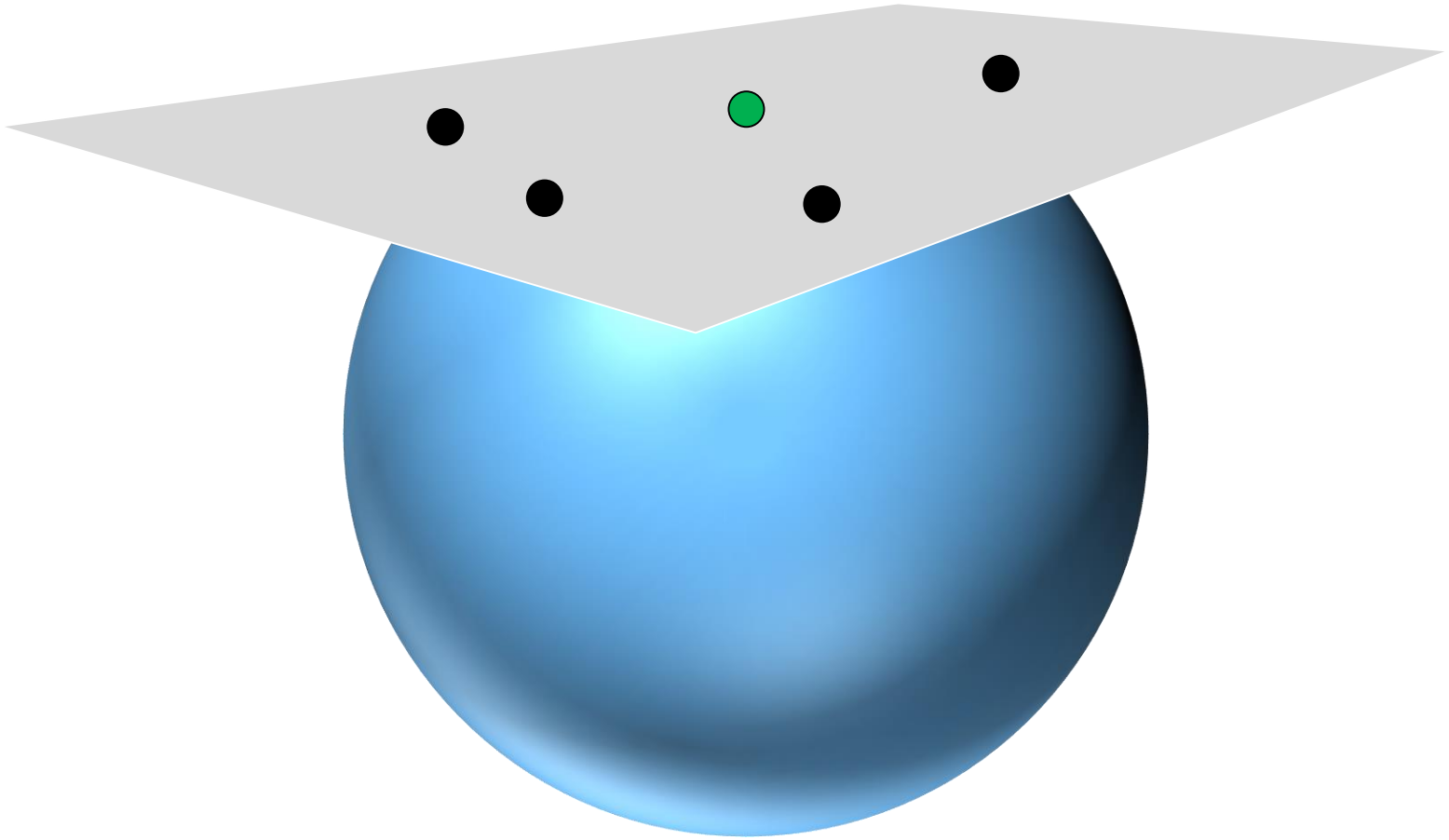
# Illustration

- Start with some initial point for first projection into tangent space



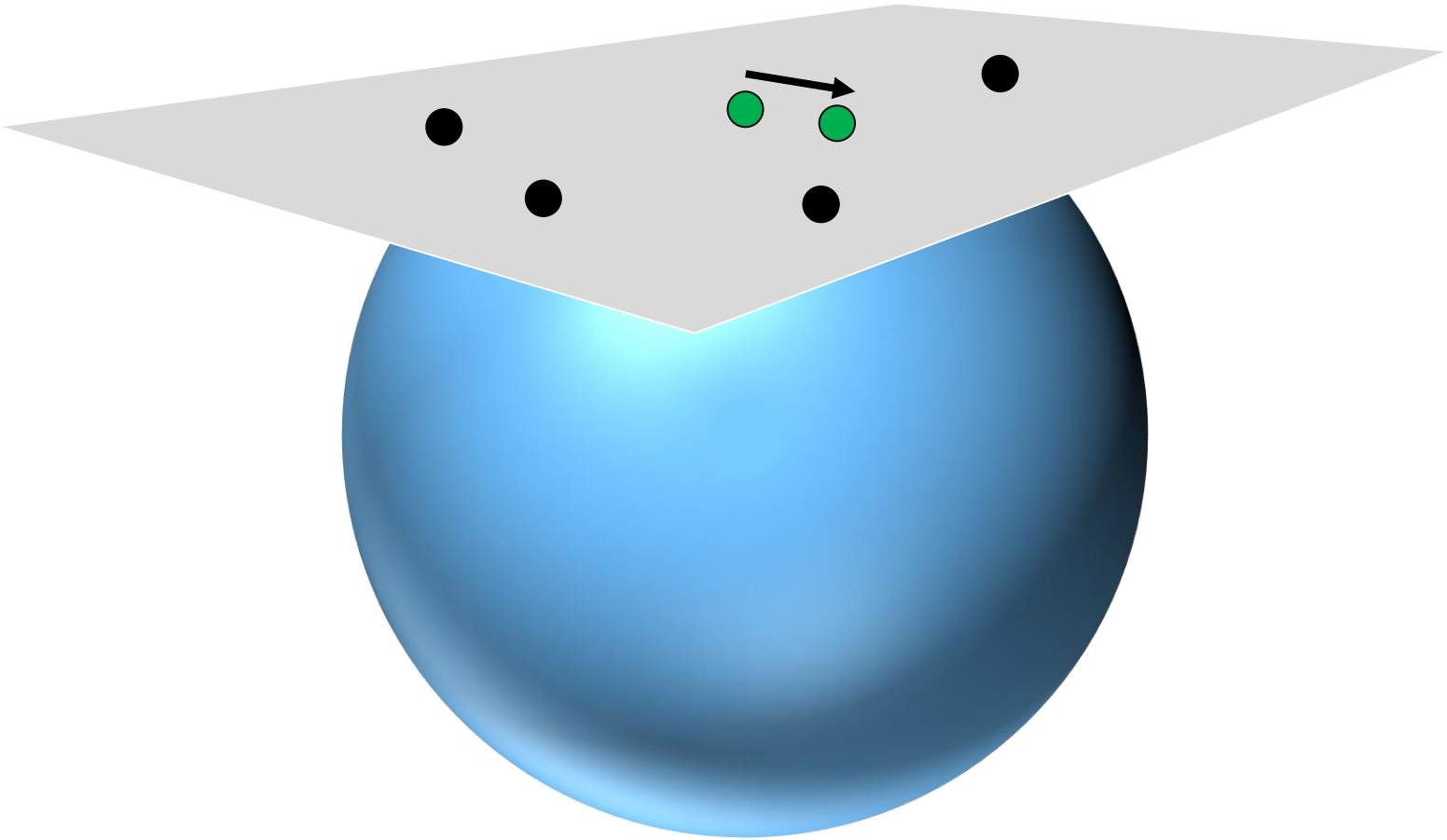
# Illustration

- Project the points to tangent space



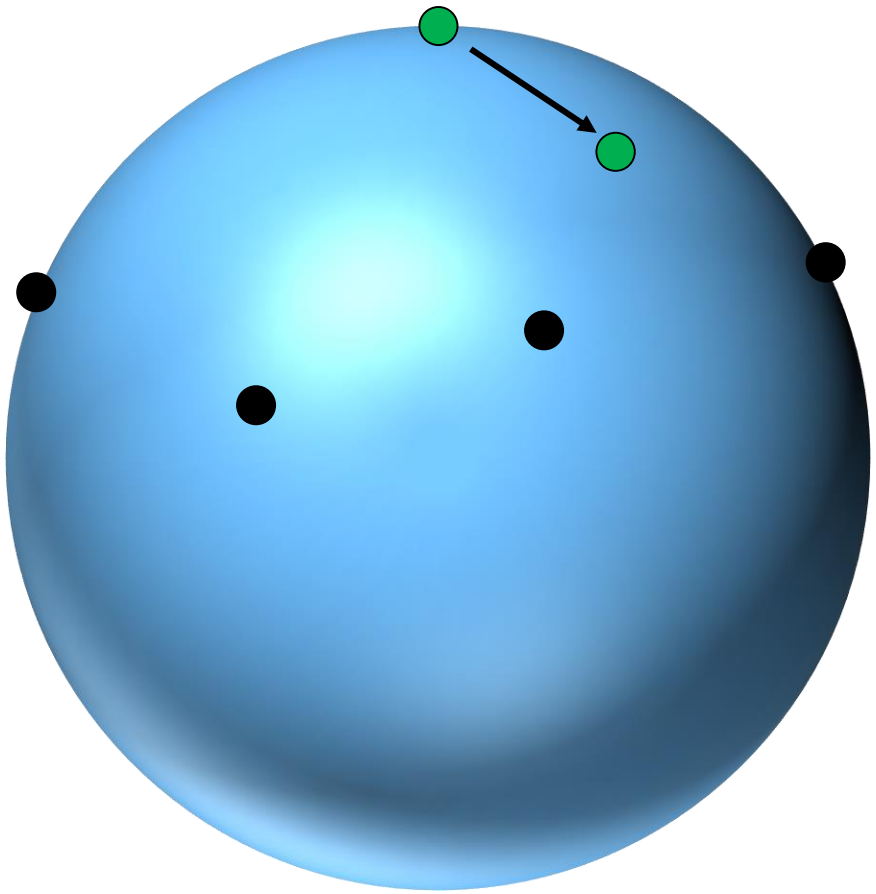
# Illustration

- Perform IRLS optimization step in tangent space



# Illustration

- Project the updated mean onto the manifold (exponential map)



# Illustration

- Create new projection to  tangent space from updated mean (will be different tangent space than in the previous step)
- Repeat all the steps until convergence