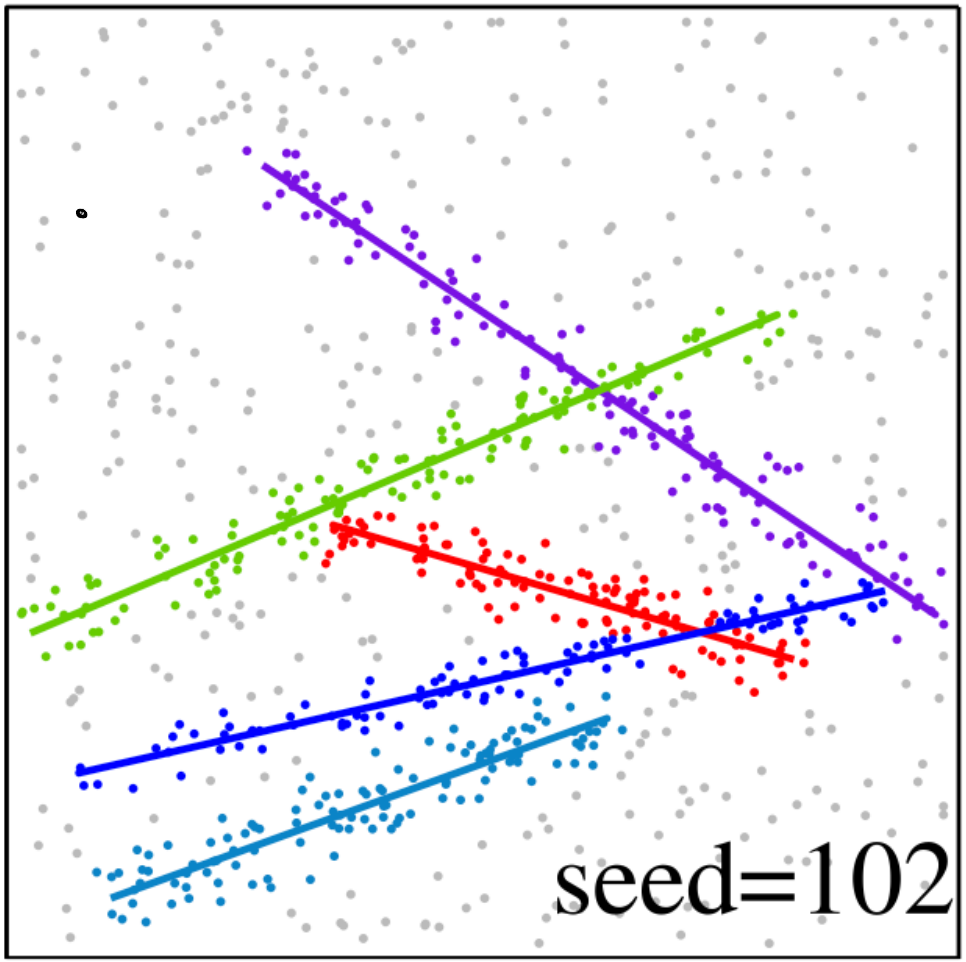
# Topic 01:

# Robust Estimation Basics

- least squares & total least squares
- robust M-estimation
- optimizing robust objective functions
- proposal generation: Hough transforms
- proposal generation: RANSAC

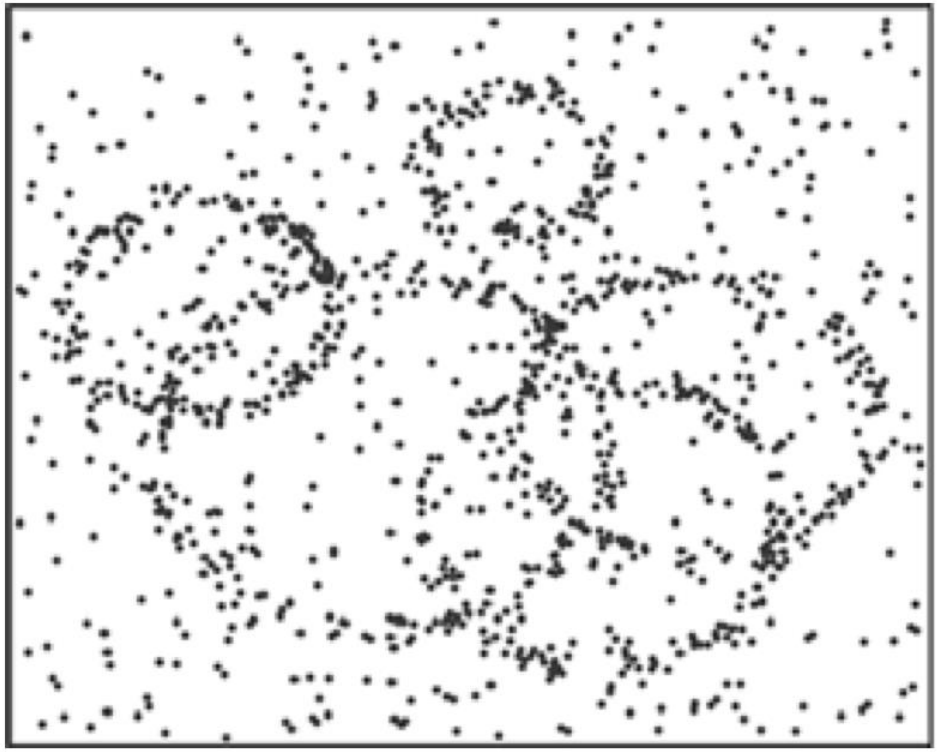
# motivation

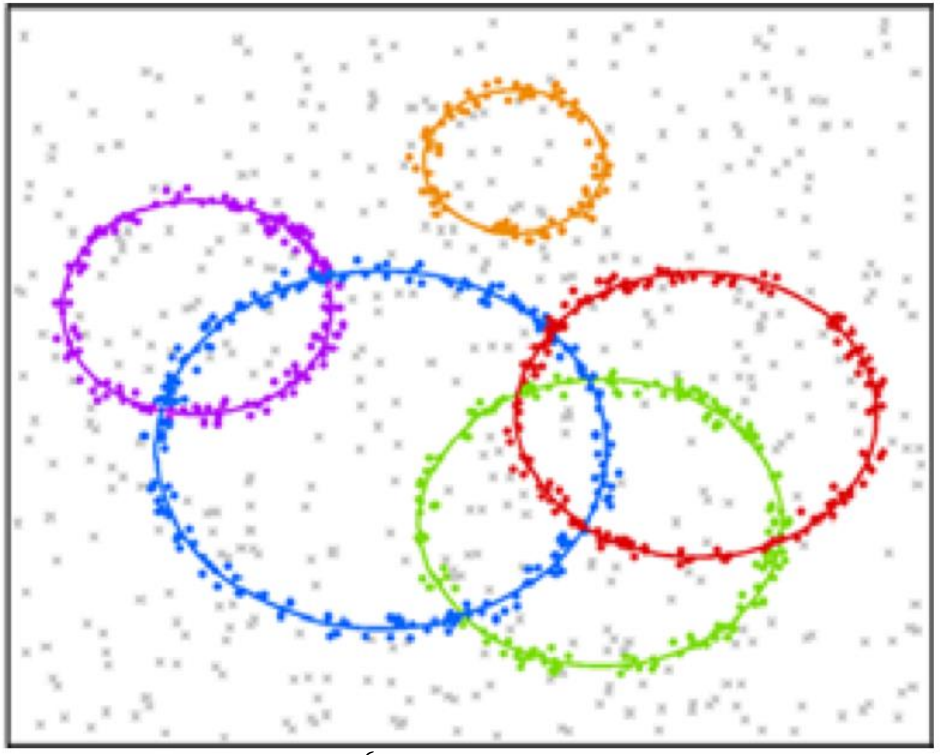
Our task will be to fit (geometric) models to noisy data




(Delong et al, IJCV12)

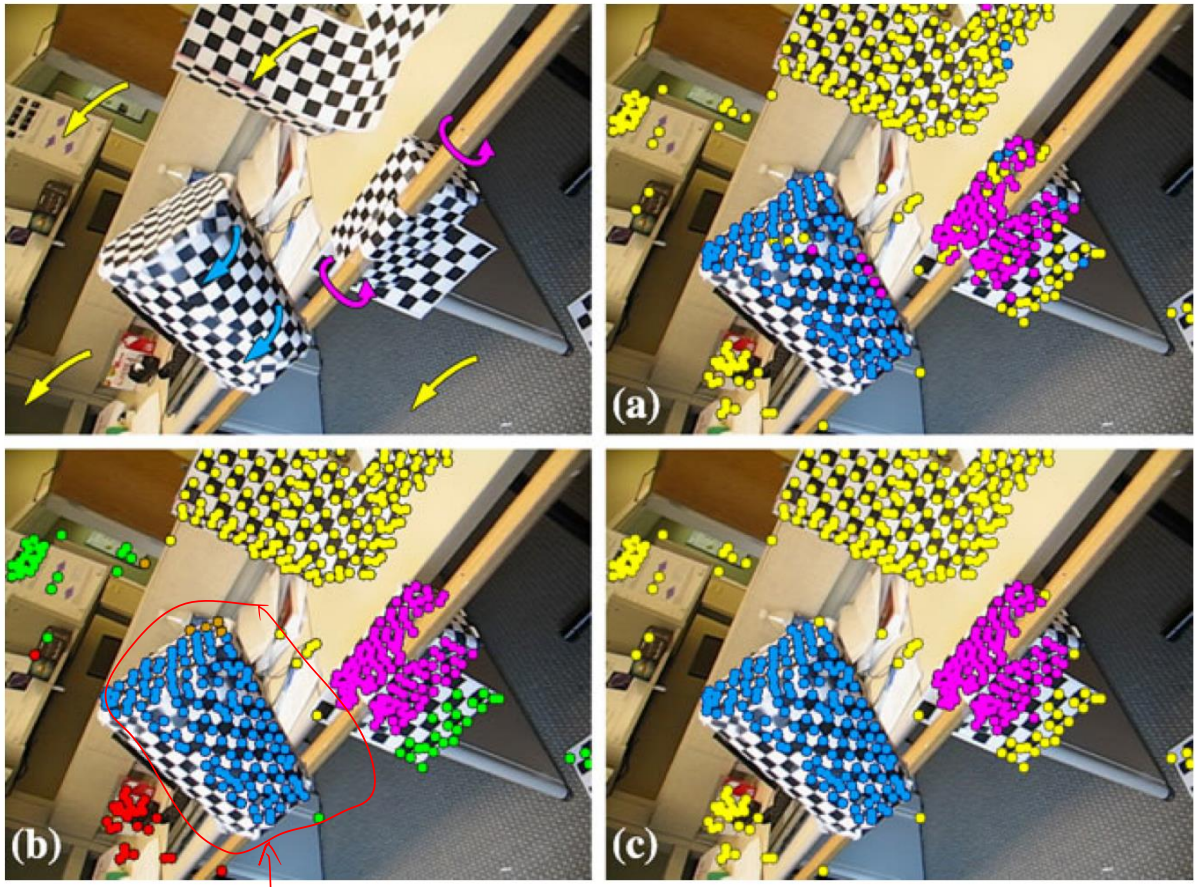
# motivation





(Delong et al, IJCV12)

# motivation

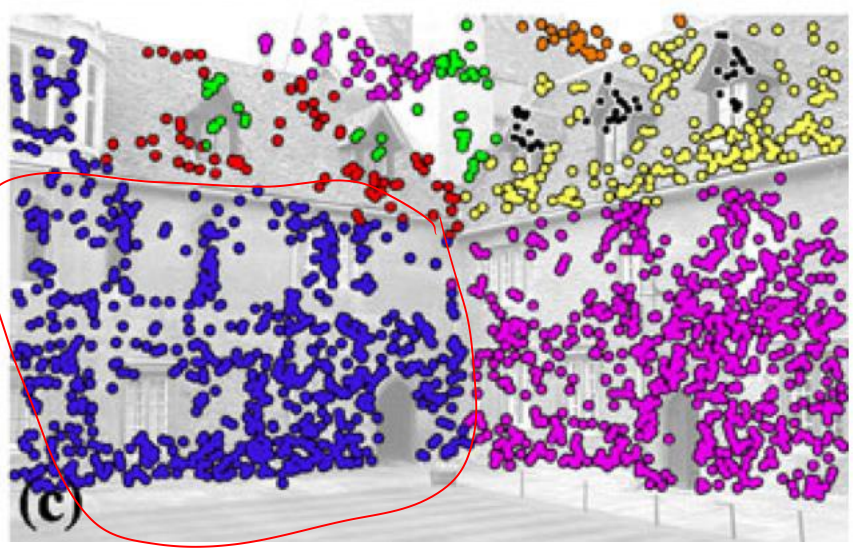



(Delong et al, IJCV12)

Points consistent with rigid-body motion

# motivation

(Delong et al, IJCV12)



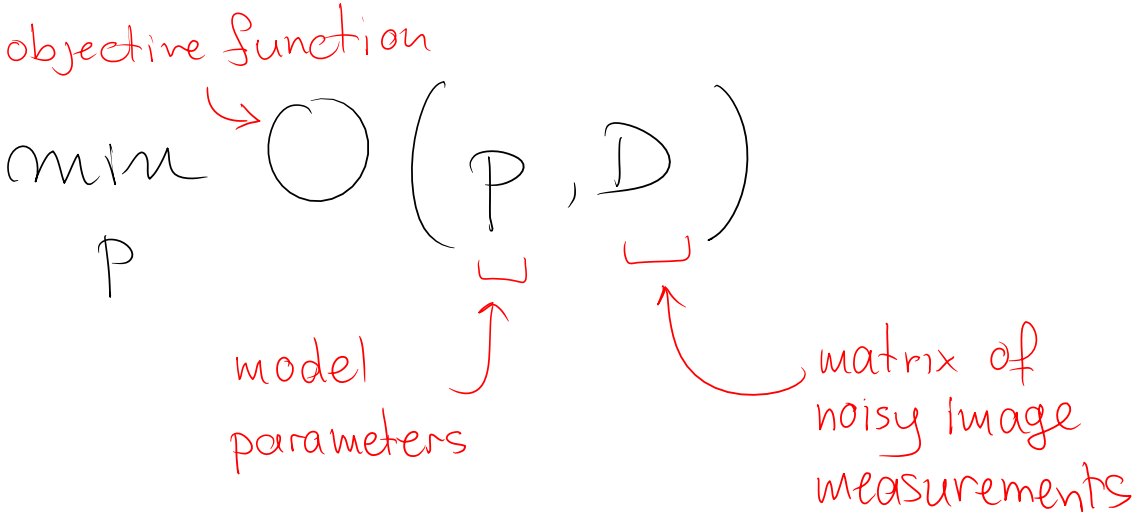

points consistent with a single plane

# motivation

**Introduction.** A common computational problem in vision is to estimate the parameters of a model from image data.

Examples of parameterized models to be fit to image data include lines and ellipses, camera calibration models, image motion models, 3D planar regions, 3D models, and human face models.

Top-level objective has the form



# motivation

Top-level objective has the form

objective function

$$\min_{P} O(P, D)$$

model parameters

matrix of noisy image measurements

**Key Difficulties:**

- The models must be fit to noisy image data.
- Initial guesses for the models must be generated automatically.
- Multiple occurrences of the models are often represented in the data. But the number and types of models are typically unknown a priori. They must also be determined from the data.
- The data typically contains (structured) outliers, i.e., observations that do not belong to the model being fitted. These must somehow be ignored during the model fitting.

# motivation

measurements:

$$\vec{x}_k = \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix}$$

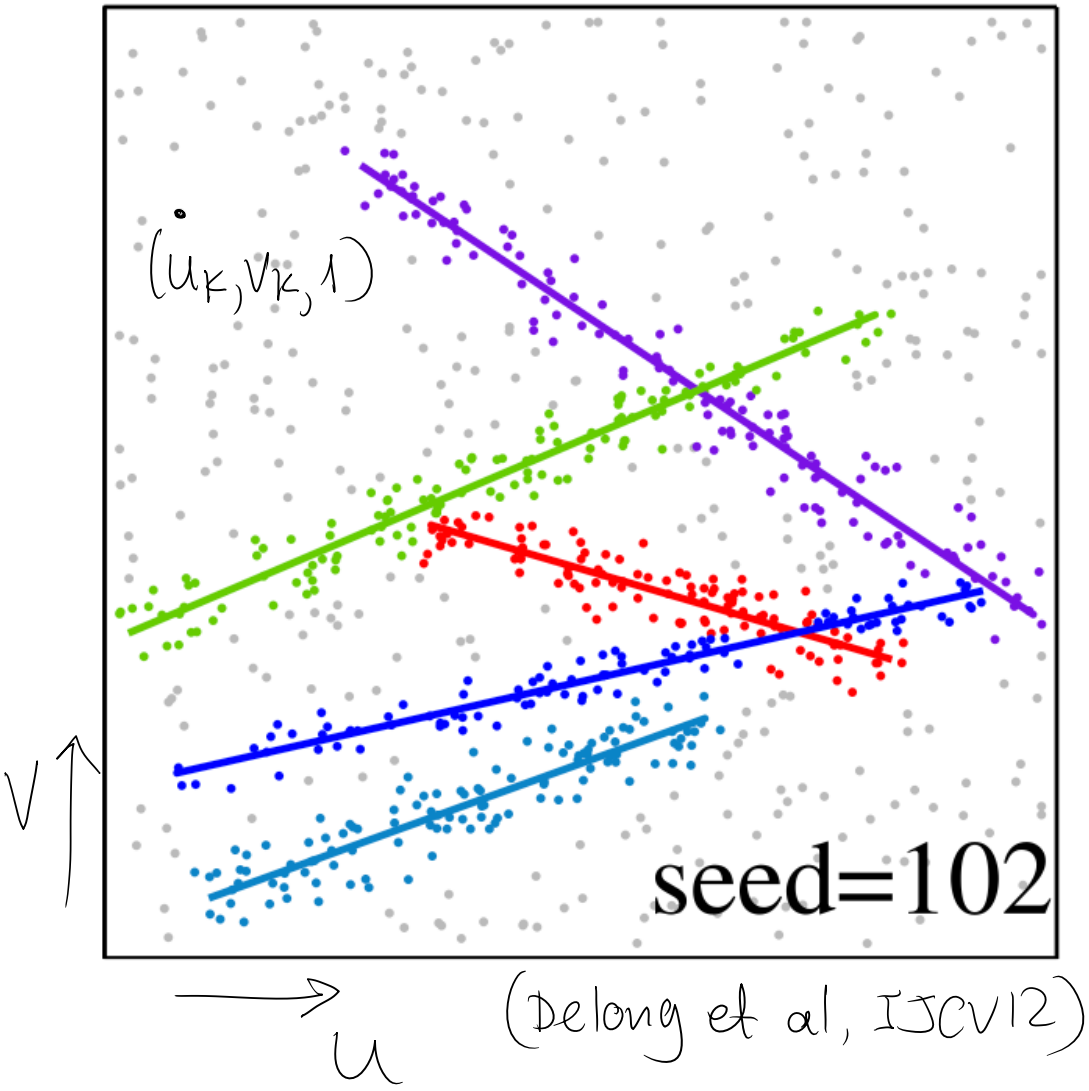
$$D = \begin{bmatrix} \vec{x}_1 & \vec{x}_2 & \cdots & \vec{x}_k \end{bmatrix}$$

parameters:

$$p = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

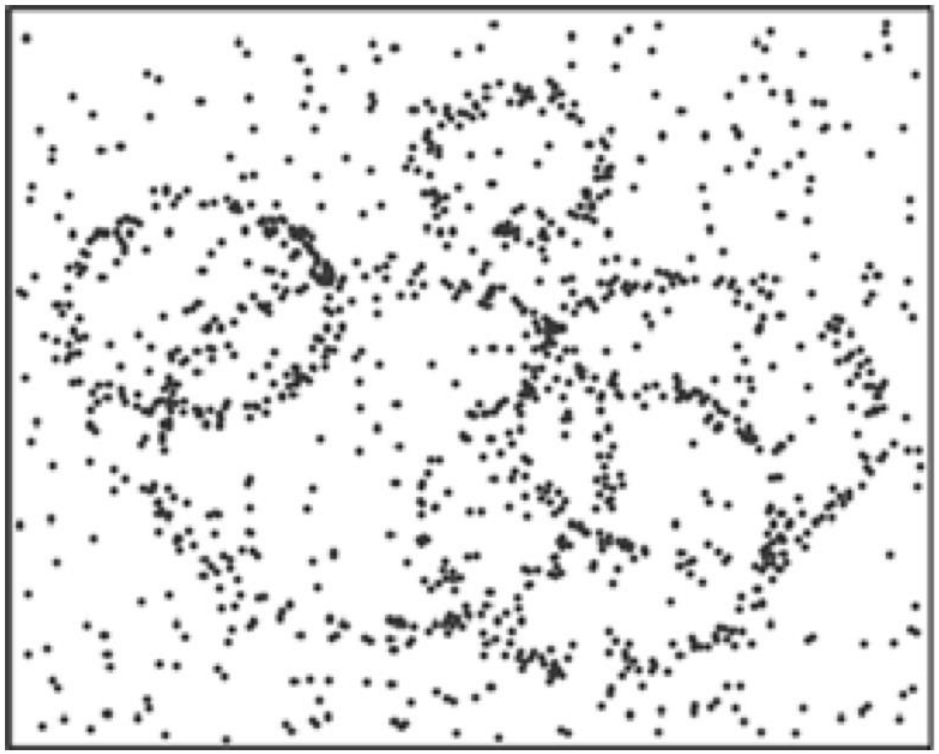
line equation:

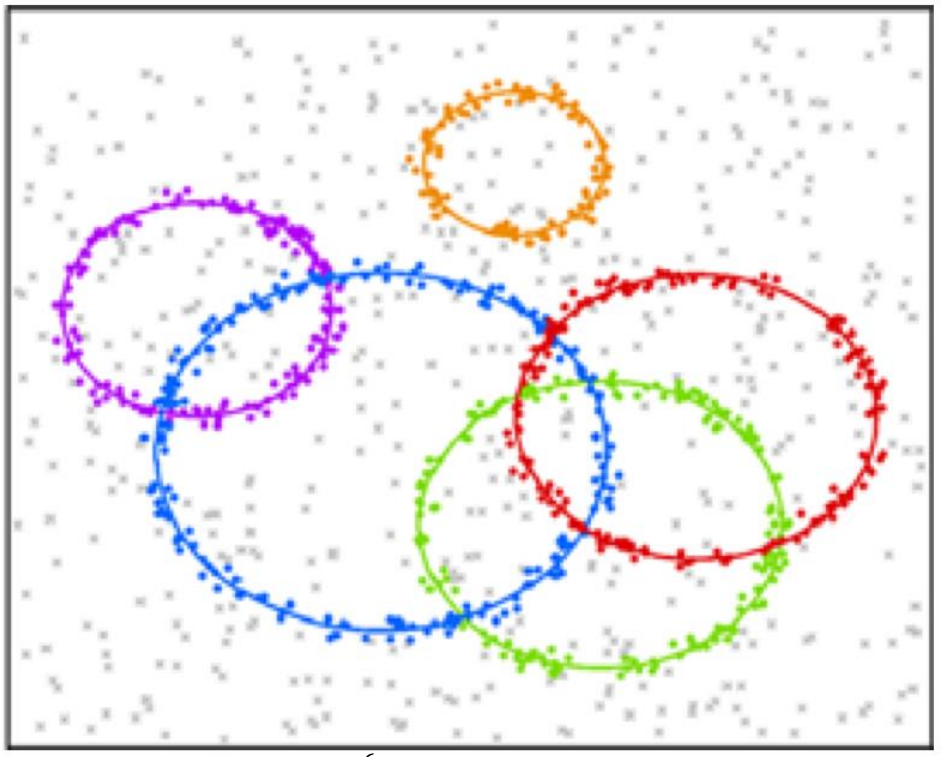
$$p^T \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} = 0$$




(Delong et al, IJCV12)

# motivation





(Delong et al, IJCV12)

measurements:

$$D = \begin{bmatrix} u_1 & u_2 & \cdots & u_k \\ v_1 & v_2 & \cdots & v_k \\ 1 & 1 & & 1 \end{bmatrix}$$

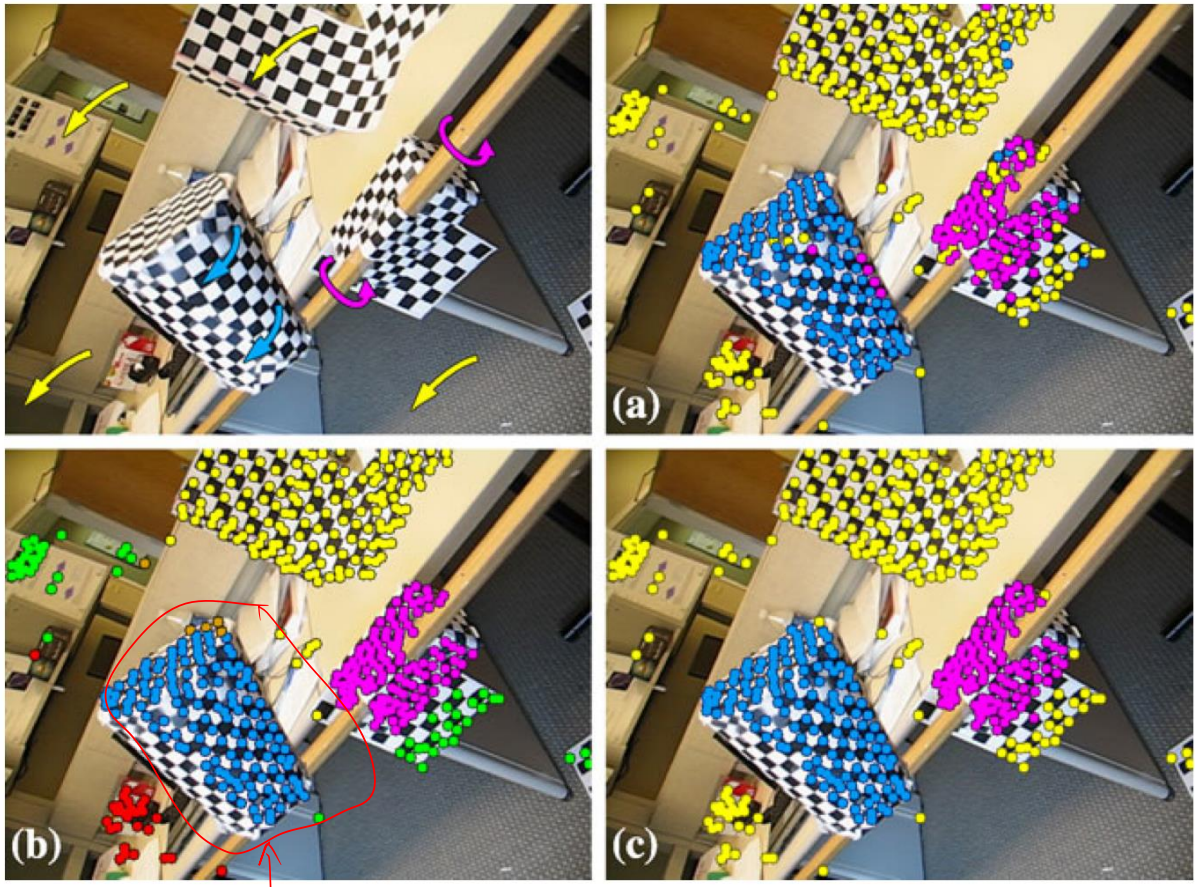
parameters:

$$p = [a\ b\ c\ d\ e\ f]$$

conic equation:

$$[u_k\ v_k\ 1] \begin{bmatrix} a & d & e \\ d & b & f \\ e & f & c \end{bmatrix} \begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} = 0$$

# motivation

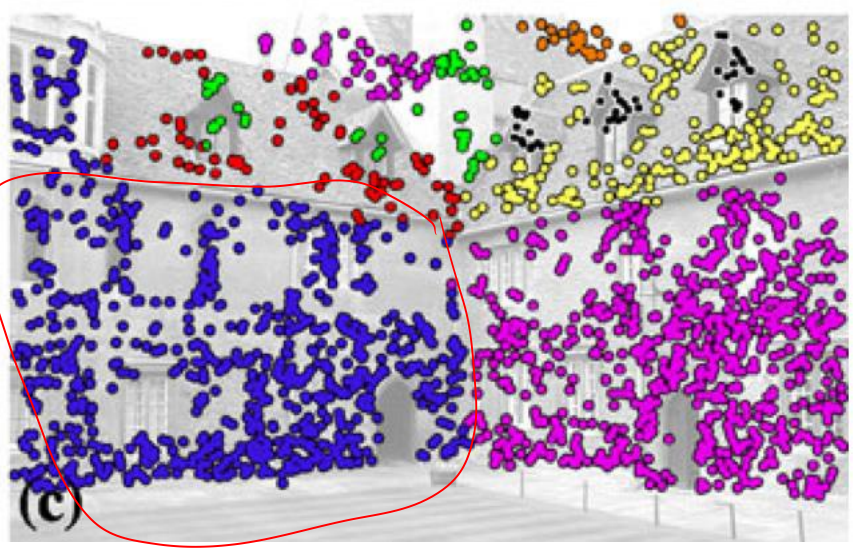



(Delong et al, IJCV12)

Points consistent with rigid-body motion

# motivation

(Delong et al, IJCV12)



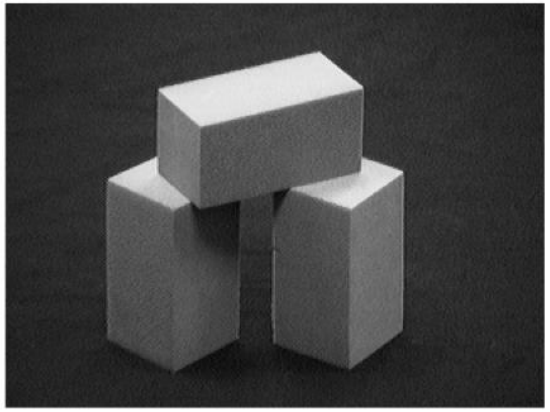

points consistent with single plane homography

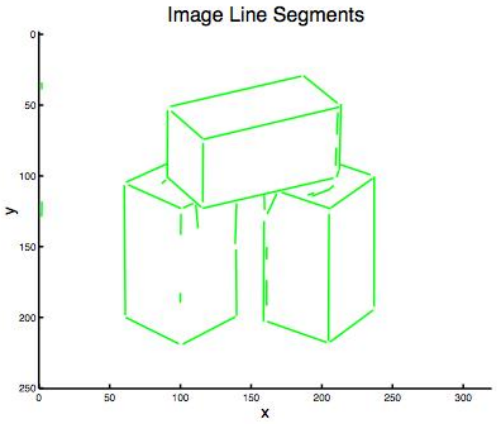
# working example

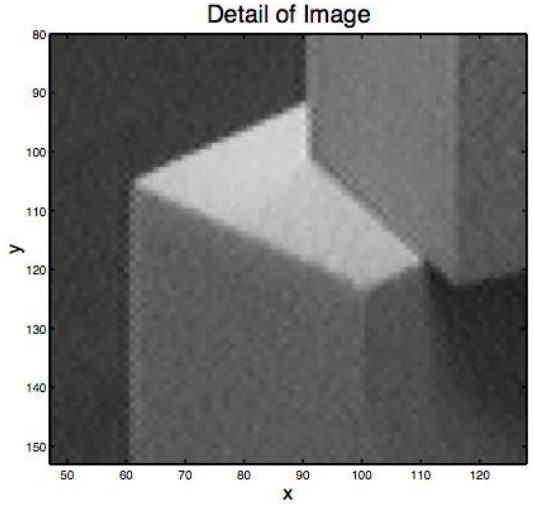
**Example Problem:** Find the best fitting line(s) to a set of image edgel positions,

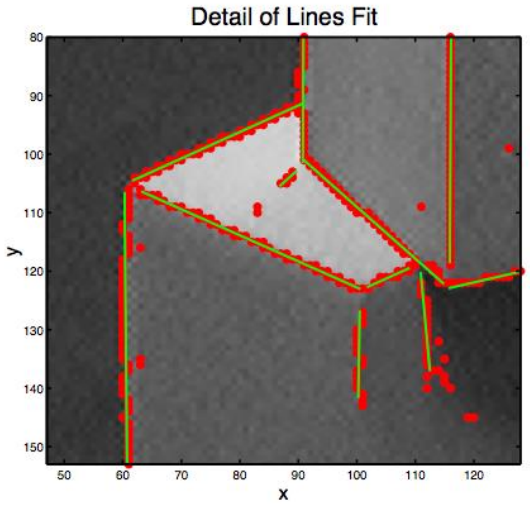
$$D = \{\vec{x}_k\}_{k=1}^{K}.$$

For simplicity, we ignore edgel strengths and orientations here.

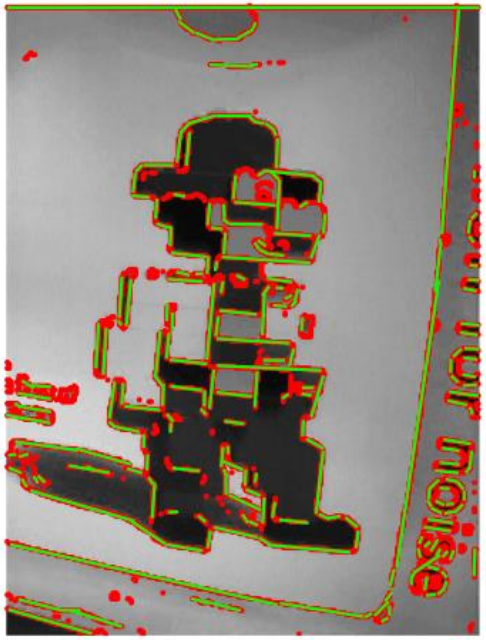


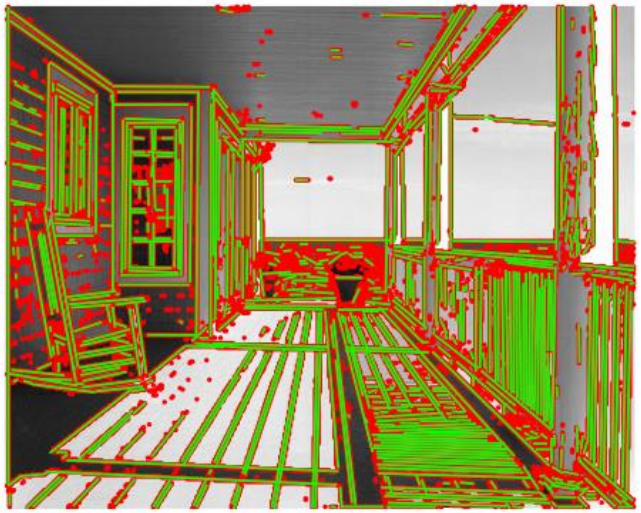


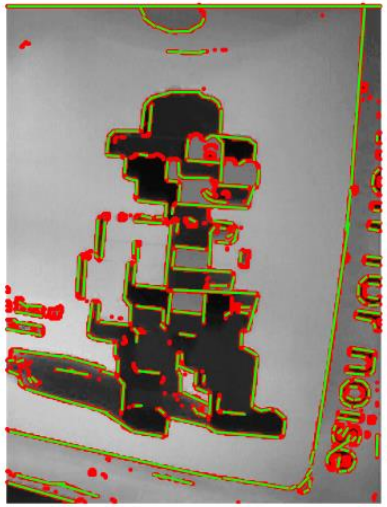
# example results

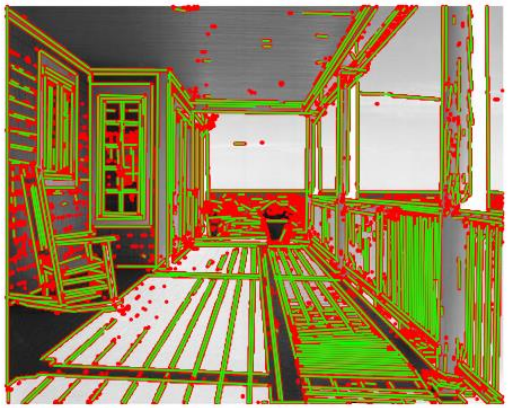



# big picture: robust line estimation algorithm

1. Initial guess
   randomly sample the data
2. Iterative fitting (single line)
   TBD
3. Verification
   decide if line has sufficient support
4. Model selection
   associate edgels to the computed line fit

# goals for today

We use the example of estimating image lines to:

- Introduce robust M-estimation and show how it deals with outliers.
- Discuss leverage points.
- Introduce methods for generating initial guesses.
- Introduce issues in model selection.
  - We have more to say about this difficult issue later in this course. Here we briefly consider selecting the number of models (i.e., lines), but not the model type (eg. lines versus curves).
- Discuss the general types of errors to be expected.

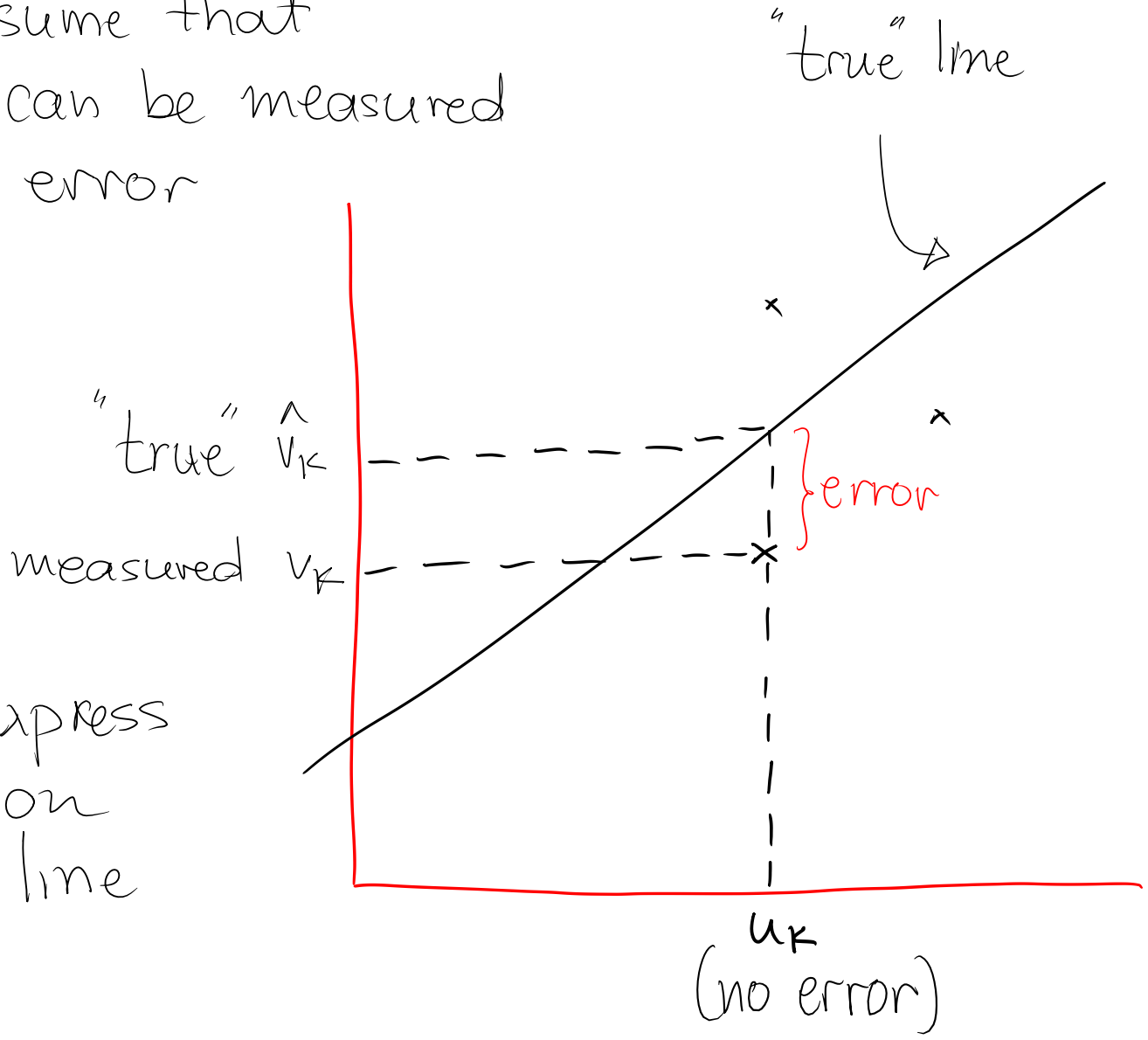
# Topic 01:

# Robust Estimation Basics

- least squares & total least squares
- robust M-estimators
- optimizing robust objective functions
- proposal generation: Hough transforms
- proposal generation: RANSAC

# least squares estimation (LS)

let us first assume that u-coordinates can be measured without <u>any</u> error




how can we express our optimization objective for line fitting?

# least squares estimation (LS)

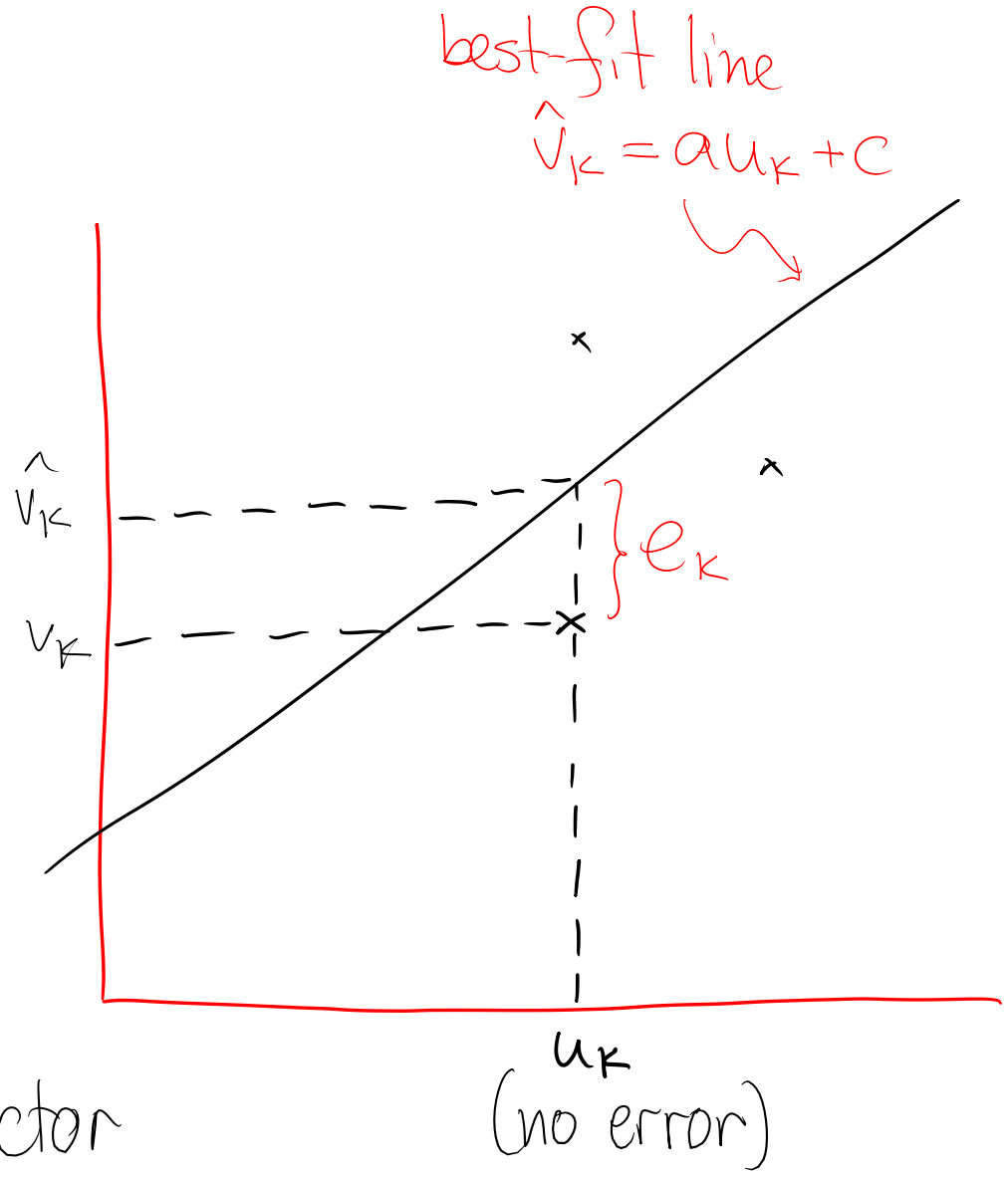
residual error:

$$e_k = v_k - au_k - c$$

$$= v_k - \begin{bmatrix} u_k & 1 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix}$$

vector of residuals:

$$e = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix} - \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_k & 1 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix}$$




objective:

find line with smallest residual vector

# least squares estimation: objective function

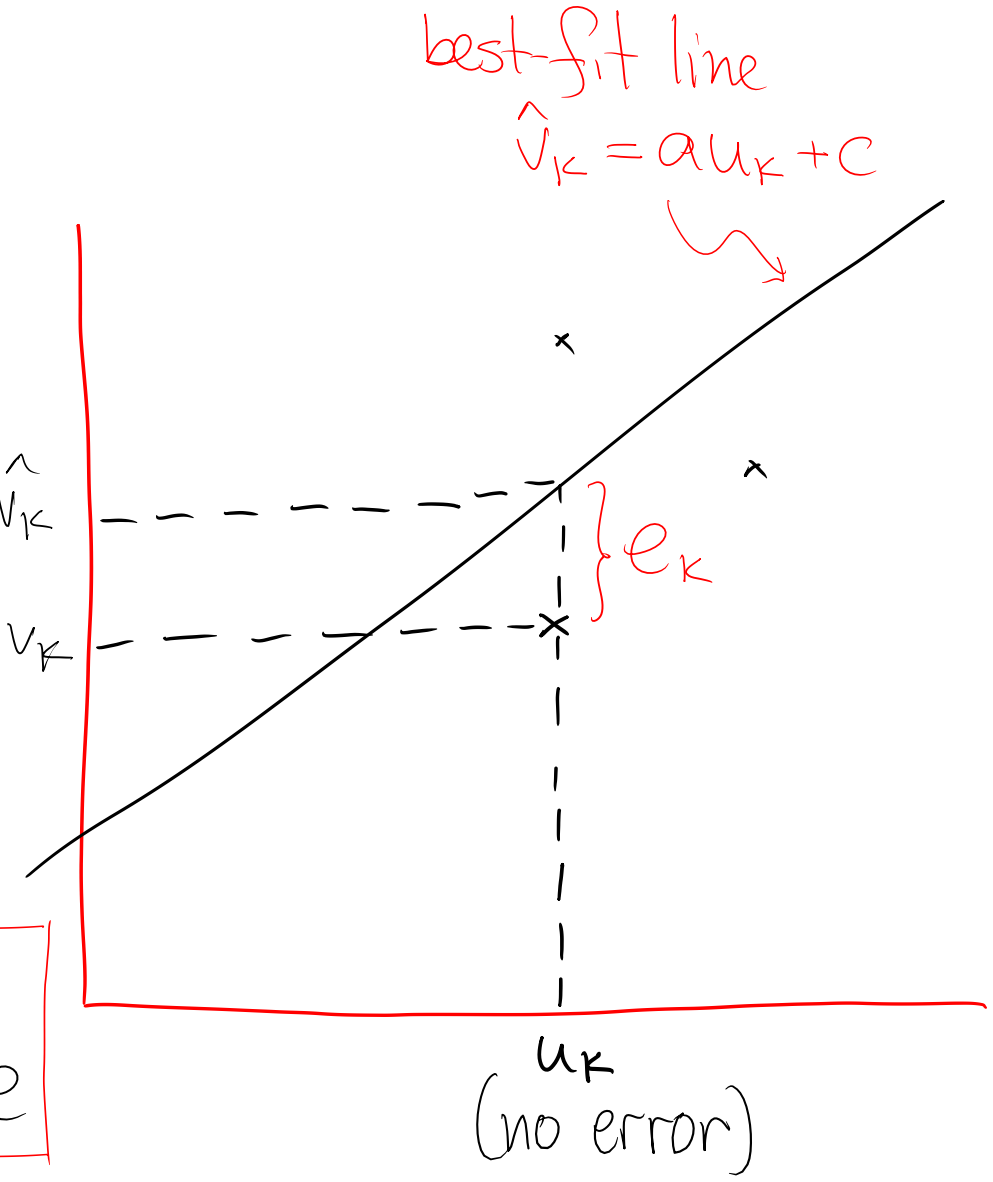
residual error:

$$e_k = v_k - au_k - c$$

$$= v_k - \begin{bmatrix} u_k & 1 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix}$$

vector of residuals:

$$e = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix} - \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_k & 1 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix}$$




Goal: minimize objective $O(a,c) = e^T e$

# least squares estimation: line fitting solution

residual error:

$$e_k = v_k - au_k - c$$

$$= v_k - \begin{bmatrix} u_k & 1 \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix}$$

vector of residuals:

$$e = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix} - \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_k & 1 \end{bmatrix} \underbrace{\begin{bmatrix} a \\ c \end{bmatrix}}_{p}$$

$\underbrace{\quad}_{v}$ $\underbrace{\quad}_{U}$

Goal: minimize objective $O(a,c) = e^T e$

$$O(p) = (v - Up)^T (v - Up)$$

$$= v^T v - 2v^T Up + p^T U^T Up$$

differentiate wrt $p$:

$$\frac{dO}{dp} = -2v^T U + 2p^T U^T U$$

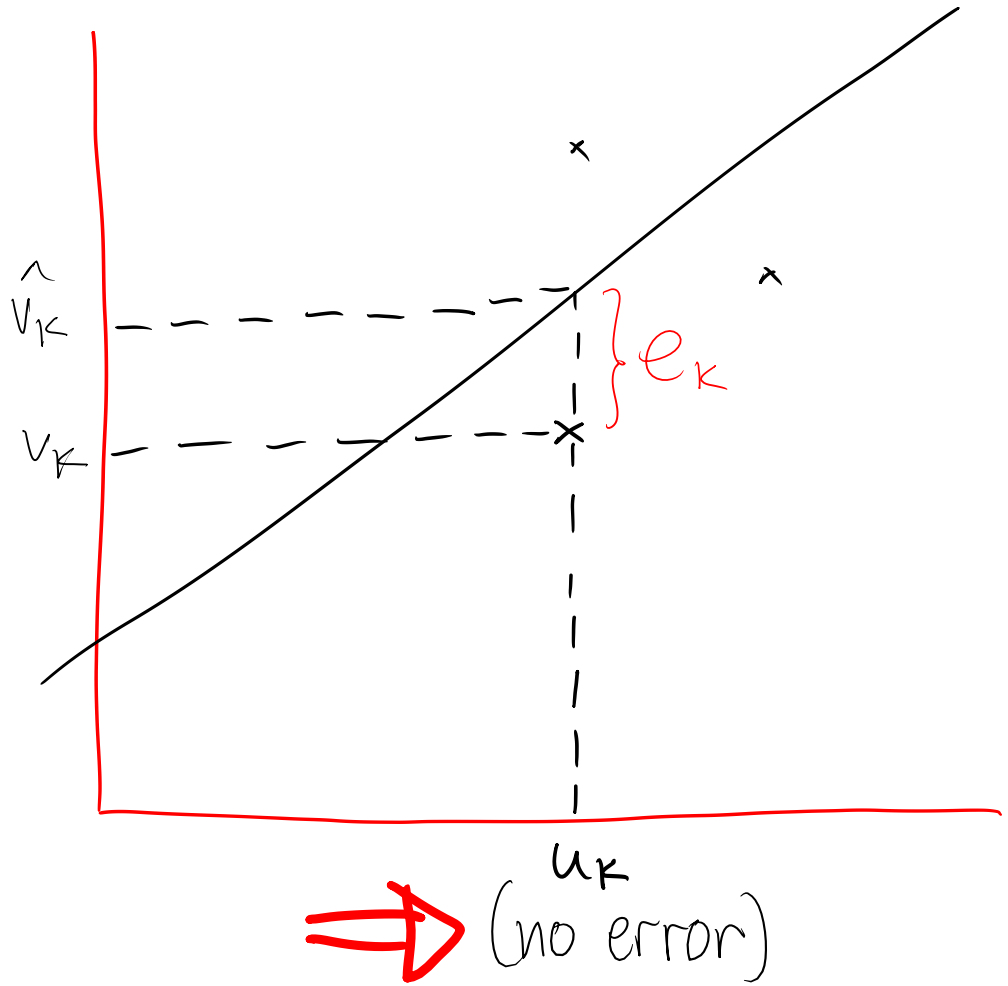
derivative is $\emptyset$ at extremum

$$U^T v = U^T U p \quad \text{or}$$

$$p = (U^T U)^{-1} (U^T v)$$

# modeling errors in all variables

In practice it is unrealistic to assume that some measurements have no error



# minimizing algebraic error

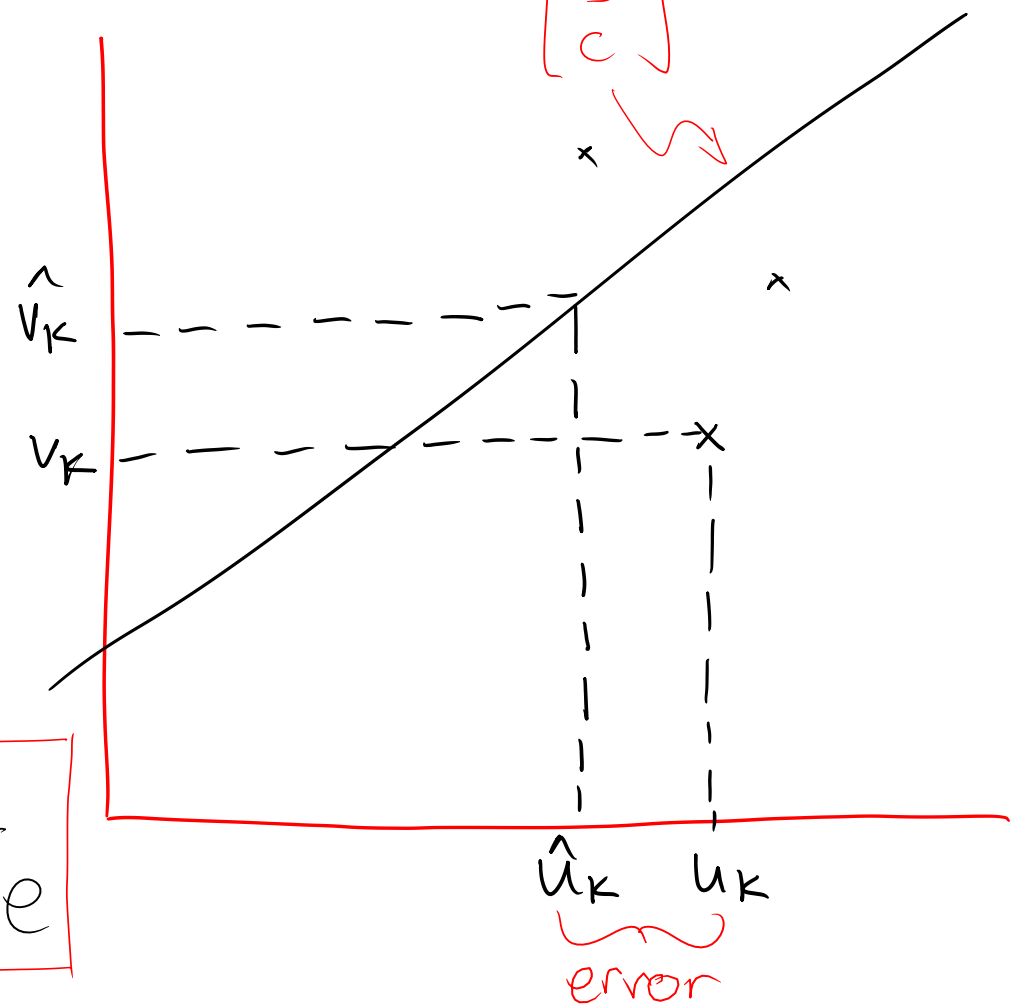
$\Rightarrow$ we need a different optimization objective

$$\begin{bmatrix} \hat{u}_k & \hat{v}_k & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

algebraic error (deviation from implicit line equation)

$$e = \begin{bmatrix} u_1 & v_1 & 1 \\ \vdots & & \\ u_k & v_k & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$D$ $p$ error




Goal: minimize objective $O(p) = e^T e$

# minimizing algebraic error

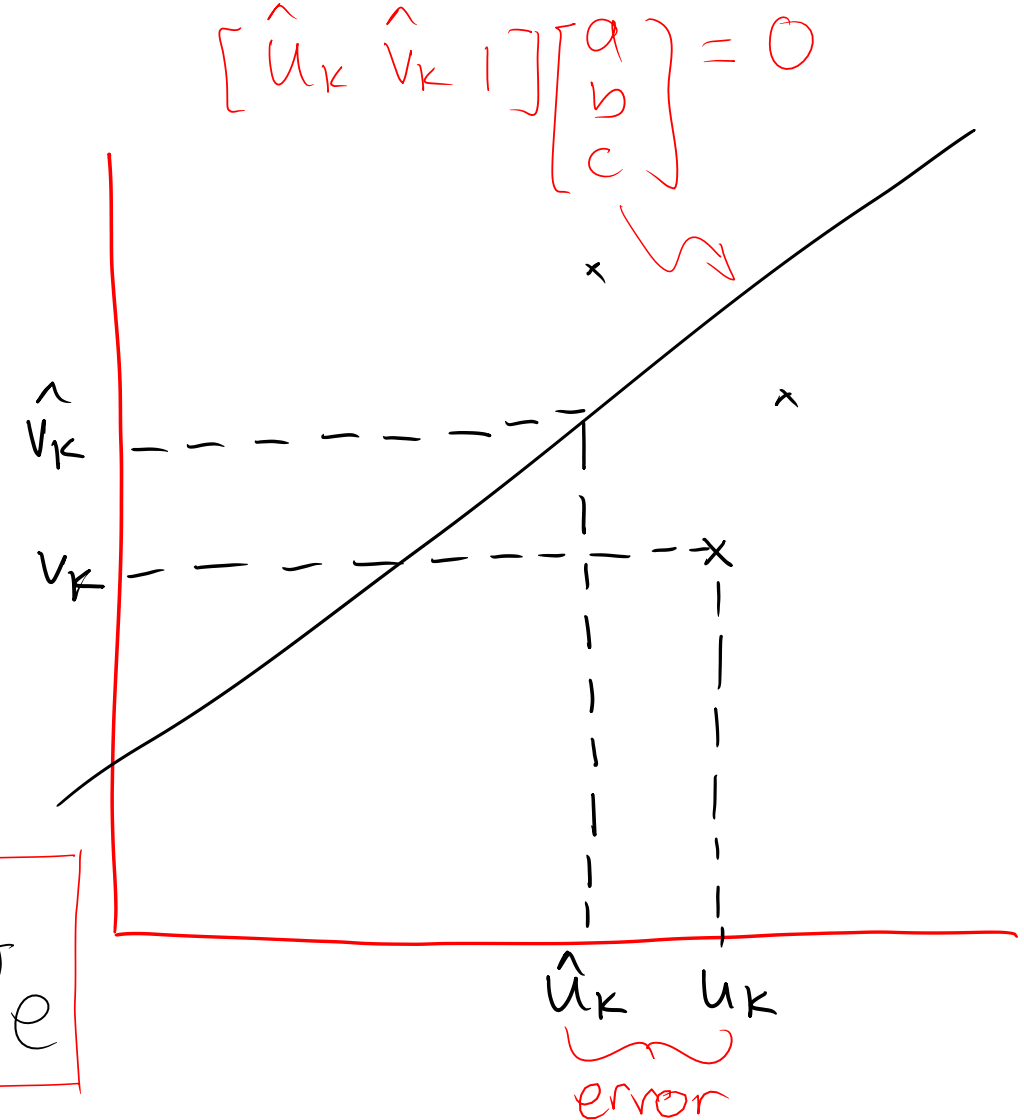
Question: Does this yield a useful optimization?

algebraic error (deviation from implicit line equation)

$$e = \begin{bmatrix} u_1 & v_1 & 1 \\ \vdots & & \\ u_K & v_K & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$D$ $p$

error




Goal: minimize objective $O(p) = e^T e$

# minimizing algebraic error

Question: Does this yield a useful optimization?
Not quite... it accepts the trivial solution

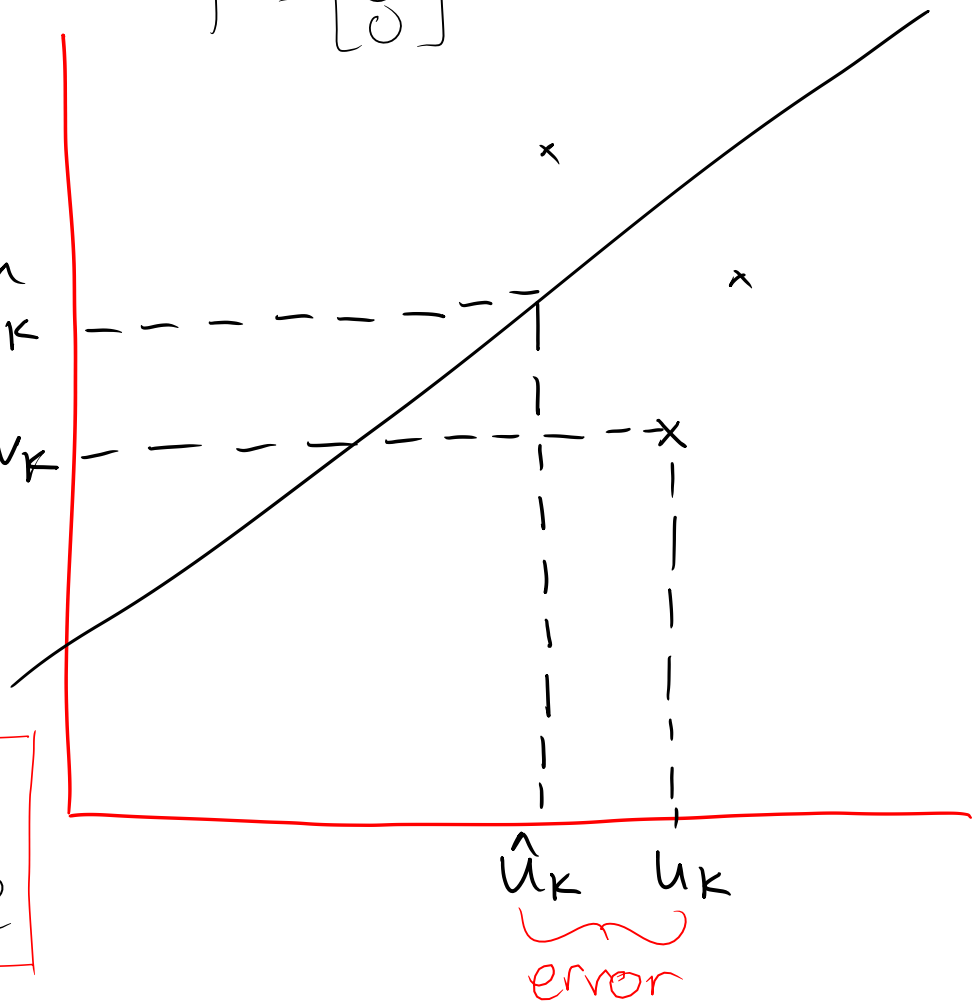
$$p = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

algebraic error
(deviation from implicit line equation)

$$e = \begin{bmatrix} u_1 & v_1 & 1 \\ \vdots & & \\ u_k & v_k & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$D$ $p$

error $\{$ $\hat{v}_k$ , $v_k$




Goal: minimize
objective $O(p) = e^T e$

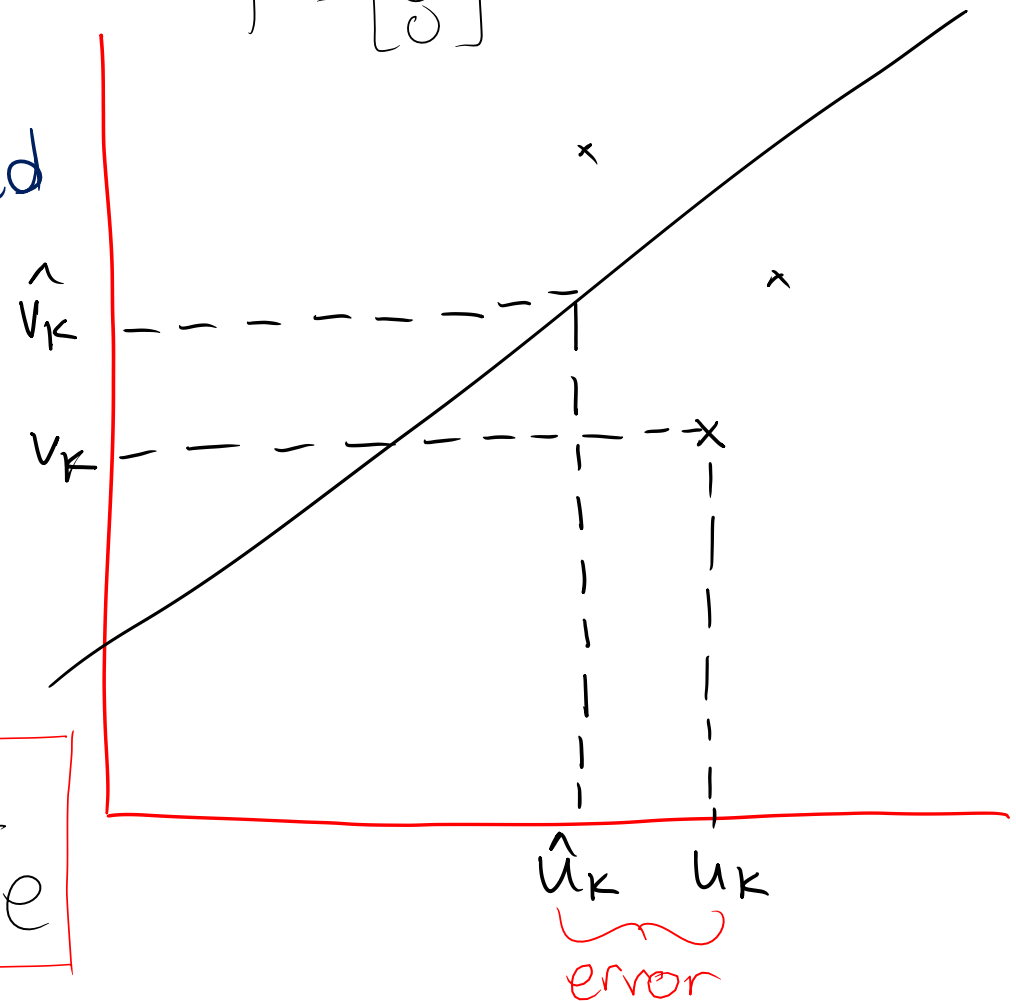
# constrained minimization of algebraic error

Question: Does this yield a useful optimization?
Not quite... it accepts the trivial solution

$$p = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

⇒Must solve a constrained minimization problem

⇒What constraints to use?




Goal: minimize objective $O(p) = e^T e$

# constrained minimization of algebraic error

Question: Does this yield a useful optimization?
Not quite... it accepts the trivial solution

$$p = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

⇒Must solve a constrained minimization problem

⇒What constraints to use?

Some possibilities:

1. $\|p\| = 1$
2. $c = 1$
3. anything else?

Are these constraints equivalent?

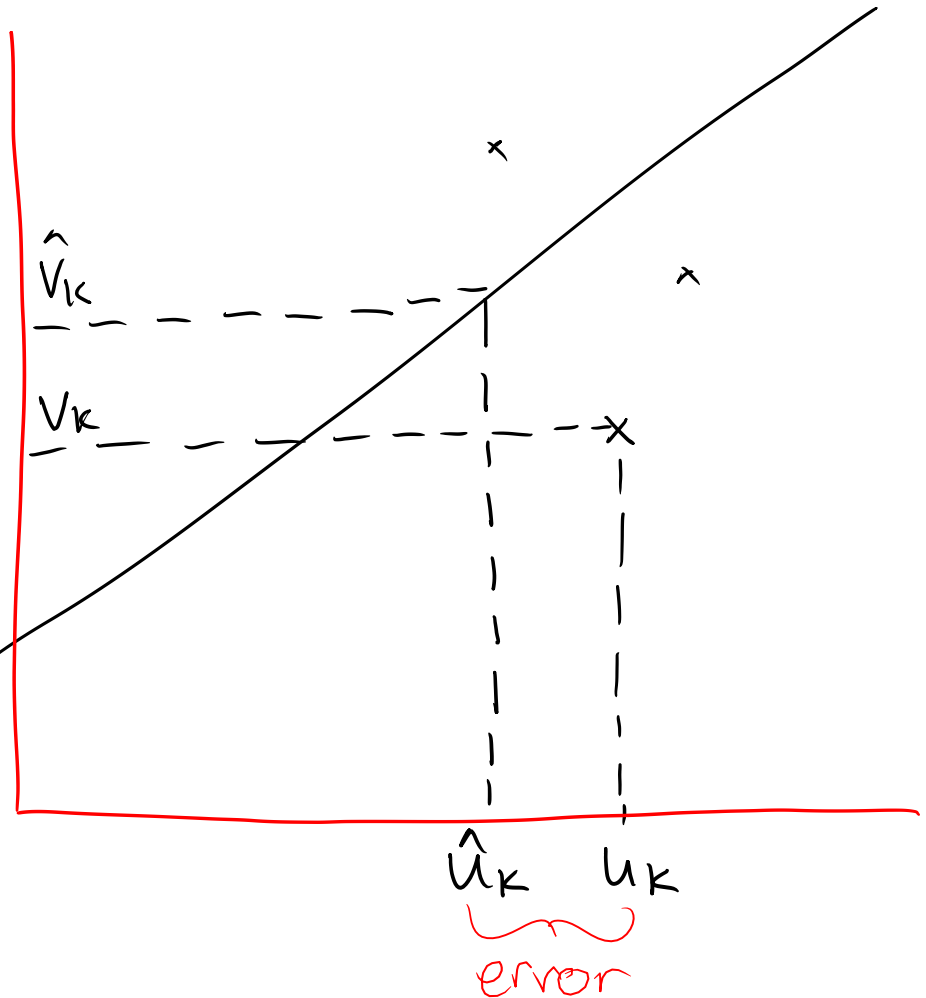
Goal: minimize objective $O(p) = e^T e$

# constrained minimization of algebraic error

Question: Does this yield a useful optimization?
Not quite... it accepts the trivial solution

Key desiderata:

1. invariance to choice of coordinate system




Goal: minimize objective $O(p) = e^T e$

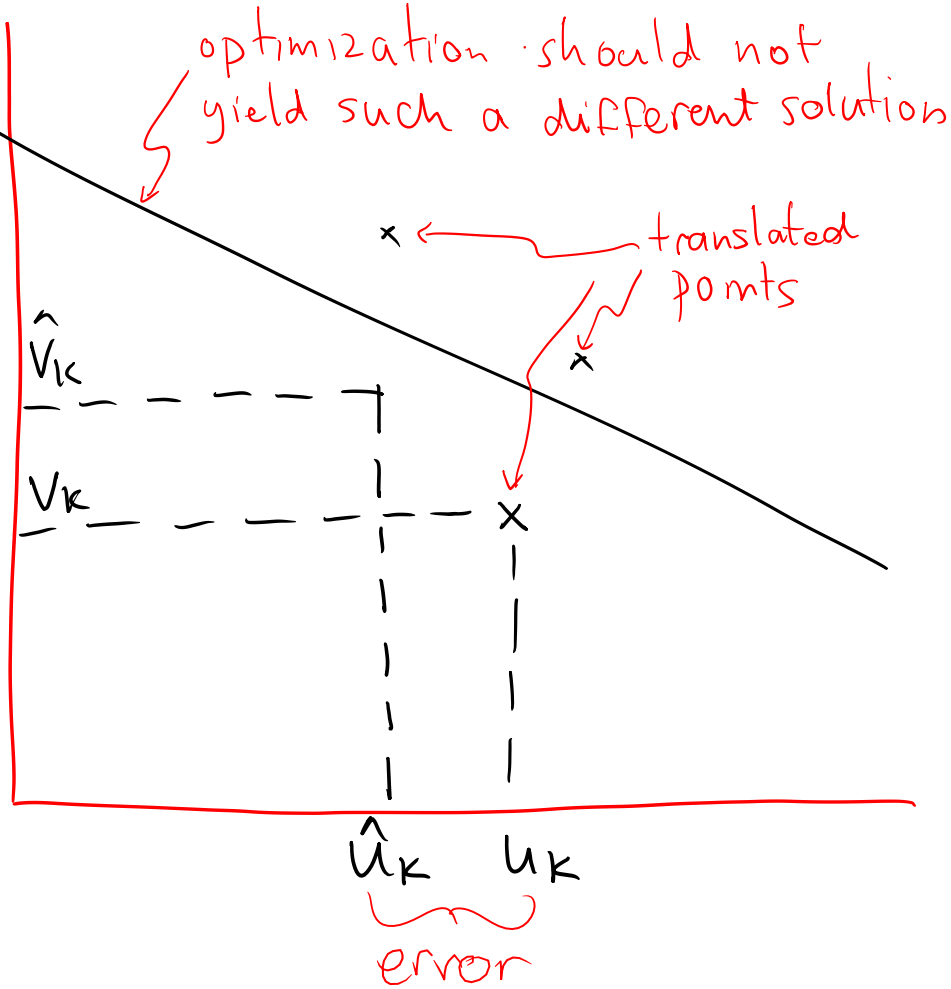
# constrained minimization of algebraic error

Question: Does this yield a useful optimization?
Not quite... it accepts the trivial solution

Key desiderata:

1. invariance to choice of coordinate system

Goal: minimize objective $O(p) = e^T e$



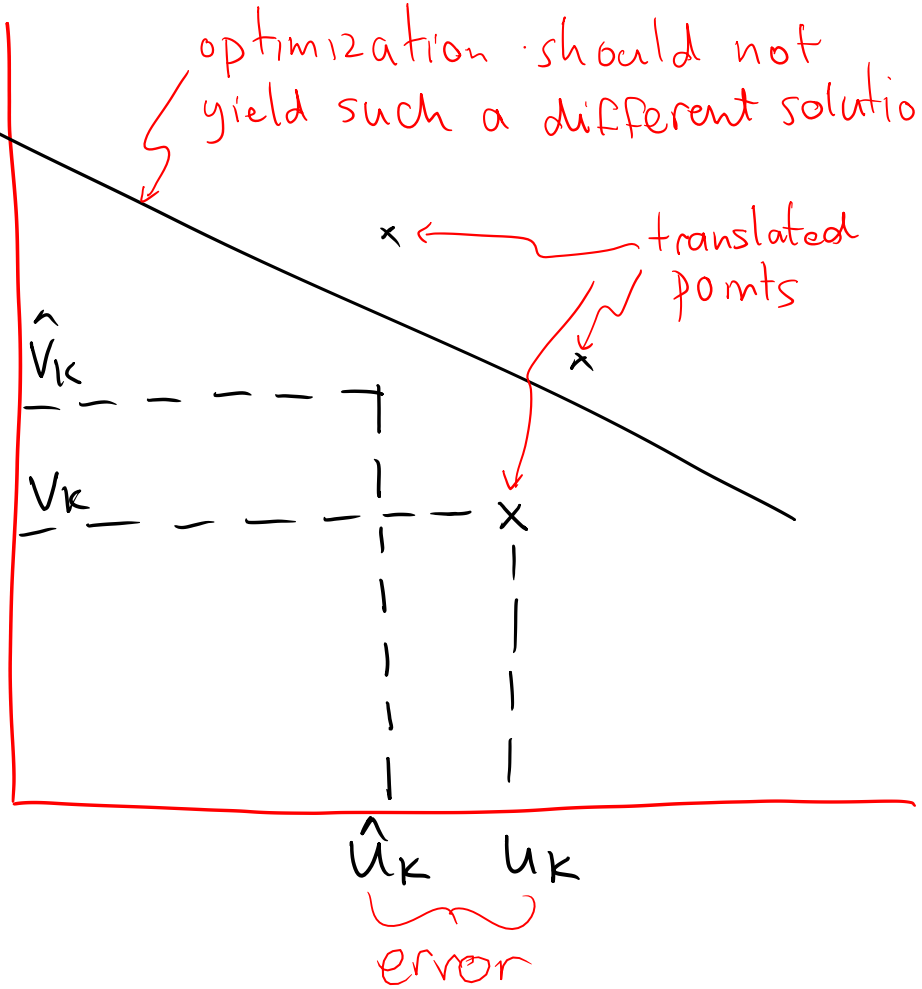
# constrained minimization of algebraic error

Question: Does this yield a useful optimization?
Not quite... it accepts the trivial solution

Key desiderata:

1. invariance to choice of coordinate system
2. invariance to position on line

Goal: minimize objective $O(p) = e^T e$



# modeling errors in all variables: orthogonal error

The equation for points $\hat{x}_k$ on one infinite line $L$ is

$$(\hat{X}_k)^T \cdot p = 0 \quad \text{with} \quad \hat{x}_k = \begin{bmatrix} \hat{u}_k \\ \hat{v}_k \\ 1 \end{bmatrix}$$

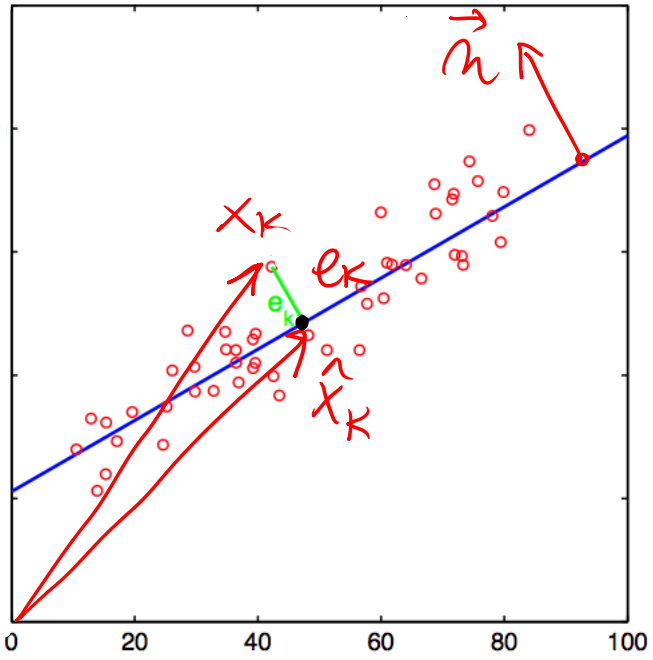
Here we normalize $||\vec{n}|| = 1$. The error in an observed point $\vec{x}_k$ (relative to the line $L$) is defined to be the perpendicular distance

$$p = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} \vec{n} \\ c \end{bmatrix}$$

(1) for noiseless $\hat{x}_k$:

$$(\hat{X}_k)^T p = 0$$

$$[\hat{u}_k \; \hat{v}_k] n + c = 0$$




(2) orthogonal distance $e_k$:

$$[u_k - \hat{u}_k \; v_k - \hat{v}_k] n =$$

$$[u_k \; v_k] n + c =$$

$$(X_k)^T p$$

# problem statement ("total least squares" -- TLS)

Given a set of edgel positions $D = \{\vec{x}_k\}_{k=1}^{K}$, consider estimating the line parameters $(\vec{n}, c)$ by minimizing the squared error

$$\mathcal{O}(\vec{n}, c) \equiv \sum_{k=1}^{K} (\vec{n}^T \vec{x}_k + c)^2 , \quad \text{for } ||\vec{n}|| = 1. \qquad (1)$$

measurements in Euclidean coordinates

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_K \end{bmatrix} = \begin{bmatrix} (x_1)^T \\ (x_2)^T \\ \vdots \\ (x_K)^T \end{bmatrix} \begin{bmatrix} \vec{n} \\ c \end{bmatrix} = \begin{bmatrix} X_{Kx2} & 1_{Kx1} \end{bmatrix} \begin{bmatrix} \vec{n} \\ c \end{bmatrix}$$

column vector of K ones

$$O(n, c) = e^T e$$

$$= [Xn + 1c]^T [Xn + 1c]$$

$$= n^T X^T X n + 2(n^T X^T 1)c + (1^T 1)c^2$$

$= Kc^2$

# total least squares: derivation of solution

$$\emptyset = \frac{\partial O}{\partial c} = 2\,c + 2n^T X^T 1 \Rightarrow c = -n^T \frac{X^T 1}{K}$$

mean of the $x_k$'s

$$\emptyset = \frac{\partial O}{\partial n} = 2n^T X^T X + 2c\,1^T X$$

equivalent to $n^T\left(X^T X - \frac{1}{K} X^T 1 1^T X\right) = n^T \emptyset$

$$\Leftrightarrow\; n^T X^T X = -c\,1^T X$$

$$\Leftrightarrow\; n^T X^T X = n^T X^T 1 1^T X \cdot \frac{1}{K}$$

$\Rightarrow$ optimal $n$ is smallest eigenvalue

$$O(n,c) = e^T e$$

$$= [Xn + 1c]^T [Xn + 1c]$$

$$= n^T X^T X n + 2(n^T X^T 1)c + (1^T 1)c^2$$

# total least squares: solution

- $\vec{n}$ must be an eigenvector of $X^T X - \frac{1}{k} X^T 1 1^T X$
- $c$ must be chosen such that the estimated line passes through the mean of the points;
- and for the solution to be a local minimum, it also can be shown that $\vec{n}$ must be the eigenvector for the *minimum* eigenvalue of

The solution is therefore unique and easy to compute. No initial guess is required.

# Topic 01:

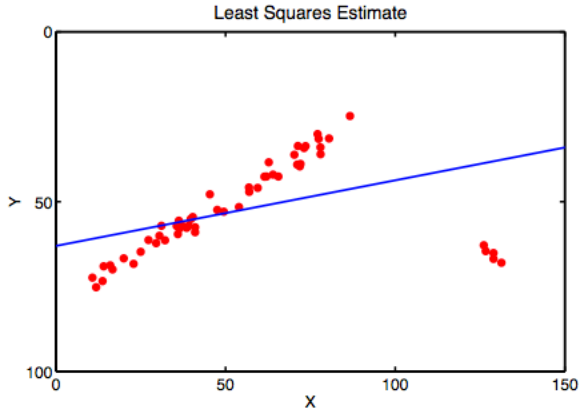
# Robust Estimation Basics

- least squares & total least squares
- robust M-estimators
- optimizing robust objective functions
- proposal generation: Hough transforms
- proposal generation: RANSAC

# effect of outliers

Unfortunately, least squares solutions are sensitive to outliers in the data.

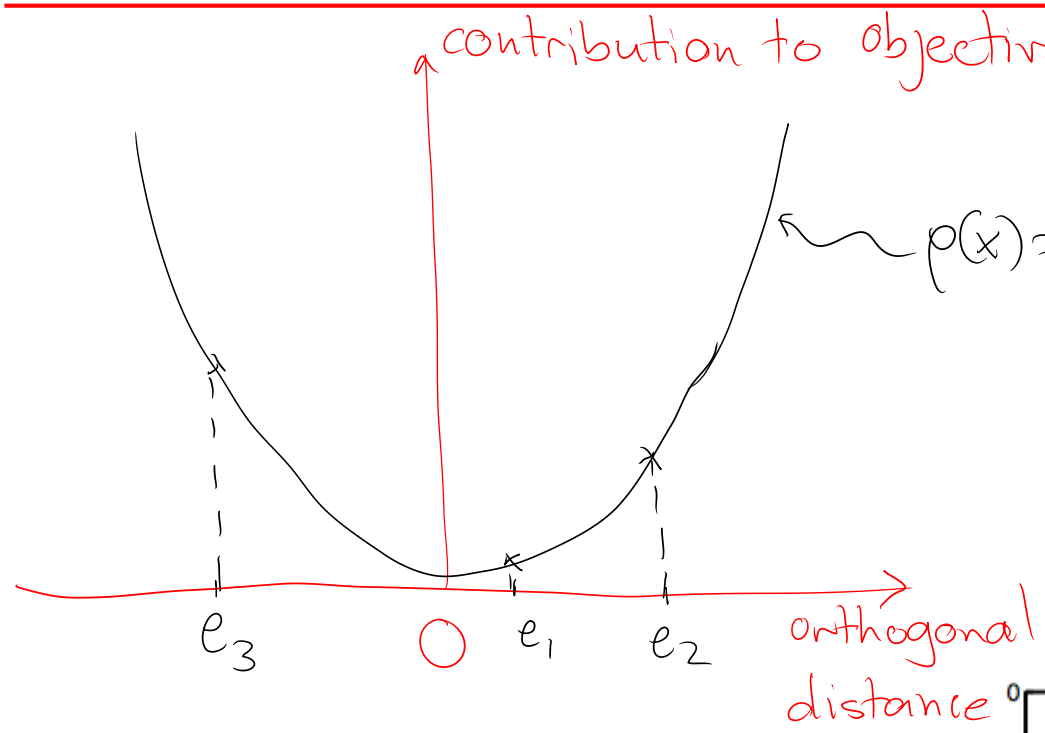
For example, consider the set of edgel point data $\{\vec{x}_k\}_{k=1}^{K}$ depicted by red dots:




The least squares estimate (blue line, above) is strongly influenced by the small cluster of outliers.

In other words, this LS approach is not directly suitable for fitting data with outliers or when multiple solutions are expected.

# effect of outliers

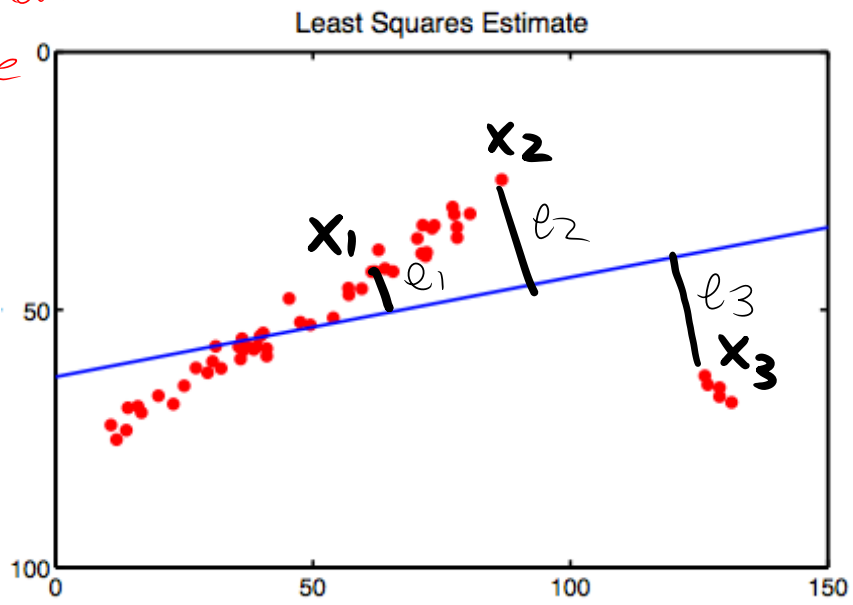



observation:

error is a quadratic function of distance $e_k$

$\Rightarrow$

large distances contribute much more to objective



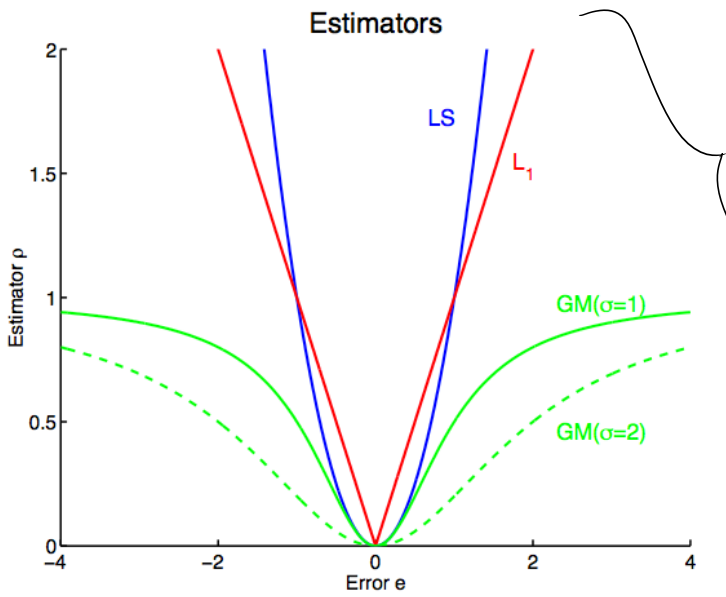
# robust M-estimation

We seek line parameters $(\vec{n}, c)$ which (locally) solve the minimization problem

$$\min \mathcal{O}(\vec{n}, c)\,, \quad \text{for } ||\vec{n}|| = 1\,, \tag{14}$$

$$\mathcal{O}(\vec{n}, c) \equiv \sum_{k=1}^{K} \rho(e(\vec{x}_k\,;\,\vec{n}, c))\,. \tag{15}$$

Here $\rho(e)$ is the "estimator", which provides a cost for any given error $e$. Common choices include:



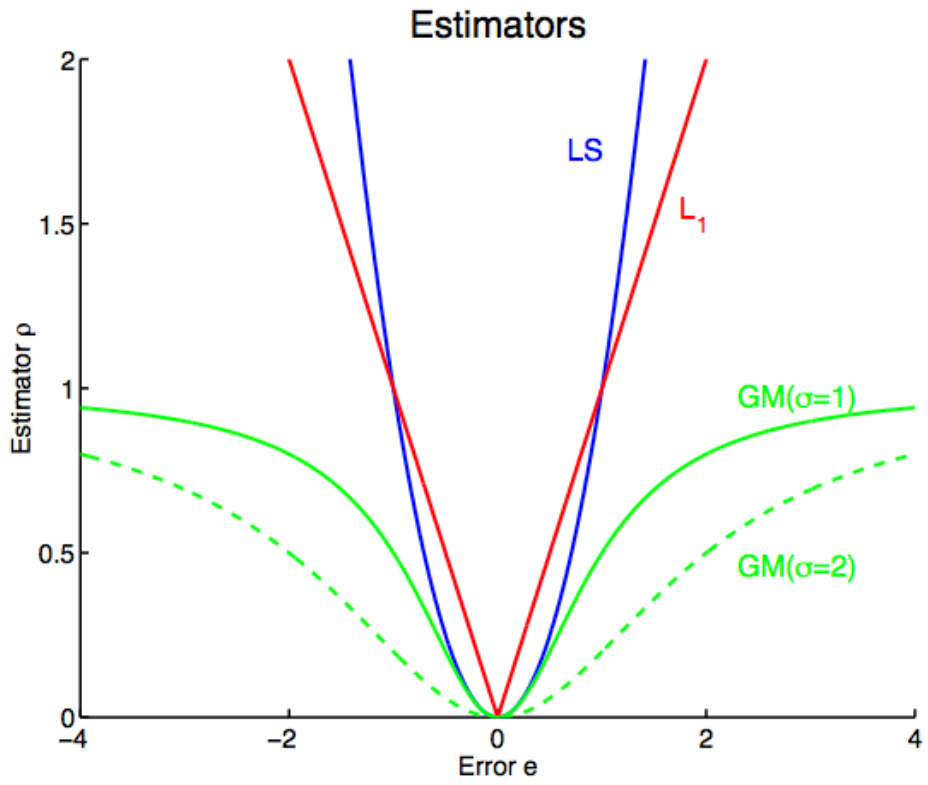

these functions grow A LOT slower than LS

# robust M-estimation

$$\rho(e) = e^2, \qquad \text{Least squares (LS) estimator,}$$

$$\rho(e) = |e|, \qquad L_1 \text{ estimator,}$$

$$\rho(e) = \frac{e^2}{\sigma^2 + e^2}, \qquad \text{Geman-McLure (GM) estimator.}$$



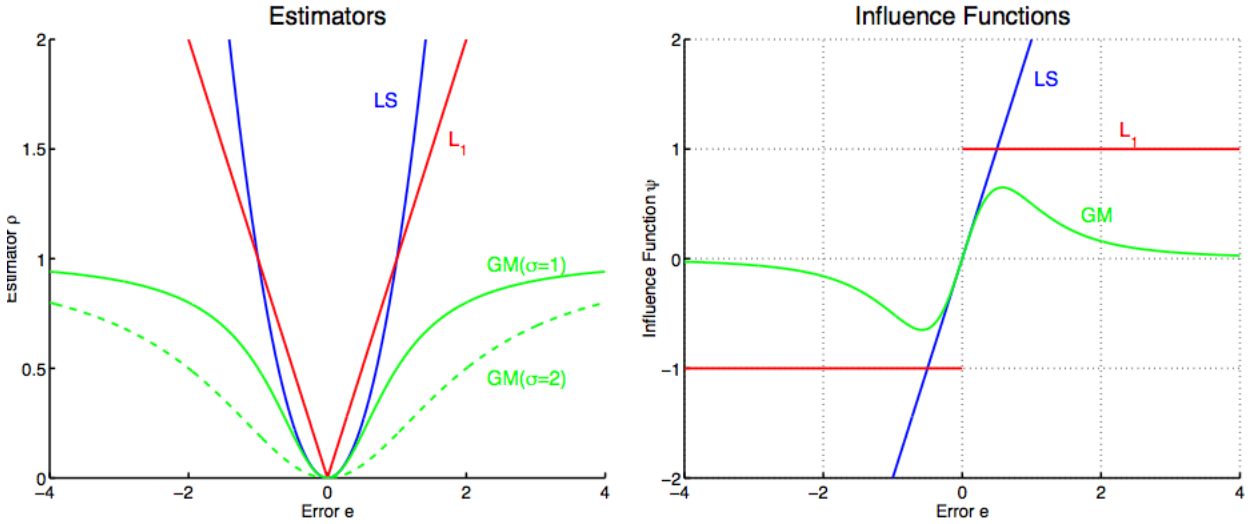
# the influence function

The influence function describes the sensitivity of the overall estimate, $(\vec{n}, c)$, on data with error $e$.

**Definition:** Given the estimator, $\rho(e)$, the influence function, $\psi(e)$, is defined to be:

rate of growth with e

$$\psi(e) = \frac{d\rho}{de}(e) . \qquad (16)$$



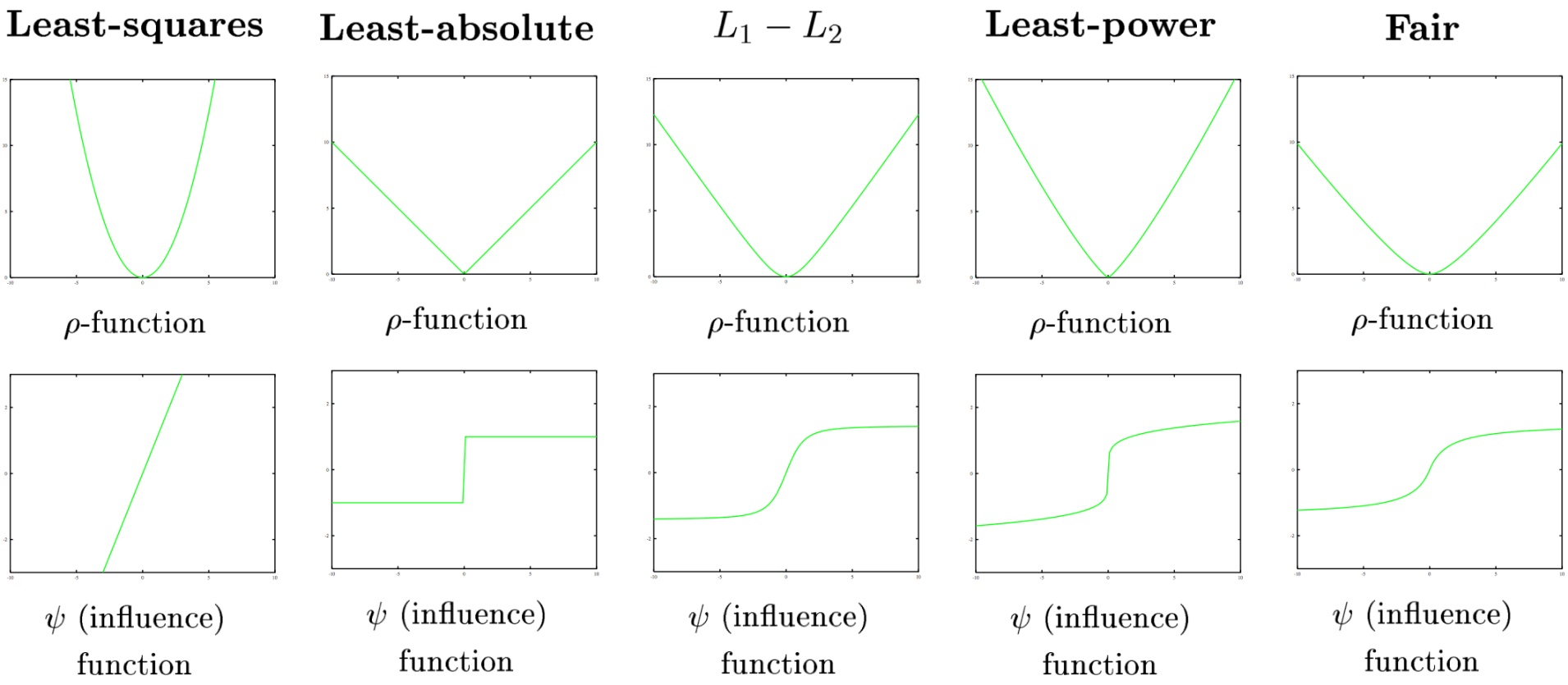
# examples of M-estimators & influence functions

Table 1: A few commonly used M-estimators

| type | $\rho(x)$ | $\psi(x)$ |
|---|---|---|
| $L_2$ | $x^2/2$ | $x$ |
| $L_1$ | $\|x\|$ | $\mathrm{sgn}(x)$ |
| $L_1 - L_2$ | $2(\sqrt{1 + x^2/2} - 1)$ | $\frac{x}{\sqrt{1 + x^2/2}}$ |
| $L_p$ | $\frac{\|x\|^\nu}{\nu}$ | $\mathrm{sgn}(x)\|x\|^{\nu-1}$ |
| "Fair" | $c^2[\frac{\|x\|}{c} - \log(1 + \frac{\|x\|}{c})]$ | $\frac{x}{1 + \|x\|/c}$ |

(Z. Zhang, IVC1997)

# examples of M-estimators & influence functions




(Z. Zhang, IVC 1997)

# examples of M-estimators & influence functions

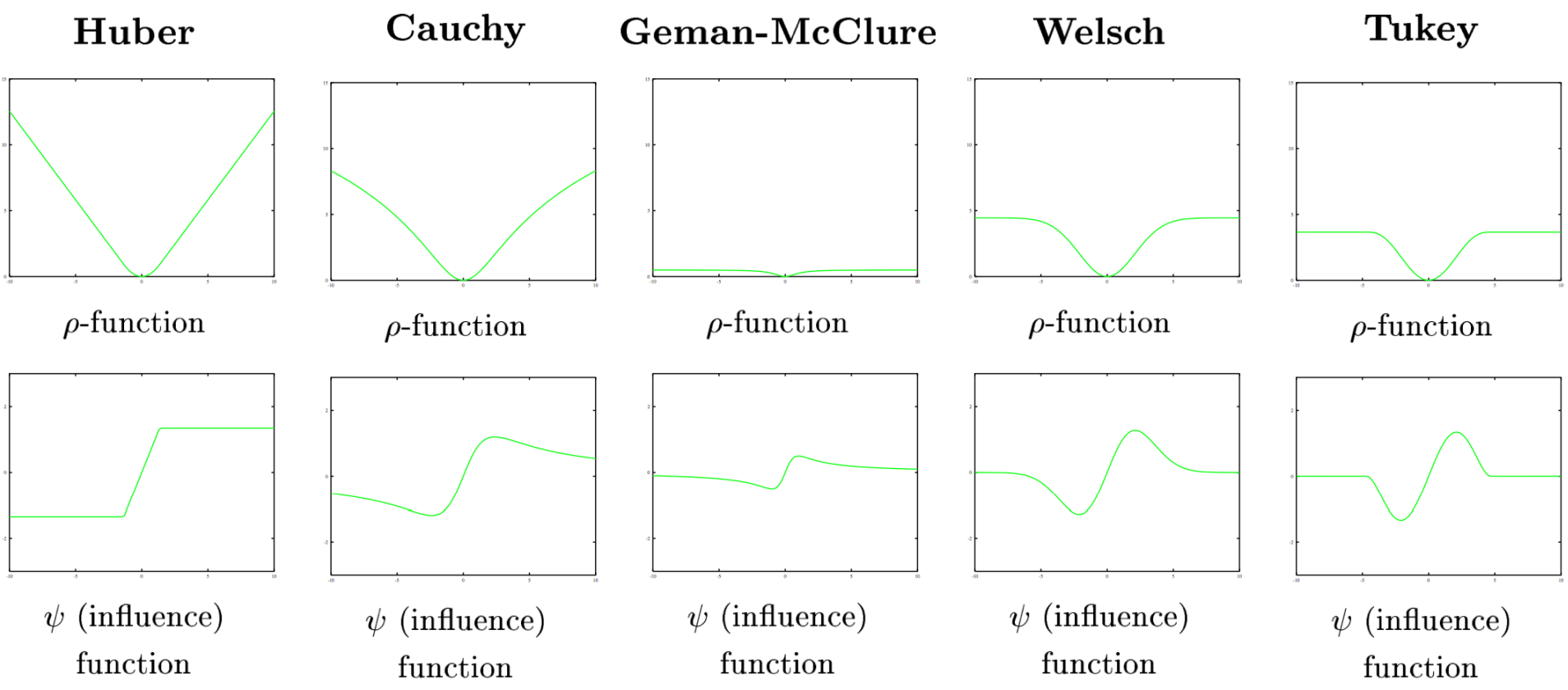
Table 1: A few commonly used M-estimators

| type | $\rho(x)$ | $\psi(x)$ |
|---|---|---|
| Huber $\begin{cases}\text{if } \lvert x\rvert \le k\\ \text{if } \lvert x\rvert \ge k\end{cases}$ | $\begin{cases}x^2/2\\ k(\lvert x\rvert - k/2)\end{cases}$ | $\begin{cases}x\\ k\,\mathrm{sgn}(x)\end{cases}$ |
| Cauchy | $\frac{c^2}{2}\log(1+(x/c)^2)$ | $\frac{x}{1+(x/c)^2}$ |
| Geman-McClure | $\frac{x^2/2}{1+x^2}$ | $\frac{x}{(1+x^2)^2}$ |
| Welsch | $\frac{c^2}{2}[1-\exp(-(x/c)^2)]$ | $x\exp(-(x/c)^2)$ |
| Tukey $\begin{cases}\text{if } \lvert x\rvert \le c\\ \text{if } \lvert x\rvert > c\end{cases}$ | $\begin{cases}\frac{c^2}{6}\left(1-[1-(x/c)^2]^3\right)\\ [0.2cm](c^2/6)\end{cases}$ | $\begin{cases}x[1-(x/c)^2]^2\\ 0\end{cases}$ |

can also be a constant 6

(Z. Zhang, IVC1997)

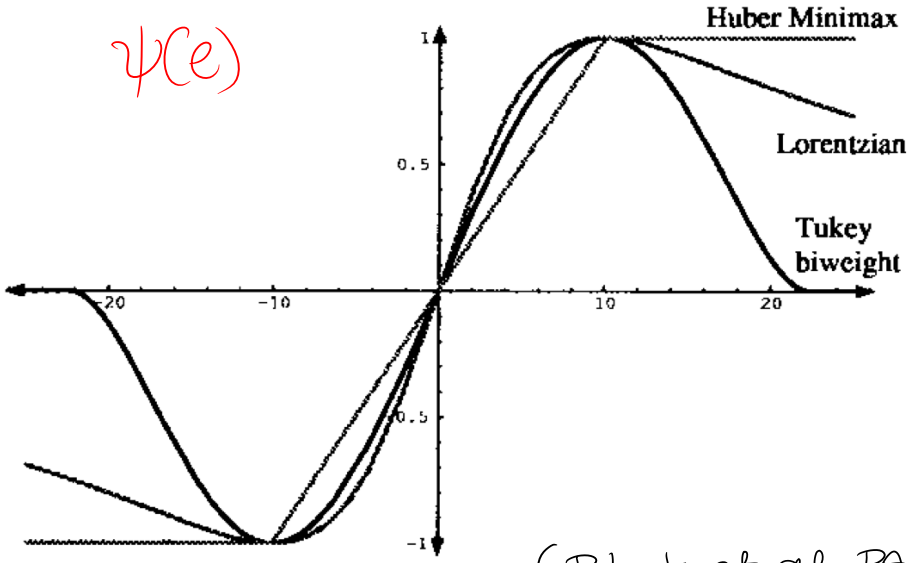
# examples of M-estimators & influence functions




(Z. Zhang, IVC1997)

# the influence function: (weak) desiderata

- bounded influence function




influence functions after appropriate scaling

(Black et al, PAMI98)

- convexity of $\rho(\,)$
- $\rho(\,)$ differentiable at its extrema

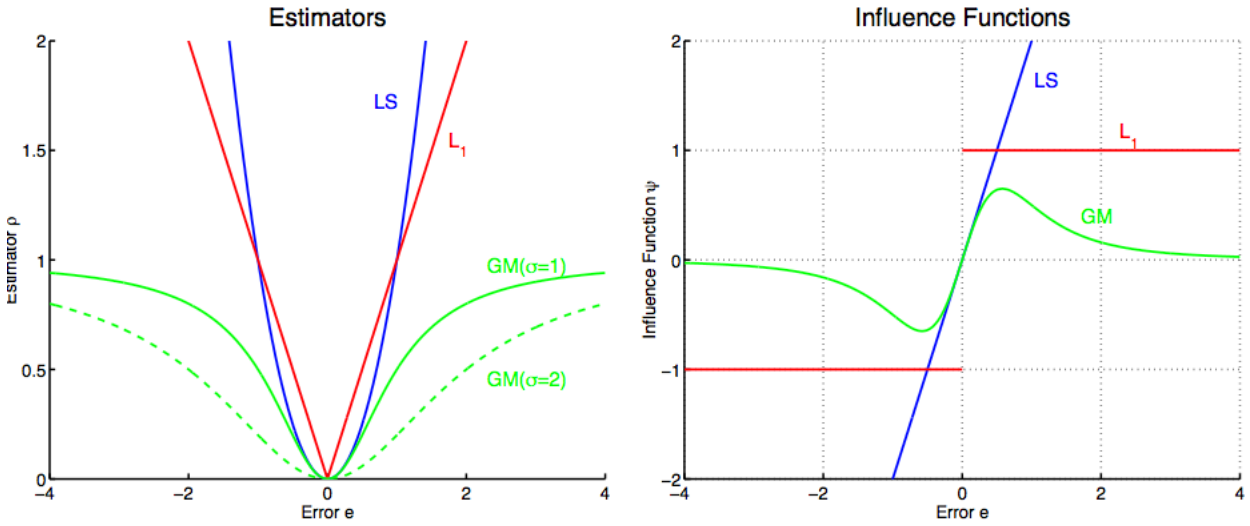
# the influence function

The influence function describes the sensitivity of the overall estimate, $(\vec{n}, c)$, on data with error $e$.

**Definition:** Given the estimator, $\rho(e)$, the influence function, $\psi(e)$, is defined to be:

rate of growth with e

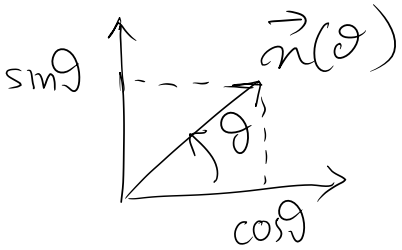
$$\psi(e) = \frac{d\rho}{de}(e) . \tag{16}$$

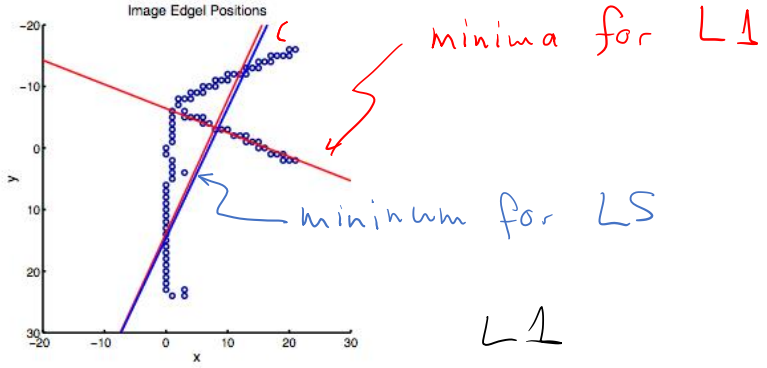



*Remark:* Unlike LS, the influence functions for $L_1$ and GM are bounded. Moreover, GM is a 'redescending estimator', that is, $\psi(e) \to 0$ as $|e| \to \infty$.

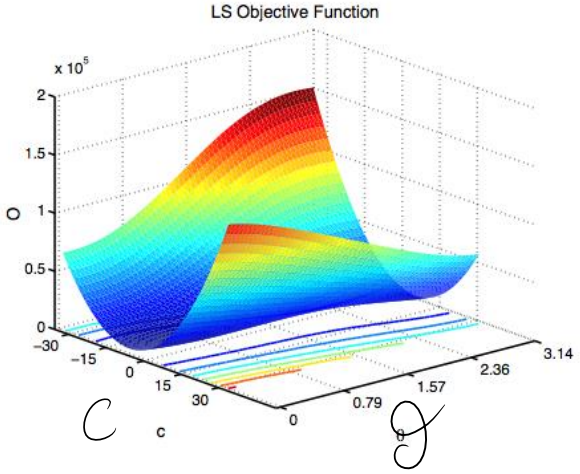
# examples: LS and L1

Given a data set $\{\vec{x}_k\}_{k=1}^K$, we can plot the objective function $\mathcal{O}(\vec{n}(\theta), c)$ for $\vec{n}(\theta) = (\cos(\theta), \sin(\theta))^T$. Note that, since $\vec{n}(\theta + \pi) = -\vec{n}(\theta)$, the objective function is periodic in $\theta$ with $\mathcal{O}(\vec{n}(\theta + \pi), c) = \mathcal{O}(\vec{n}(\theta), -c)$.

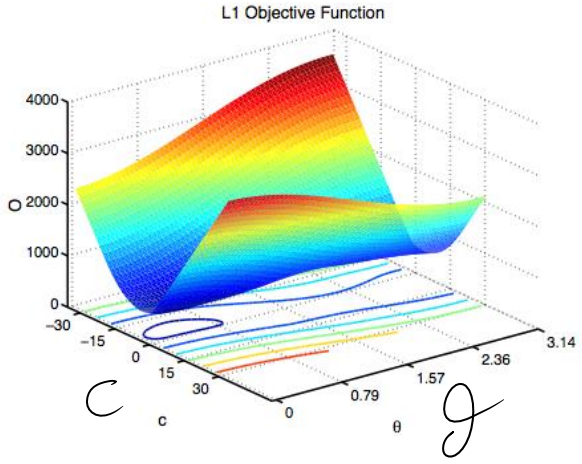



LS

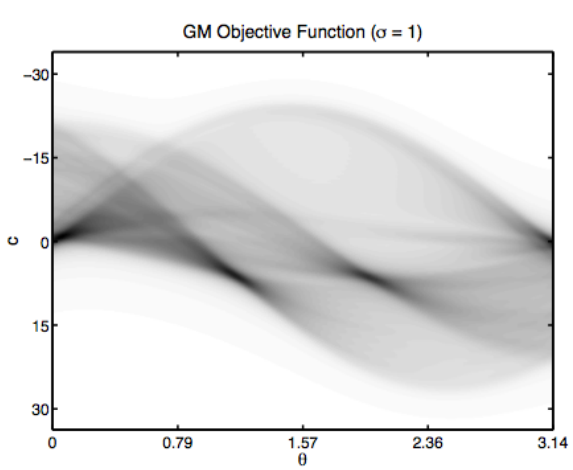


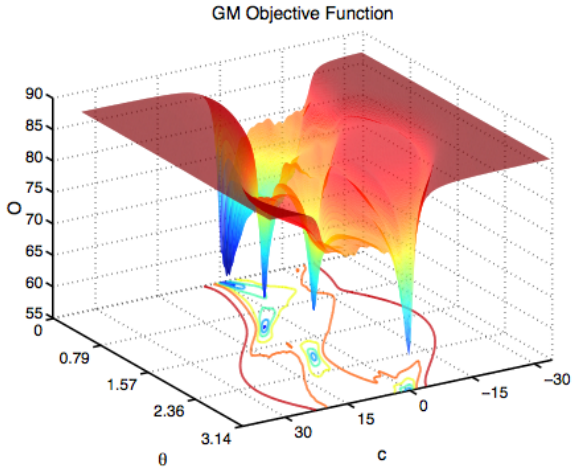

L1



# examples: GM

In contrast, for the GM estimator the influence of a data point with large error (essentially) vanishes. For the same data as before:

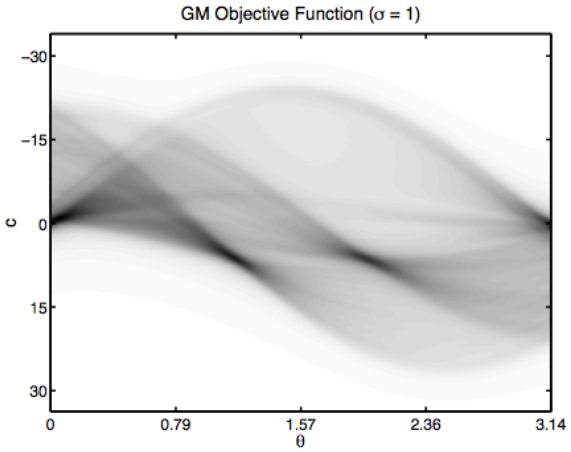


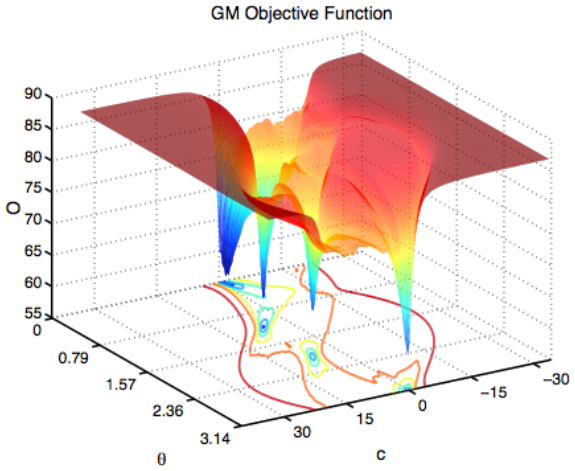



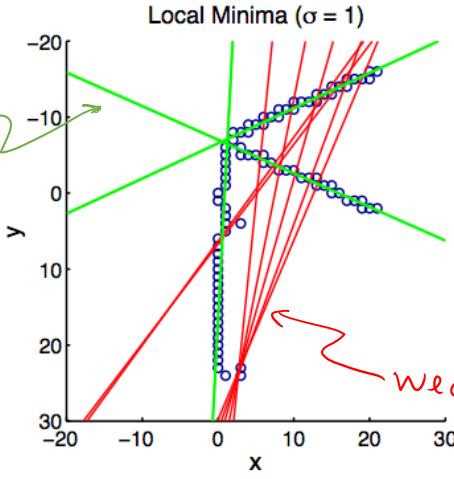

For the GM estimator, the objective function exhibits 3 strong minima (the green lines below). The outer pair of the deep valleys above are actually parts of the same valley, due to periodicity in $\theta$.

# examples: GM

In contrast, for the GM estimator the influence of a data point with large error (essentially) vanishes. For the same data as before:




strongest local minima

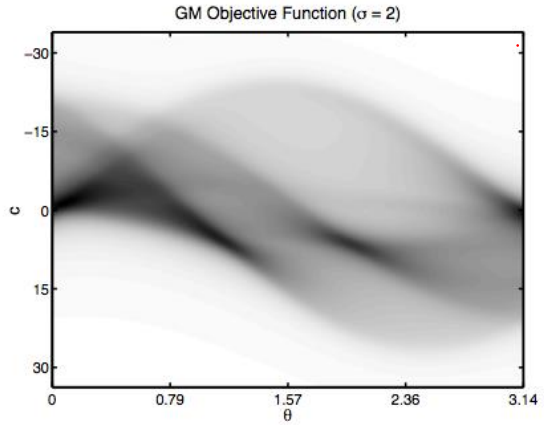
weaker local minima

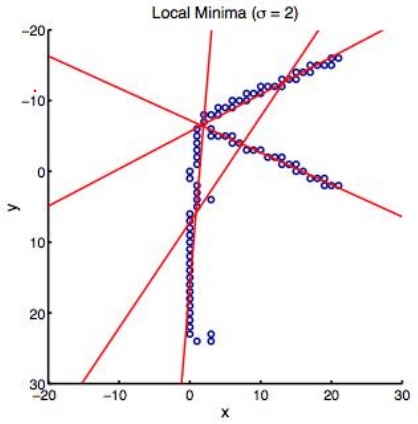
We show only those local minima for which $\mathcal{O} < K - 10$, where $K$ is the number of data points.

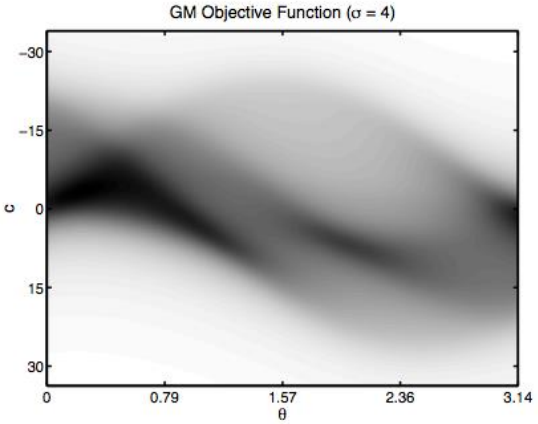
Note that $\rho(e) \leq 1$, so $K$ is the maximum possible value for $\mathcal{O}$.

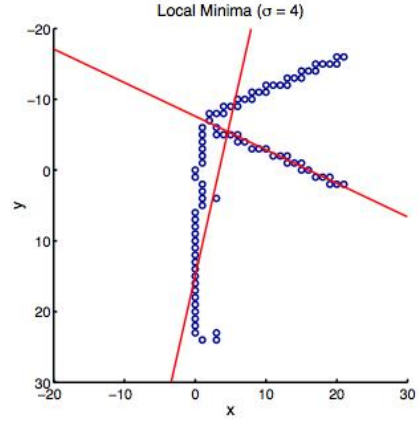
# examples: GM for varying sigma

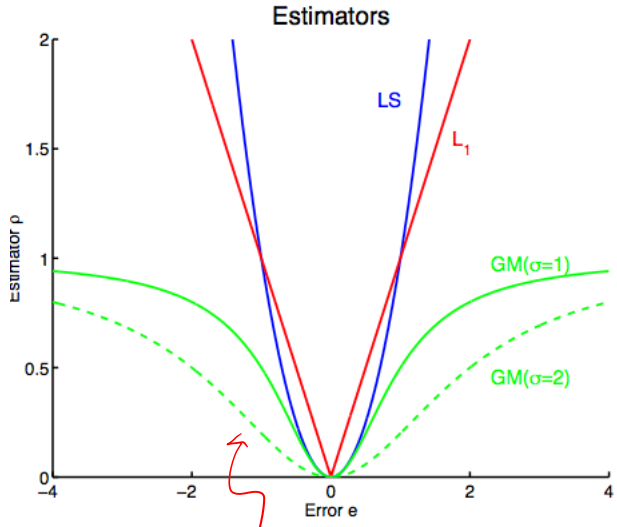
Increasing $\sigma$ in the GM-estimator smooths the objective function:



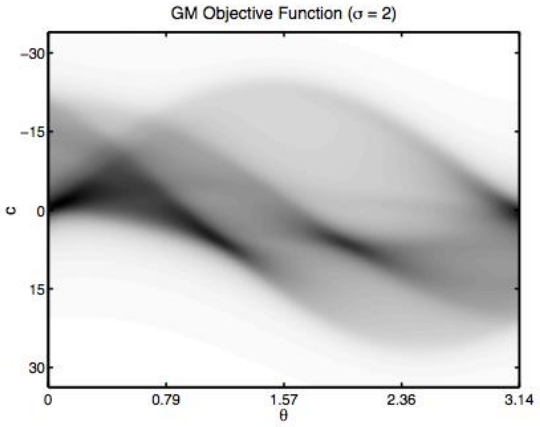



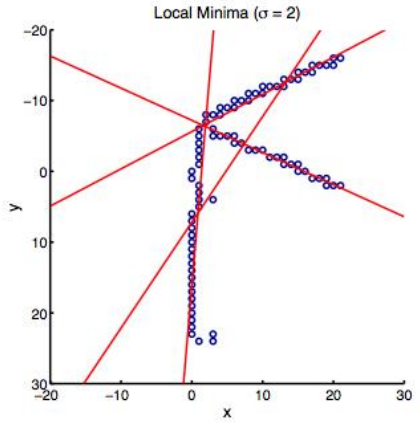

as 6 increases, the GM estimator approaches a convex function

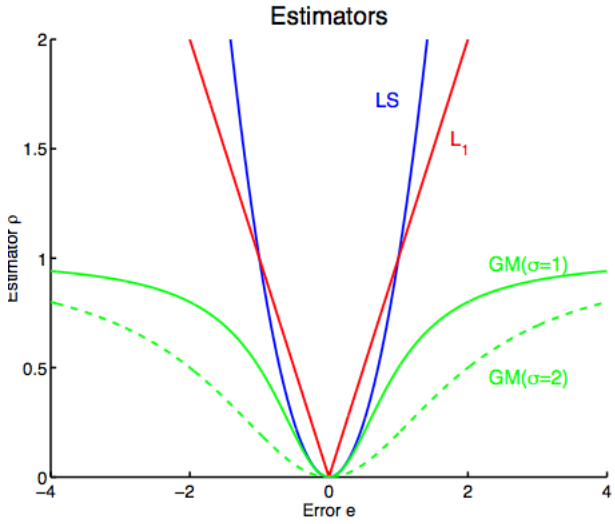
↑ these look like "blurred" versions of the 6=1 objective (more on this later)

# examples: GM for varying sigma

Increasing $\sigma$ in the GM-estimator smooths the objective function:




- Note there are fewer local minima for larger $\sigma$. (We show *all* the local minimum beyond minor fluctuations.)
- As $\sigma$ is increased, outliers receive greater influence, and therefore the bias in the estimates often increases (e.g., bottom right plot).
- As $\sigma \to \infty$ the objective function can be shown to approximate the one for least squares (with a unique solution).

Lecture stopped here

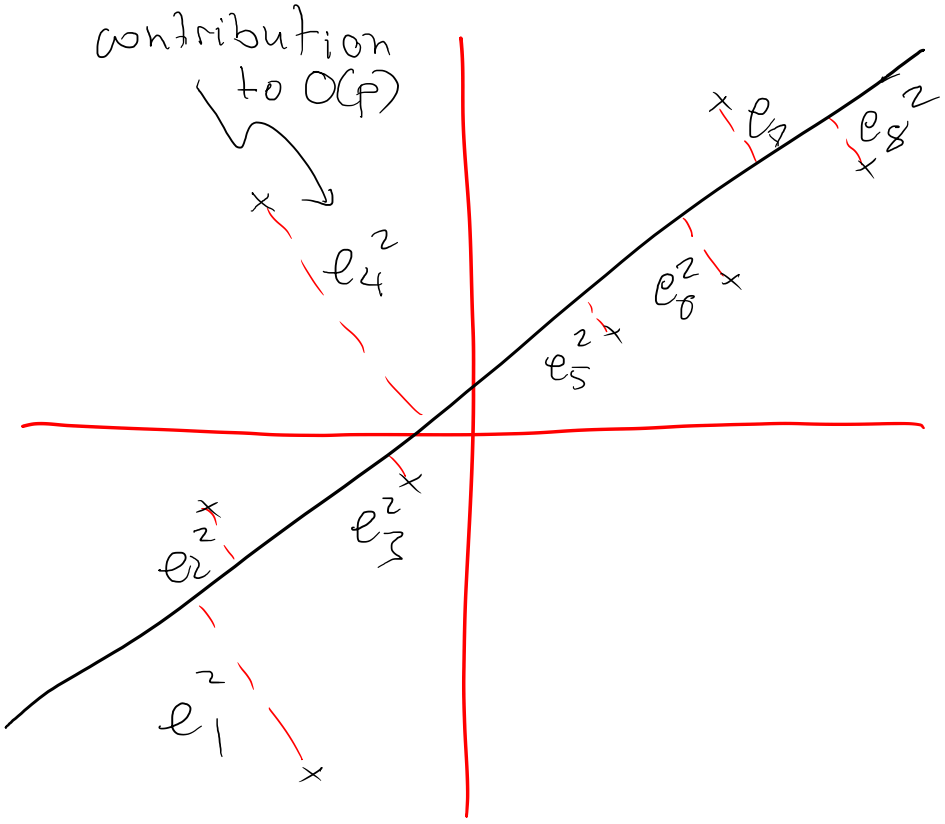
To be continued next week

# Topic 01:

# Robust Estimation Basics

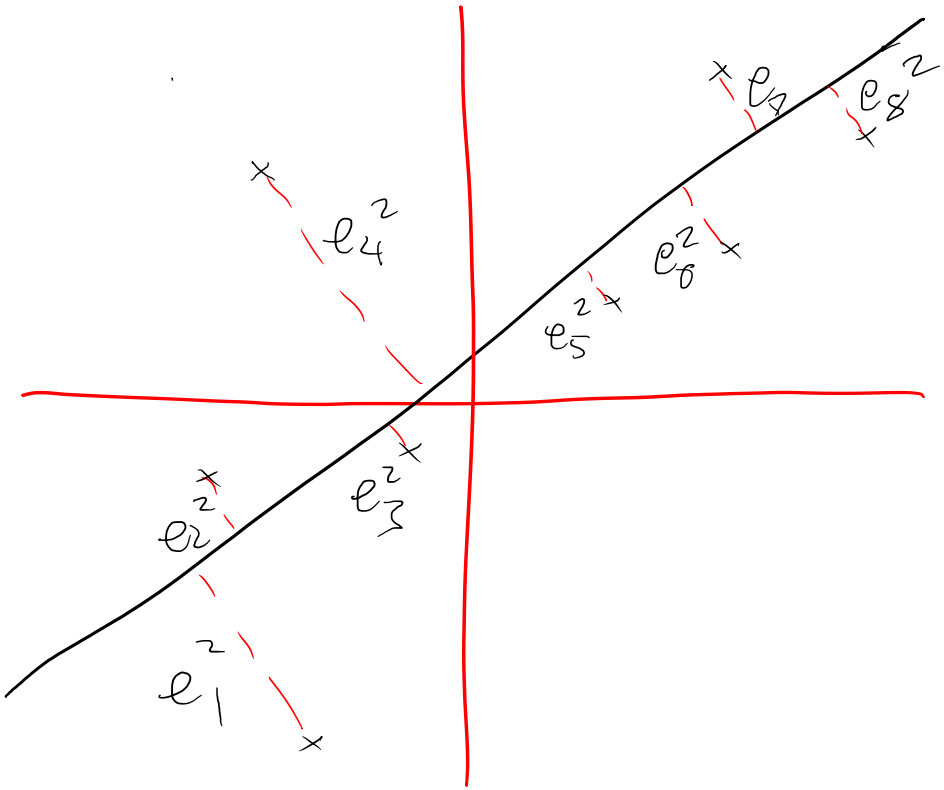
- least squares & total least squares
- robust M-estimators
- optimizing robust objective functions
- proposal generation: Hough transforms
- proposal generation: RANSAC

# our starting point: total least squares estimation

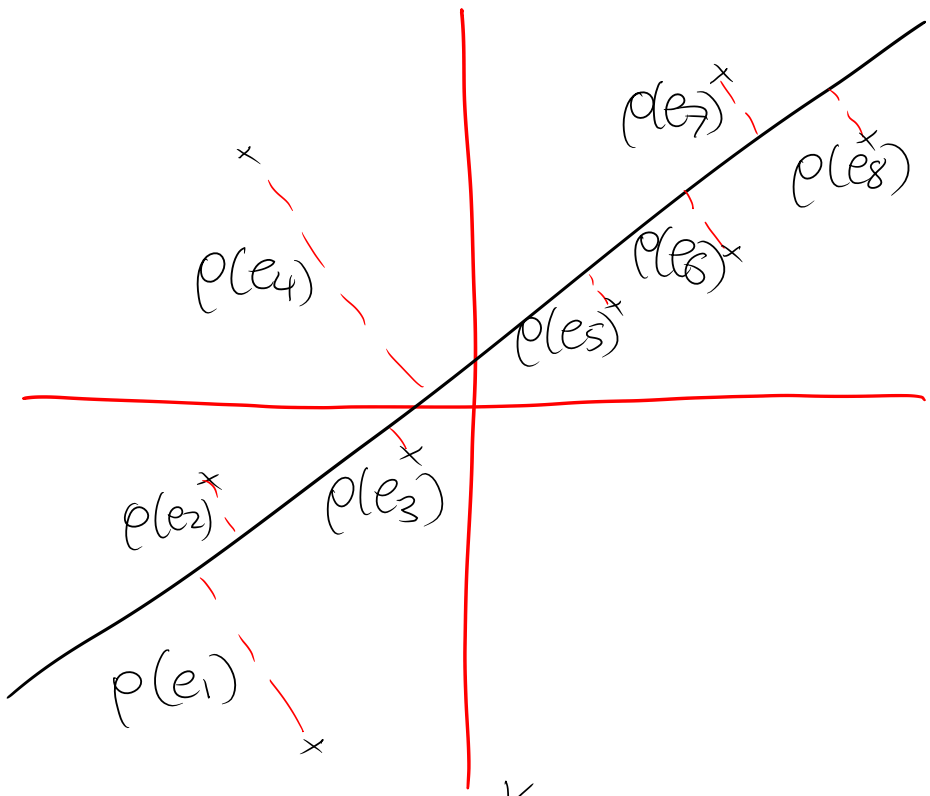



$$O(p) = \sum_{k=1}^{K} e_k^2$$

# our target: M-estimation



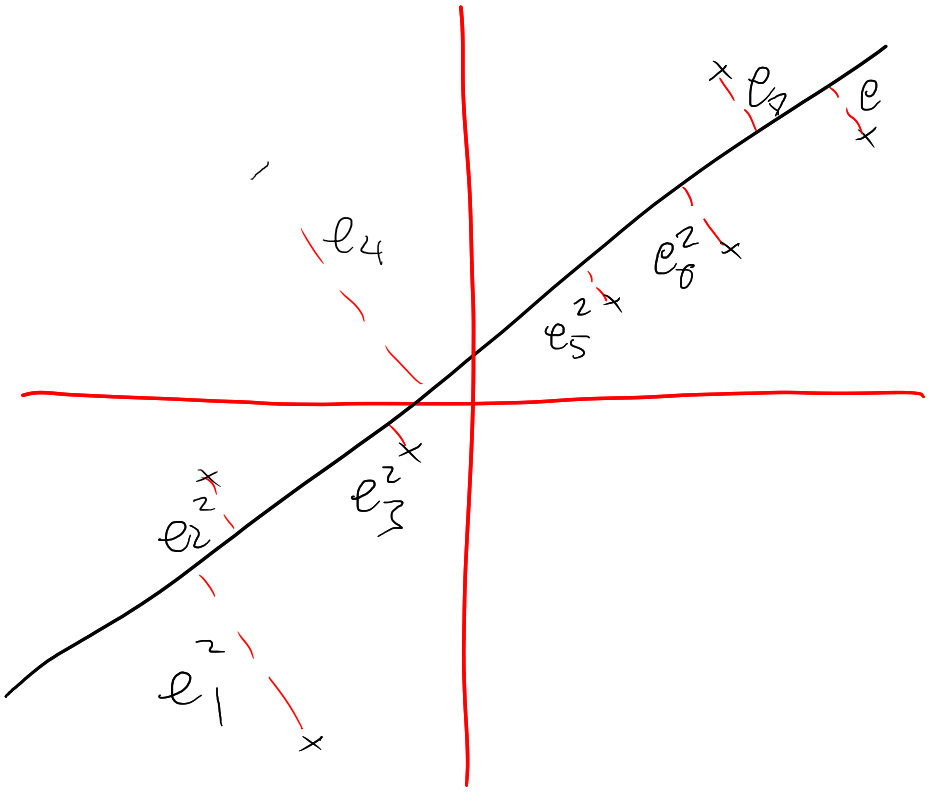

$$O(p) = \sum_{k=1}^{K} e_k^2$$



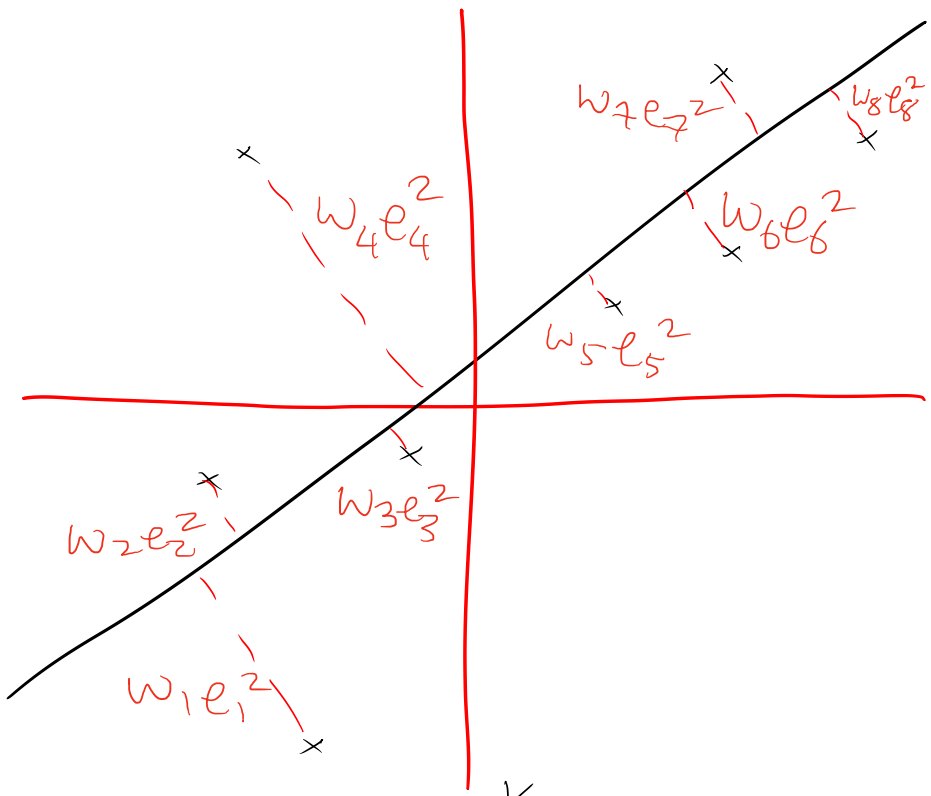

$$O(p) = \sum_{k=1}^{K} \rho(e_k)$$

contribution depends on $e_k$

# "easier" problem: weighted least squares (WLS)



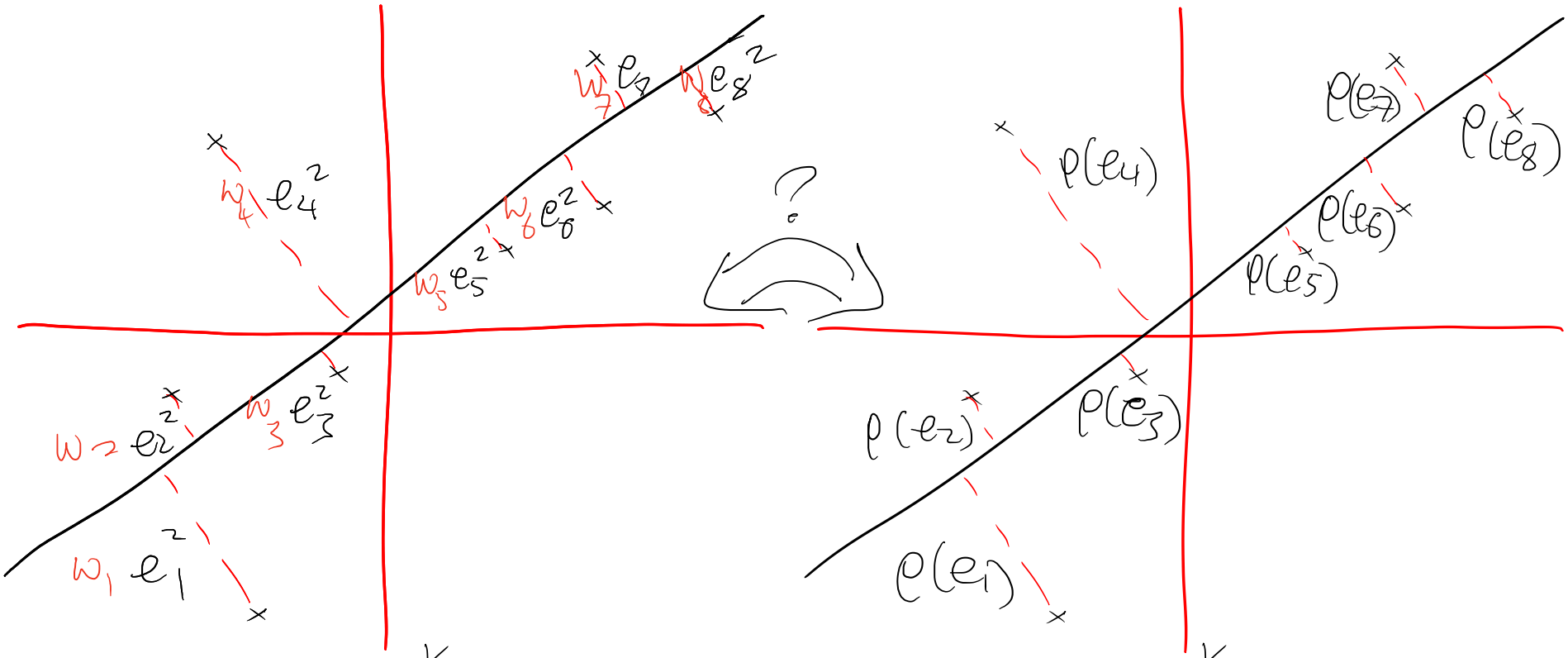

$$O(p) = \sum_{k=1}^{K} e_k^2$$




$$O(p) = \sum_{k=1}^{K} w_k e_k^2$$

depends on position k but NOT $e_k$

# idea: treat M-estimation as a form of WLS



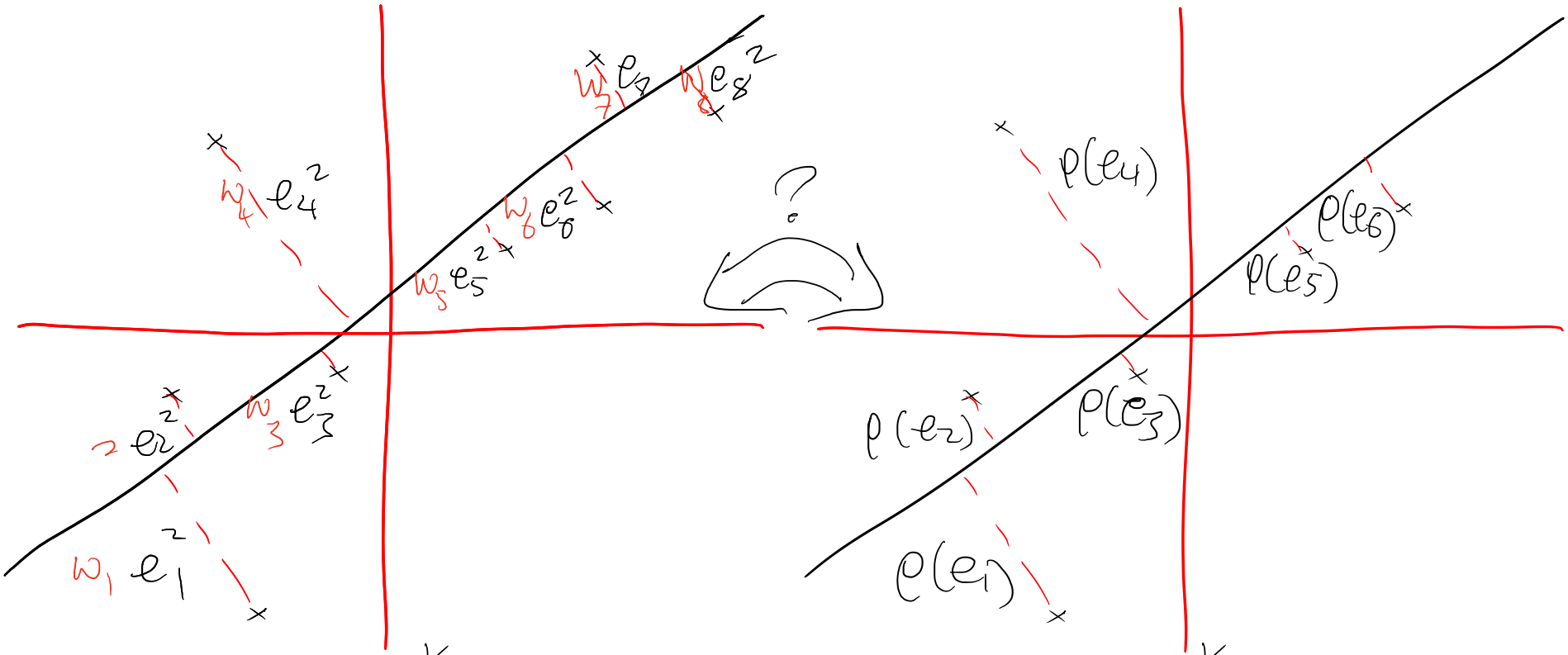

$$O(p) = \sum_{k=1}^{K} W_k e_k^2$$

depends on position k but NOT $e_k$

$$O(p) = \sum_{k=1}^{K} \rho(e_k)$$

contribution depends on $e_k$

# idea: treat M-estimation as a form of WLS



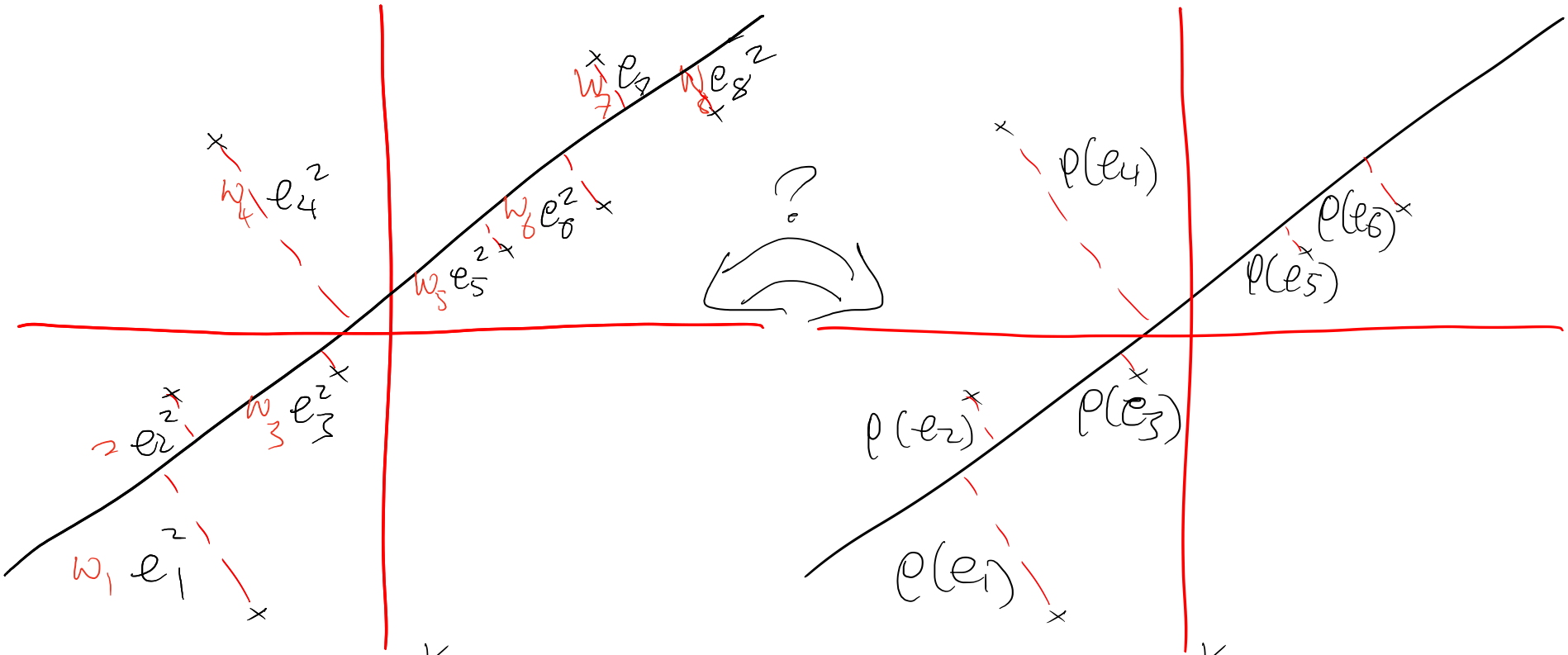

$$O(p) = \sum_{k=1}^{K} w_k e_k^2$$

$$\frac{\partial O}{\partial p} = \sum_{k=1}^{K} 2 w_k e_k \frac{\partial e_k}{\partial p}$$

$$O(p) = \sum_{k=1}^{K} \rho(e_k)$$

$$\frac{\partial O}{\partial p} = \sum_{k=1}^{K} \frac{\partial \rho}{\partial e_k} \cdot \frac{\partial e_k}{\partial p}$$

# idea: treat M-estimation as a form of WLS




$$O(\rho) = \sum_{k=1}^{K} w_k e_k^2$$

$$\frac{\partial O}{\partial \rho} = \sum_{k=1}^{K} 2 w_k e_k \frac{\partial e_k}{\partial \rho}$$

$$O(\rho) = \sum_{k=1}^{K} \rho(e_k)$$

$$\frac{\partial O}{\partial \rho} = \sum_{k=1}^{K} \frac{\partial \rho}{\partial e_k} \frac{\partial e_k}{\partial \rho}$$

# the weight function

Define

$$w_k = \frac{1}{e_k}\frac{\partial \rho}{\partial e_k}(e_k)$$

$$= \frac{1}{e_k}\psi(e_k)$$

influence function

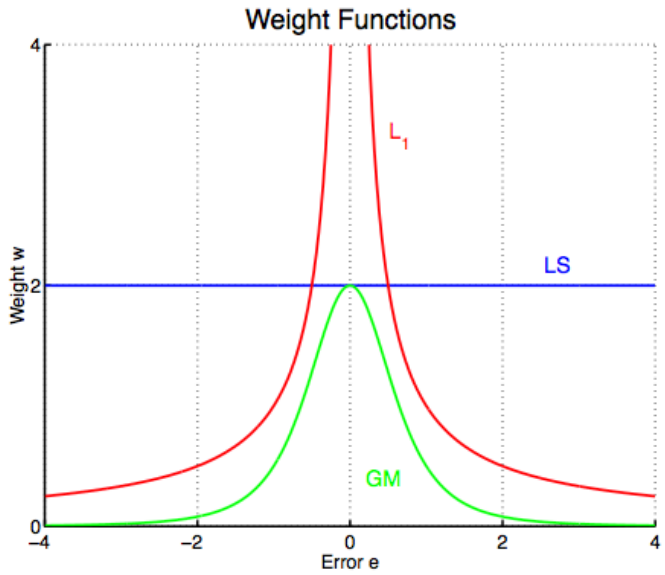
$$O(\rho) = \sum_{k=1}^{K} w_k e_k^2$$

$$\frac{\partial O}{\partial \rho} = \sum_{k=1}^{K} 2 w_k e_k \frac{\partial e_k}{\partial \rho}$$

$$O(\rho) = \sum_{k=1}^{K} \rho(e_k)$$

$$\frac{\partial O}{\partial \rho} = \sum_{k=1}^{K} \frac{\partial \rho}{\partial e_k}\frac{\partial e_k}{\partial \rho}$$

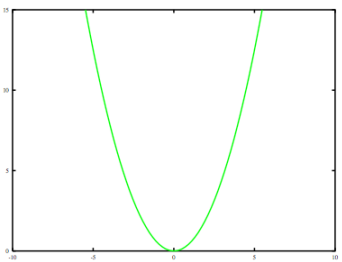
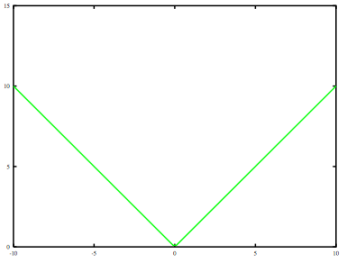
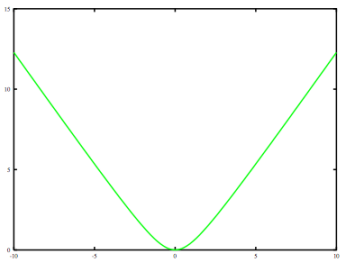
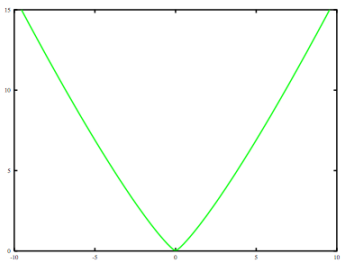
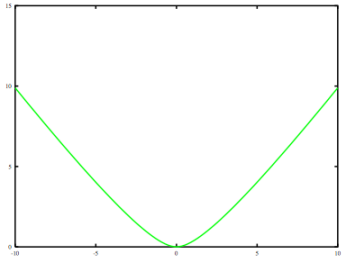
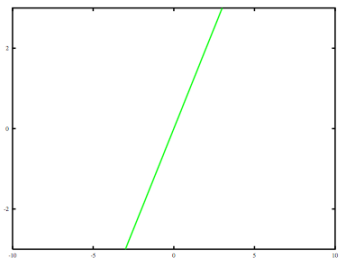
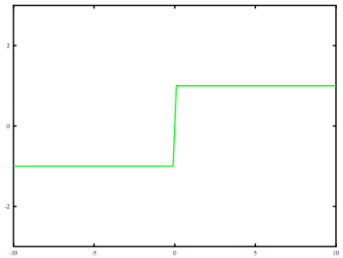
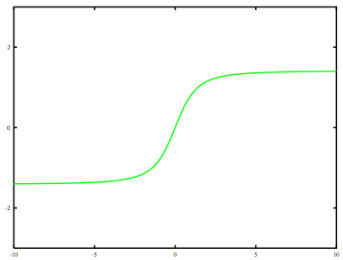
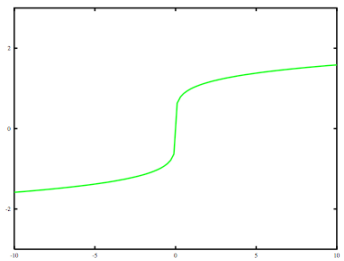
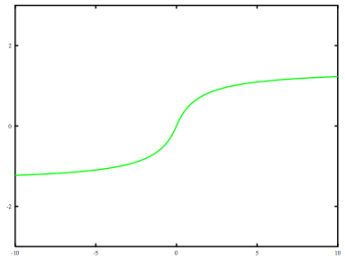
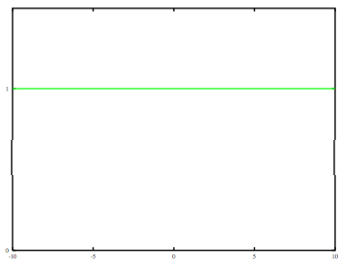
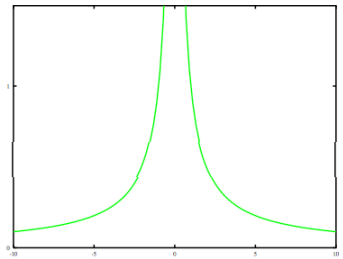
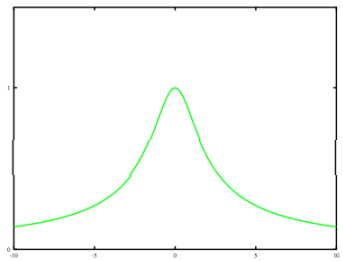
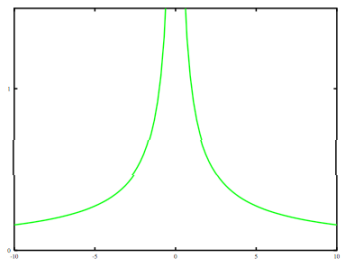
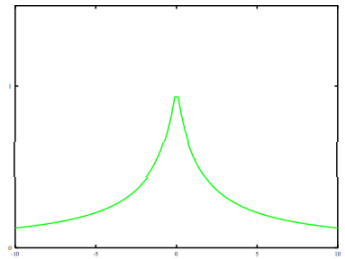
# the weight function




*Remarks:*

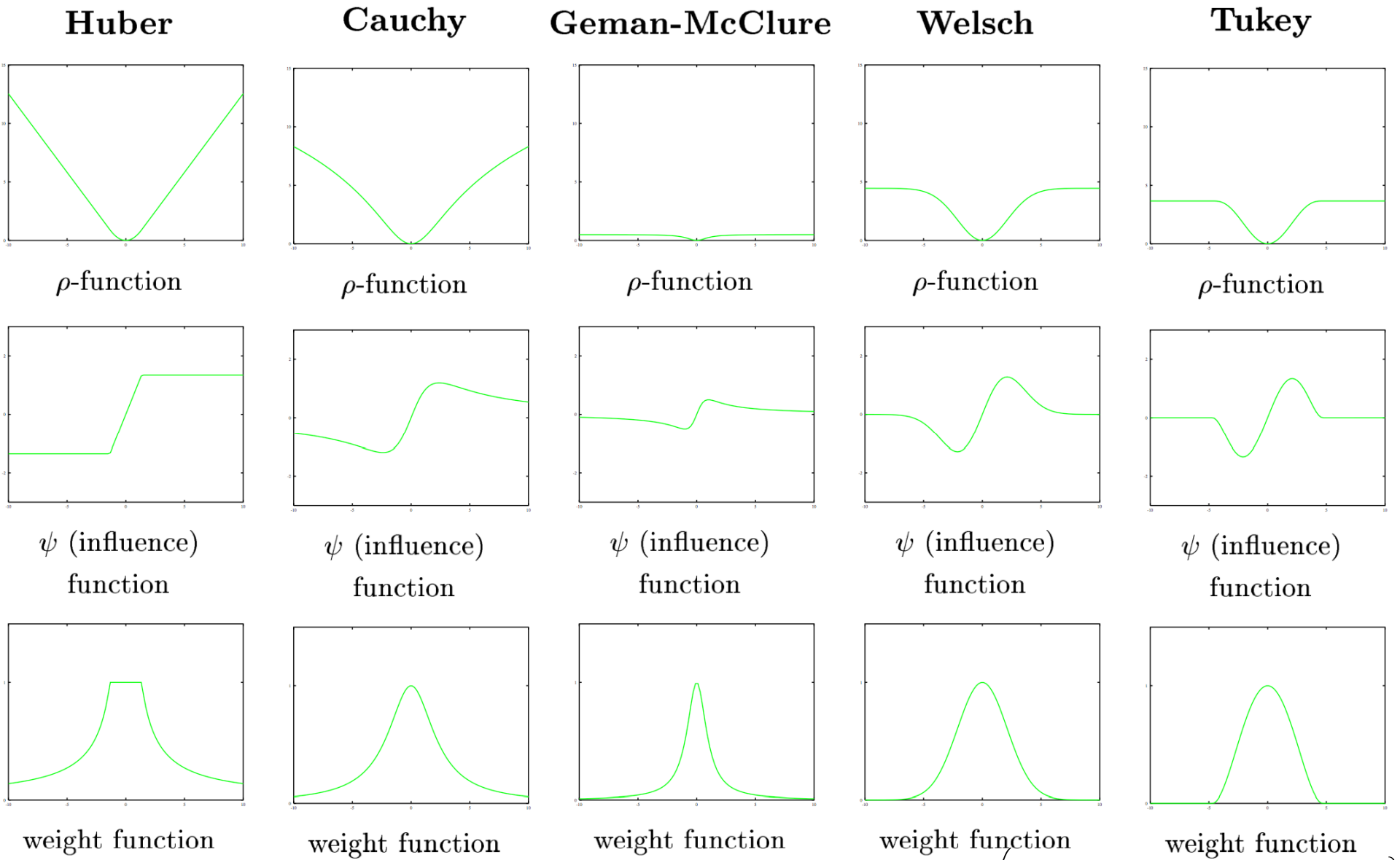
- For $\rho(e) = e^2$, the weights are constant, $w = 2$, and (17) is just LS.
- If the weights were independent of $\vec{\theta}$, then (17) would be the gradient of a *weighted* LS estimator.
- For robust estimators, for bounded weights, Eqn (17) has the form of *weighted* LS, but it is nonlinear as the weights depend on $\vec{\theta}$.

# example influence functions & weight functions

| **Least-squares** | **Least-absolute** | $L_1 - L_2$ | **Least-power** | **Fair** |
|---|---|---|---|---|
|  |  |  |  |  |
| $\rho$-function | $\rho$-function | $\rho$-function | $\rho$-function | $\rho$-function |
|  |  |  |  |  |
| $\psi$ (influence) function | $\psi$ (influence) function | $\psi$ (influence) function | $\psi$ (influence) function | $\psi$ (influence) function |
|  |  |  |  |  |
| weight function | weight function | weight function | weight function | weight function |

(Z. Zhang, IVC 1997)

# example influence functions & weight functions




(Z. Zhang, IVC 1997)

# examples of M-estimators & influence functions

Table 1: A few commonly used M-estimators

| type | $\rho(x)$ | $\psi(x)$ | $w(x)$ |
|---|---|---|---|
| $L_2$ | $x^2/2$ | $x$ | $1$ |
| $L_1$ | $\lvert x\rvert$ | $\mathrm{sgn}(x)$ | $\frac{1}{\lvert x\rvert}$ |
| $L_1 - L_2$ | $2(\sqrt{1+x^2/2}-1)$ | $\frac{x}{\sqrt{1+x^2/2}}$ | $\frac{1}{\sqrt{1+x^2/2}}$ |
| $L_p$ | $\frac{\lvert x\rvert^\nu}{\nu}$ | $\mathrm{sgn}(x)\lvert x\rvert^{\nu-1}$ | $\lvert x\rvert^{\nu-2}$ |
| "Fair" | $c^2[\frac{\lvert x\rvert}{c} - \log(1+\frac{\lvert x\rvert}{c})]$ | $\frac{x}{1+\lvert x\rvert/c}$ | $\frac{1}{1+\lvert x\rvert/c}$ |

(Z. Zhang, IVC 1997)

# examples of M-estimators & influence functions

| type | $\rho(x)$ | $\psi(x)$ | $w(x)$ |
|---|---|---|---|
| Huber $\begin{cases}\text{if } \lvert x\rvert \le k\\ \text{if } \lvert x\rvert \ge k\end{cases}$ | $\begin{cases}x^2/2\\ k(\lvert x\rvert - k/2)\end{cases}$ | $\begin{cases}x\\ k\,\mathrm{sgn}(x)\end{cases}$ | $\begin{cases}1\\ k/\lvert x\rvert\end{cases}$ |
| Cauchy | $\frac{c^2}{2}\log(1+(x/c)^2)$ | $\frac{x}{1+(x/c)^2}$ | $\frac{1}{1+(x/c)^2}$ |
| Geman-McClure | $\frac{x^2/2}{1+x^2}$ | $\frac{x}{(1+x^2)^2}$ | $\frac{1}{(1+x^2)^2}$ |
| Welsch | $\frac{c^2}{2}[1-\exp(-(x/c)^2)]$ | $x\exp(-(x/c)^2)$ | $\exp(-(x/c)^2))$ |
| Tukey $\begin{cases}\text{if } \lvert x\rvert \le c\\ \text{if } \lvert x\rvert > c\end{cases}$ | $\begin{cases}\frac{c^2}{6}\left(1-[1-(x/c)^2]^3\right)\\ [0.2cm](c^2/6)\end{cases}$ | $\begin{cases}x[1-(x/c)^2]^2\\ 0\end{cases}$ | $\begin{cases}[1-(x/c)^2]^2\\ 0\end{cases}$ |

(Z. Zhang, IVC 1997)

# iteratively reweighted least squares (IRLS)

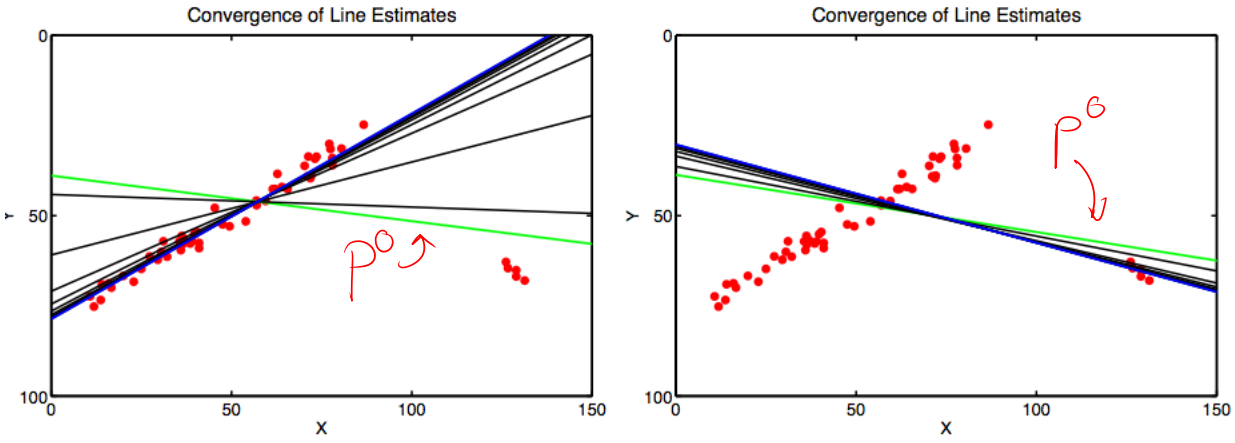
idea: we take a step along the gradient direction $\frac{\partial O}{\partial p}$ by solving a weighted least squares problem

algorithm

0. start with an initial guess $p^0 = \begin{bmatrix} \vec{n}_0 \\ c_0 \end{bmatrix}$
1. compute the weights $w_k^i = w(x_k^T p^{i-1})$
2. solve the WLS problem for $w_k^i, x_k$ to obtain line parameters $p^i$
3. repeat until convergence (ie $p^i - p^{i-1}$ is small)

# example results

For the GM estimator (with $\sigma$ equal to the standard deviation of the noise for the edgels near the dominant line) the iterations of the IRLS algorithm are shown below (black lines). The initial guesses (green) and the converged states (blue) are also shown.

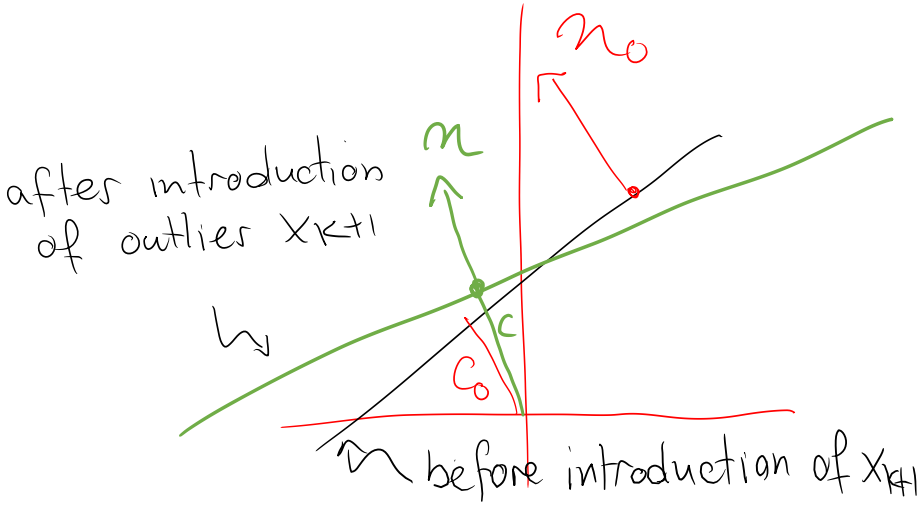



The solution on the left indicates the outliers have been rejected, while the iterations on the right show the algorithm converging to one of several undesirable local minimum. How can we avoid these?

# analysis of influence

Given a solution $(\vec{n}_0, c_0)$ of (14), we consider the effect of one additional outlier $\vec{x}_{K+1}$. We wish to estimate the perturbation to the solution caused by this outlier. The size of this perturbation will determine the *influence* of that outlier.

To simplify the analysis, we freeze the weights for the original data, $w_k = w(\vec{n}_0^T \vec{x}_k + c_0)$ for $k = 1, \ldots, K$, and set $w_{K+1} = w(\vec{n}_0^T \vec{x}_{K+1} + c_0)$. With all these weights frozen, we seek the solution of the weighted least squares problem



# analysis of influence

Given a solution $(\vec{n}_0, c_0)$ of (14), we consider the effect of one additional outlier $\vec{x}_{K+1}$. We wish to estimate the perturbation to the solution caused by this outlier. The size of this perturbation will determine the *influence* of that outlier.

To simplify the analysis, we freeze the weights for the original data, $w_k = w(\vec{n}_0^T \vec{x}_k + c_0)$ for $k = 1, \ldots, K$, and set $w_{K+1} = w(\vec{n}_0^T \vec{x}_{K+1} + c_0)$. With all these weights frozen, we seek the solution of the weighted least squares problem

$$\min \mathcal{O}_{WLS}(\vec{n}, c, \epsilon), \quad \text{for } ||\vec{n}|| = 1, \tag{25}$$

$$\mathcal{O}_{WLS}(\vec{n}, c, \epsilon) \equiv \left[ \sum_{k=1}^{K} \frac{w_k}{2} (\vec{n}^T \vec{x}_k + c)^2 \right] + \epsilon \frac{w_{K+1}}{2} (\vec{n}^T \vec{x}_{K+1} + c)^2.$$

controls influence of (K+1)-th outlier

$\epsilon = 0 \Longrightarrow$ original IRLS solution

$\epsilon = 1 \Longrightarrow$ first update step of IRLS

# influence of outliers on line parameters

**Theorem.** Let $\psi_{K+1} = \psi(\vec{n}_0^T \vec{x}_{K+1} + c_0)$ be the influence function evaluated at $\vec{x}_{K+1}$, and $\vec{t}_0$ be a unit vector tangent to the initial line (i.e., $\vec{t}_0 \perp \vec{n}_0$ ). Then the optimal solution $(\vec{n}(\epsilon), c(\epsilon))$ of (25) satisfies

$$\left.\frac{dc}{d\epsilon}\right|_{\epsilon=0} = -\frac{\psi_{K+1}}{s}, \tag{27}$$

$$\left.\frac{d\vec{n}}{d\epsilon}\right|_{\epsilon=0} = -\frac{\psi_{K+1}}{(\mu_1 - \mu_2)s}\vec{t}_0\vec{t}_0^T(\vec{x}_{K+1} - \vec{m}), \tag{28}$$

proportional to $\psi_{K+1}$

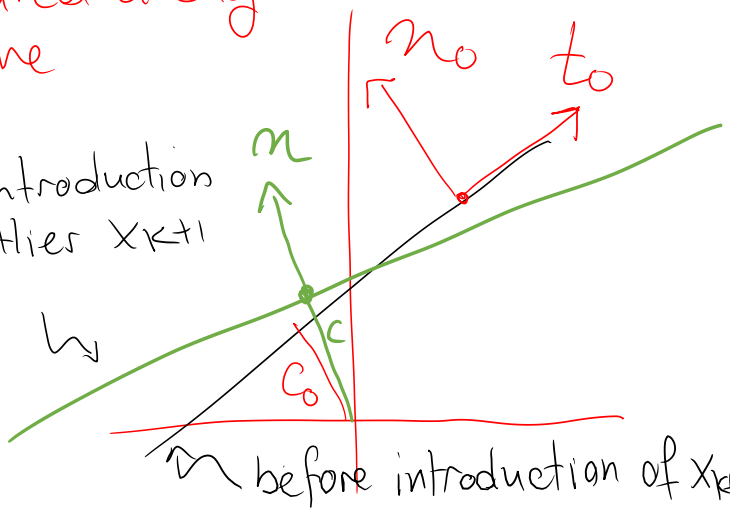
difference of eigenvalues measures eccentricity of data points

distance of $x_{K+1}$ from mean, measured along the line

$\vec{n}$ may not be stable even for bounded $\psi(\cdot)$

vs.





# influence of outliers on line parameters

**Theorem.** Let $\psi_{K+1} = \psi(\vec{n}_0^T \vec{x}_{K+1} + c_0)$ be the influence function evaluated at $\vec{x}_{K+1}$, and $\vec{t}_0$ be a unit vector tangent to the initial line (i.e., $\vec{t}_0 \perp \vec{n}_0$ ). Then the optimal solution $(\vec{n}(\epsilon), c(\epsilon))$ of (25) satisfies

$$\left.\frac{dc}{d\epsilon}\right|_{\epsilon=0} = -\frac{\psi_{K+1}}{s}, \quad (27)$$

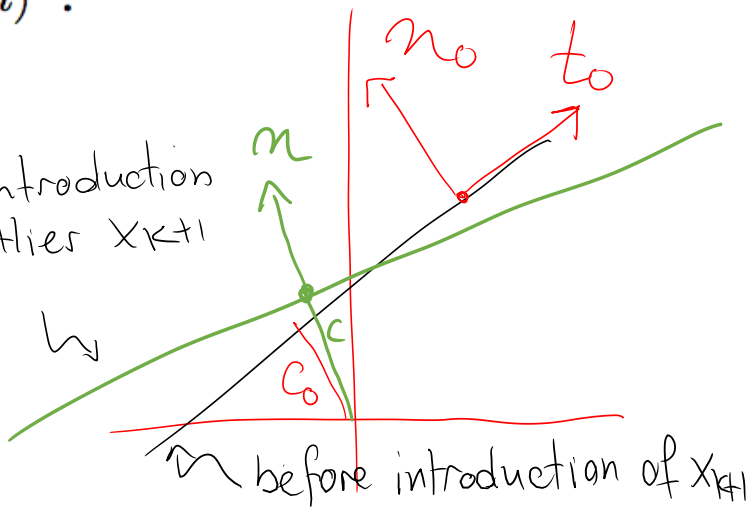
$$\left.\frac{d\vec{n}}{d\epsilon}\right|_{\epsilon=0} = -\frac{\psi_{K+1}}{(\mu_1-\mu_2)s}\vec{t}_0\vec{t}_0^T(\vec{x}_{K+1}-\vec{m}), \quad (28)$$

where $s \equiv \sum_{k=1}^K w_k$, $\vec{m} = \frac{1}{s}\sum_{k=1}^K w_k\vec{x}_k$, and $\mu_1 > \mu_2$ are the two eigenvalues of $C = \sum_{k=1}^K w_k(\vec{x}_k - \vec{m})(\vec{x}_k - \vec{m})^T$.

proportional to $\psi_{K+1}$

$c$ stable when $\psi()$ bounded

$\vec{n}$ may not be stable even for bounded $\psi()$



# influence of outliers on line parameters

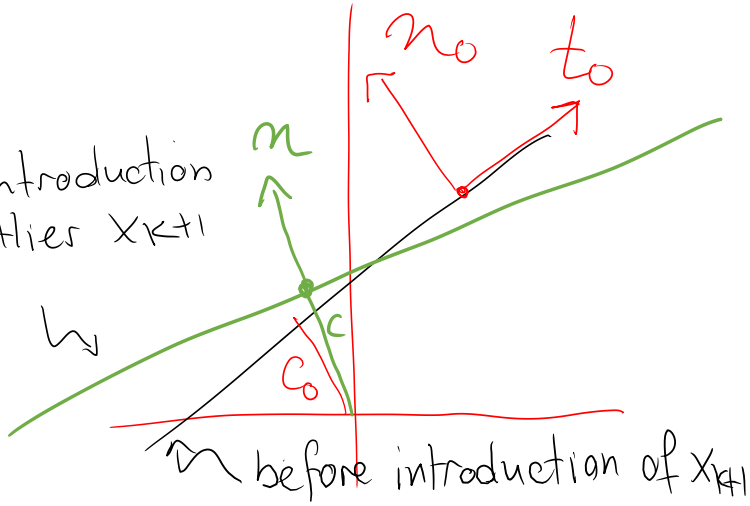
**Theorem.** Let $\psi_{K+1} = \psi(\vec{n}_0^T \vec{x}_{K+1} + c_0)$ be the influence function evaluated at $\vec{x}_{K+1}$, and $\vec{t}_0$ be a unit vector tangent to the initial line (i.e., $\vec{t}_0 \perp \vec{n}_0$ ). Then the optimal solution $(\vec{n}(\epsilon), c(\epsilon))$ of (25) satisfies

$$\left.\frac{dc}{d\epsilon}\right|_{\epsilon=0} = -\frac{\psi_{K+1}}{s}, \tag{27}$$

$$\left.\frac{d\vec{n}}{d\epsilon}\right|_{\epsilon=0} = -\frac{\psi_{K+1}}{(\mu_1 - \mu_2)s} \vec{t}_0 \vec{t}_0^T (\vec{x}_{K+1} - \vec{m}), \tag{28}$$

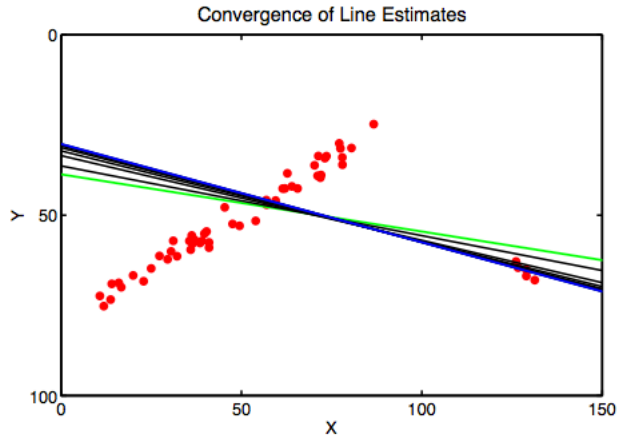
$\vec{n}$ may not be stable even for bounded $\psi(\,)$

points where this is large are called leverage points



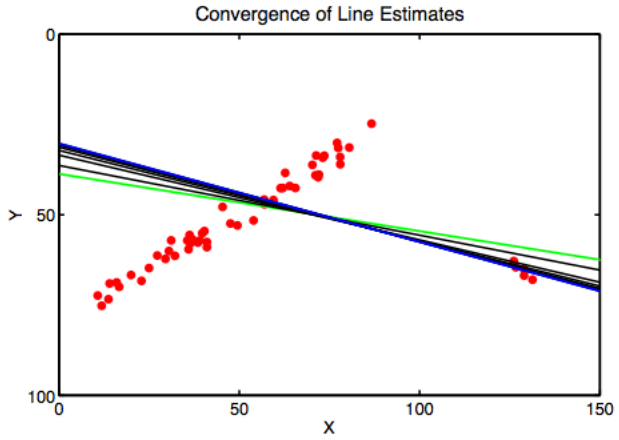
# controlling leverage

The previous Theorem shows that even a redescending estimator, such as GM, is not robust to leverage points.




In this example (from p. 23) the estimator fails to reject a cluster of outliers in favour of points on the dominant line. For some initial guesses, this cluster has enough leverage to unduly influence the solution.

# controlling leverage

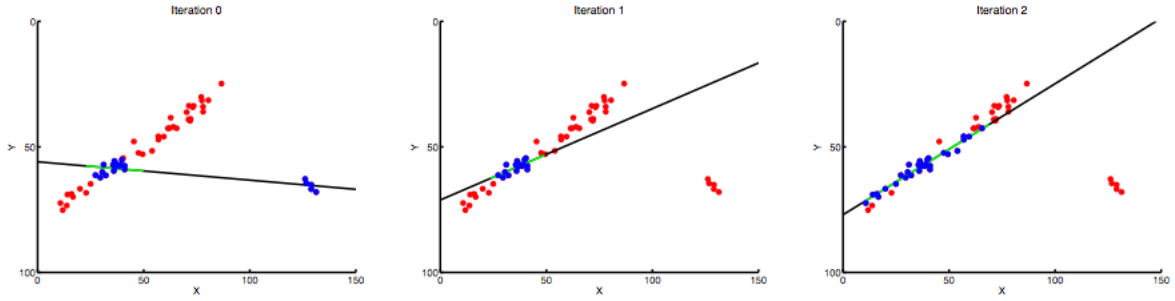



A simple strategy for dealing with leverage points is:

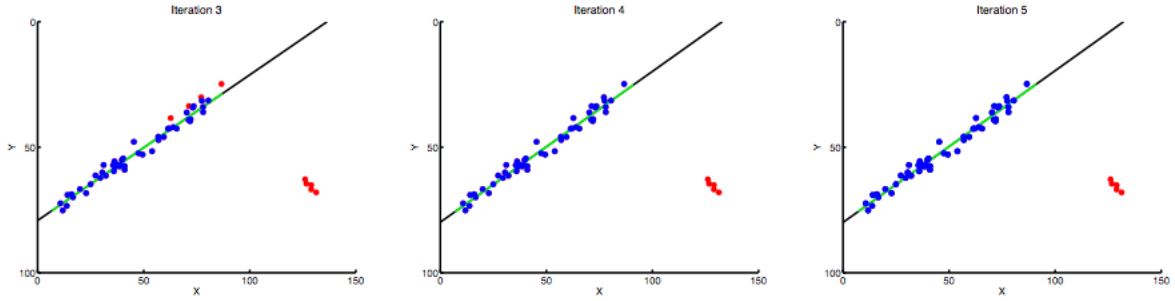
1. Estimate the distribution of data support along the fitted line (eg. project the weights $w_k$ from $\vec{x}_k$ onto the line and blur).
2. Determine a contiguous region of support along the fitted line, that is, an interval of support without any large gaps.
3. Reduce the weights $w_k$ in the IRLS algorithm for points $\vec{x}_k$ significantly outside the region of contiguous support (eg. set such weights to $0$).

# leverage example

We can reduce leverage problems by controlling the support interval within the robust line estimation algorithm.

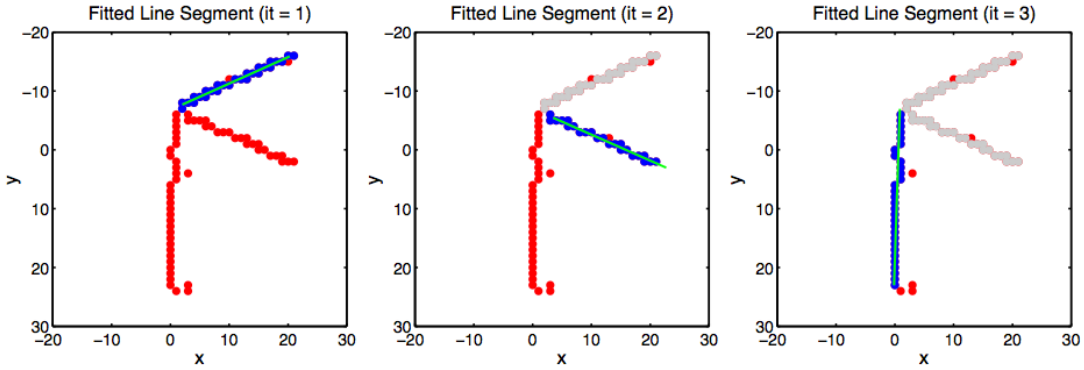



Iterations 0 to 2 above. Current line estimate (black). Edgels with significant weights (blue). The estimate for a contiguous region of support is also shown (green segment).

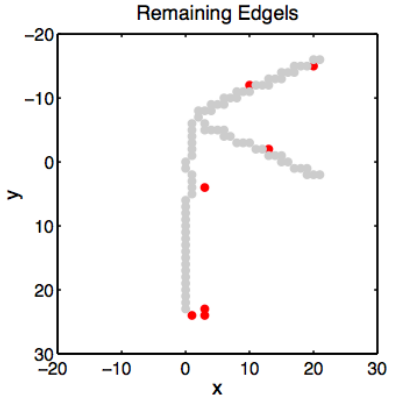



The remaining iterations 3 to 5, at which point the algorithm has converged.

# simple algorithm for finding multiple lines




The first fitted segment is shown (left). Edgels having significant robust weights (blue), can then be removed from the active data set (red). A second line segment is fit (middle) to the remaining edgels, and the new supporting edgels (blue) are removed. Then a third segment is fit.

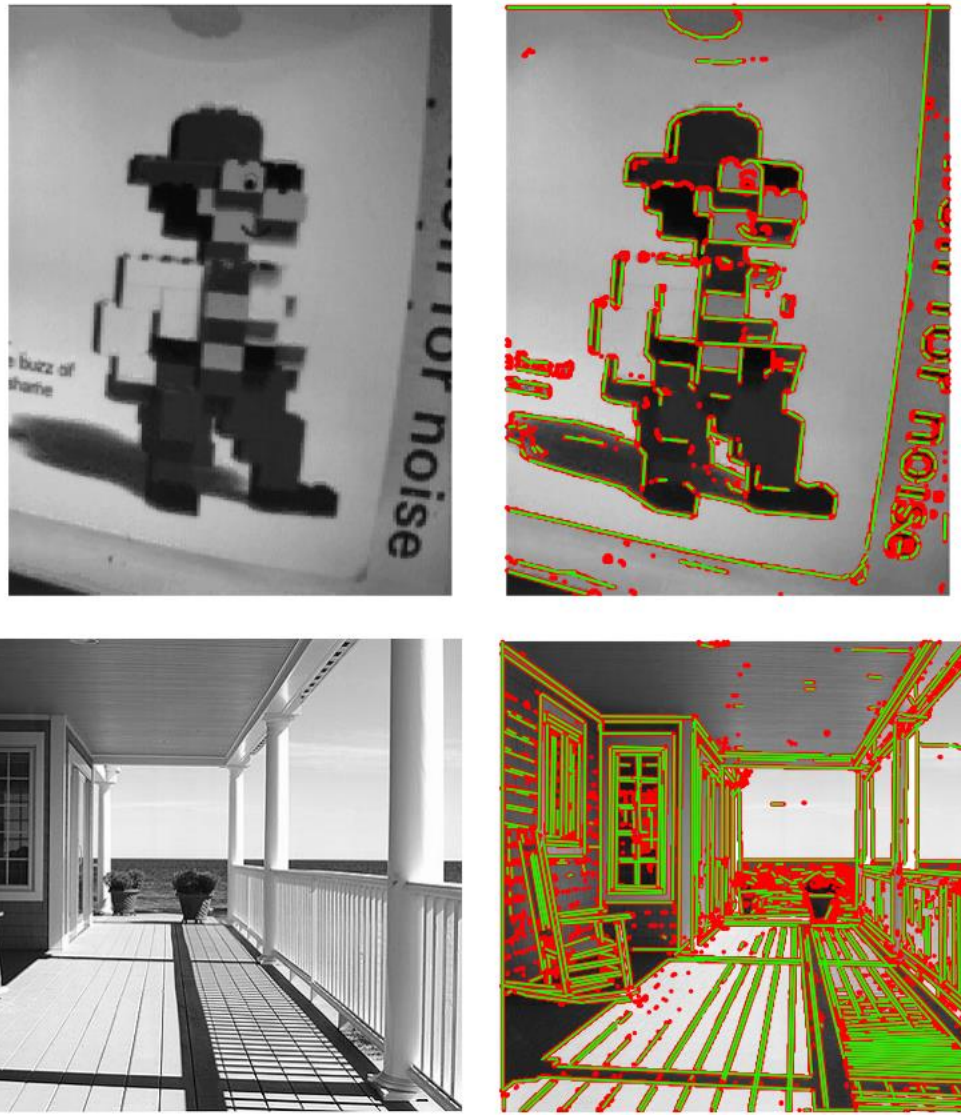



This fitting process continues until no further lines with contiguous support regions can be fit. The remaining active edgels are outliers.

# recap: robust line estimation algorithm

1. Initial Guess.
   - Randomly sample from the data. Eg. Set $(\vec{n}_0, c_0)$ according the the position and orientation of a sampled edgel.
2. Iterative Fitting.
   - Use the iteratively reweighted least squares algorithm to fit the line parameters. Maintain information about the support (from the data) for the line along its length, and use it to downweight data likely to cause leverage problems.
3. Verification.
   - Given a converged solution, decide whether that line has sufficient data support (e.g., consider the sum of the weights). Discard solutions with insufficient support.
4. Model Selection.
   - For each verified line segment, remove the edgels that provide significant support for it from the data set.

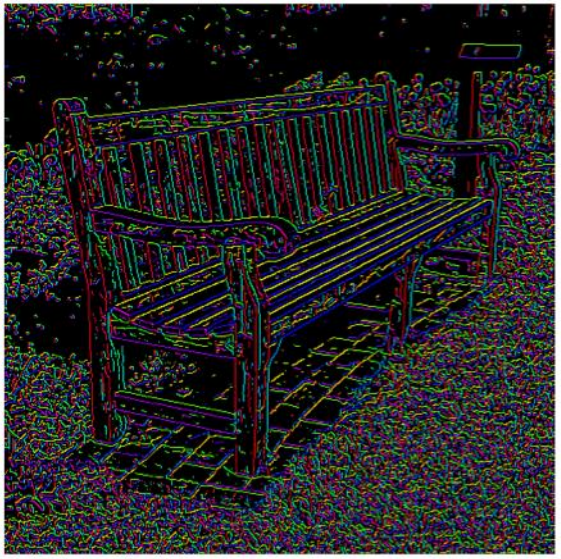
# example results



# example results

Parkbench Image

Colour-coded Edgel Orientation



matrix $S_p$ captures distribution of gradient orientations around p

$$S_p = \sum_{q \in N(p)} \nabla I_q (\nabla I_q)^T$$

↑ neighborhood of pixel p

Orientation Tensor

Estimated Lines

# types of errors

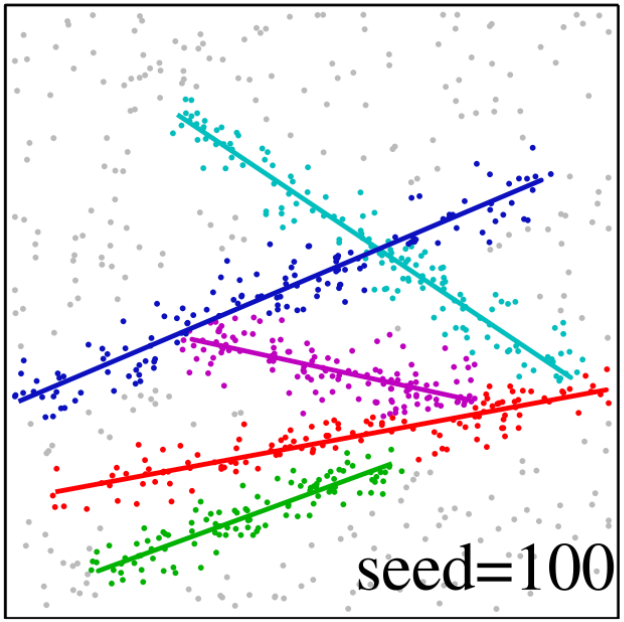
- **Drop-outs:** Parts or all of a true line that are missing in the estimation results.
- **False positives:** An estimated line which does not correspond to any combination of true lines.
- **Over-segmentation:** Breaking one true line into several estimated segments (possibly colinear, parallel or otherwise).
- **Under-segmentation:** Joining two or more true lines into one estimated segment.
- **Parameter noise:** Small errors in the position and orientation estimates.
- **Model-type errors:** The somewhat inappropriate use of our current model (i.e., line segments fitted to edgels) for curves, texture, or thin bars in the image. More generally, model-type errors occur when the system can fit several types of models (eg. curves versus lines), but selects the wrong choice.

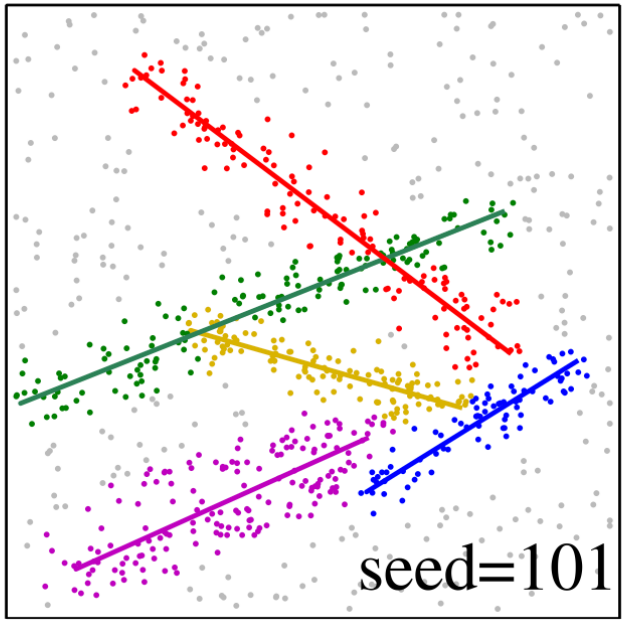
# Topic 01:

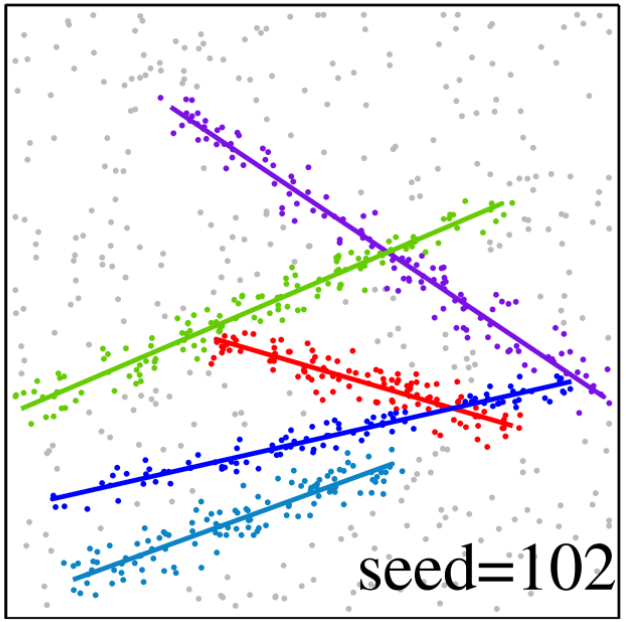
# Robust Estimation Basics

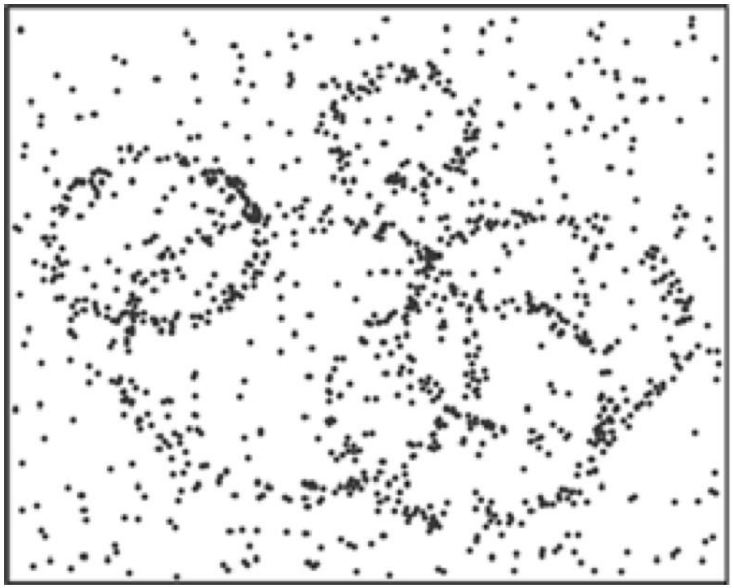
- least squares & total least squares
- robust M-estimators
- optimizing robust objective functions
- proposal generation: Hough transforms
- proposal generation: RANSAC

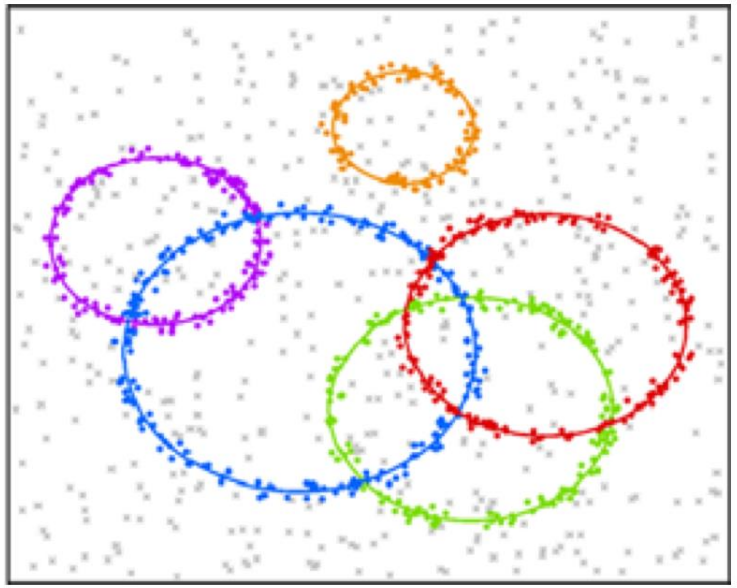
# motivation



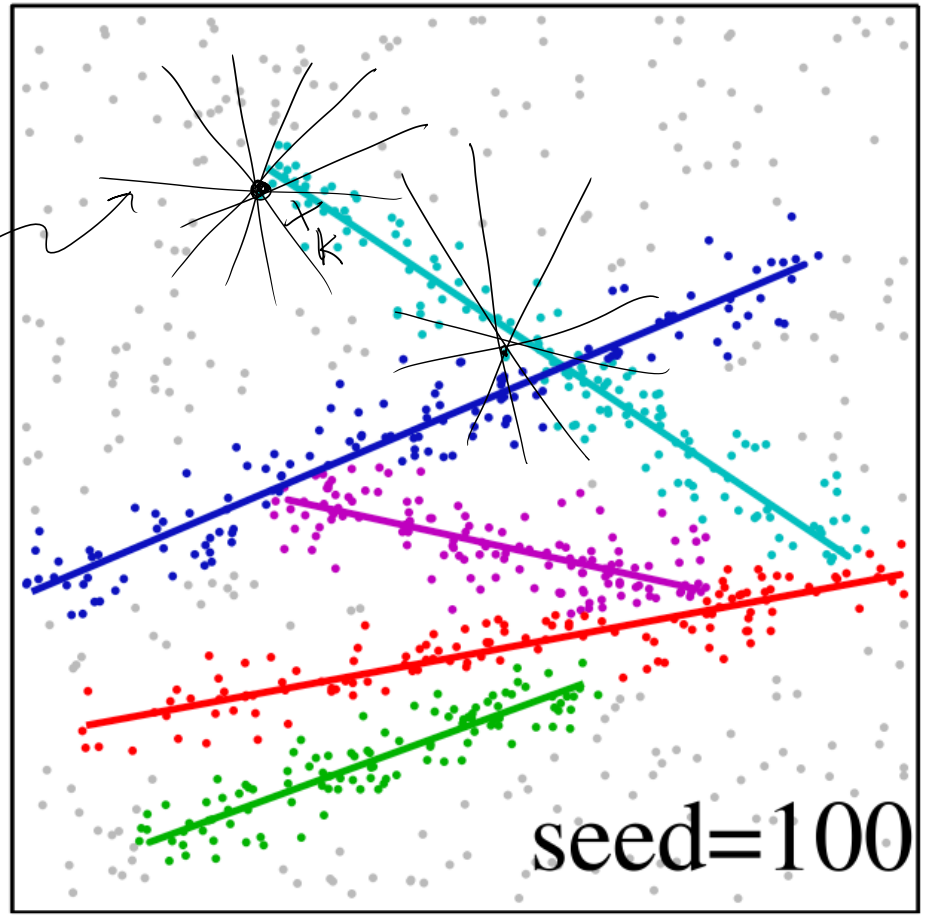





# voting methods & the Hough transform

each measurement "votes" for all parameter setting that are consistent with it

votes cast for all lines through $x_k$



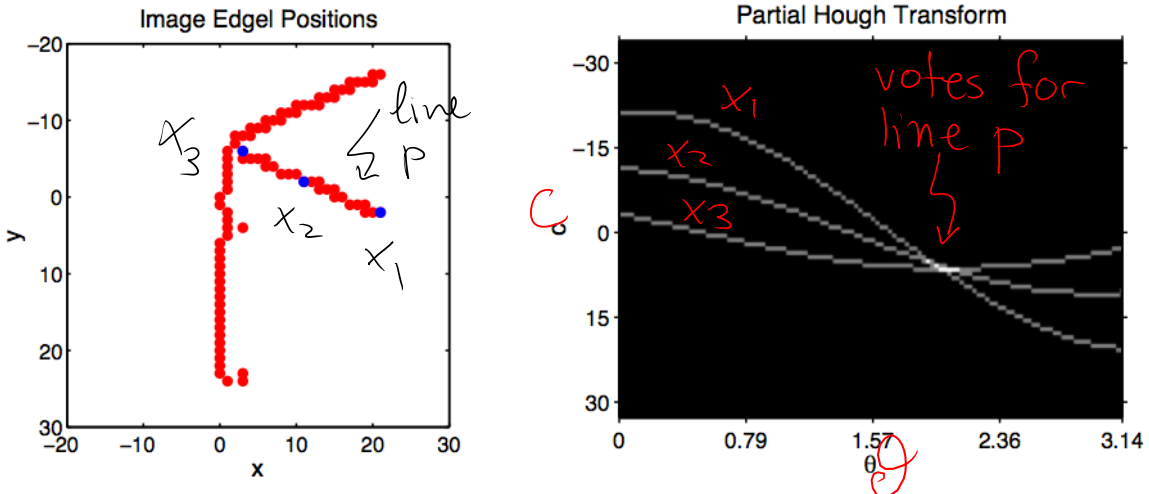
# voting methods & the Hough transform

each measurement "votes" for all parameter setting that are consistent with it

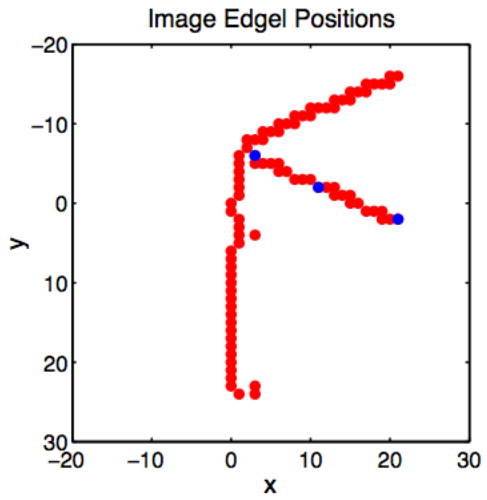
For our working example, image lines $\vec{n}^T(\theta)\vec{x} + c = 0$ are parameterized by $\theta$ and $c$, with $\vec{n}(\theta) = (\cos(\theta), \sin(\theta))^T$.

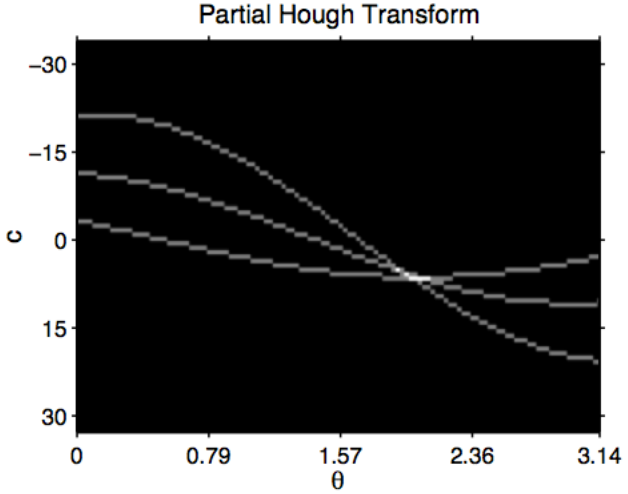
A discrete voting space is formed by quantizing $\theta \in [0, \pi)$ and $c$. For each edgel $\vec{x}_k$, and each discrete $\theta_i$, we add one vote to the bin $(\theta_i, c_j)$ which has minimal absolute error, that is

$$\vec{n}^T(\theta_i)\,\vec{x}_k + c_j \;\in\; [-\delta c/2, \delta c/2). \qquad (13)$$

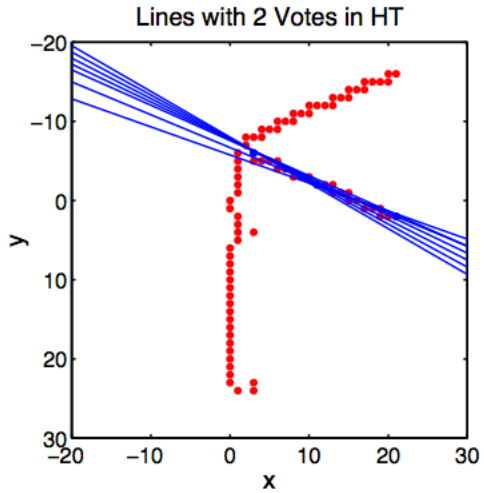


# Hough example: voting for lines



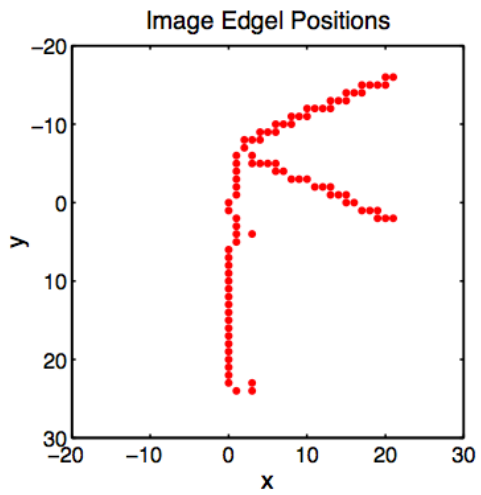



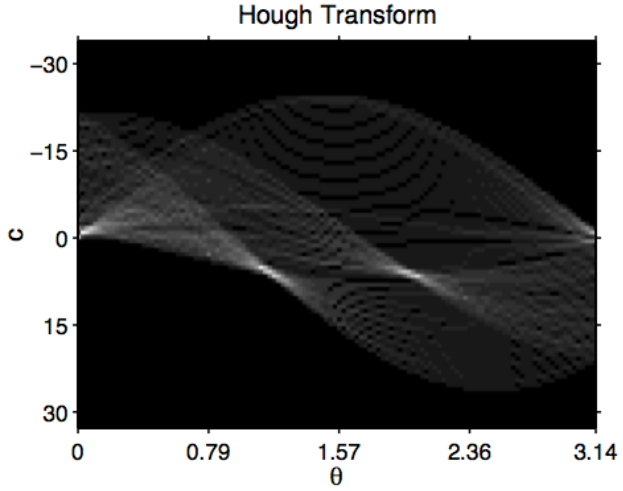

The maximum number of votes here is 2, not 3, since the three edgels are not precisely colinear. The centers of the bins with 2 votes provide the parameters for blue lines below.



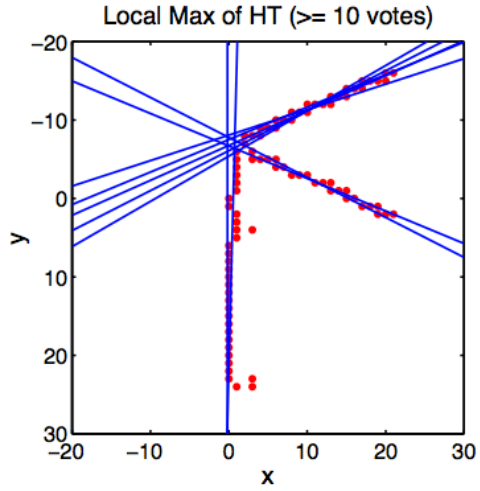

Increasing the bin size $\delta c$ would allow one (or more) bins to have votes from all three edgels. In practice, selecting appropriate bin sizes for the HT can be tricky.

# Hough example: vote accumulation




Three strong peaks are visible in the HT, corresponding roughly to the three line segments apparent in the data. (Enlarge the neighbourhood of these peaks when viewing an electronic copy.)



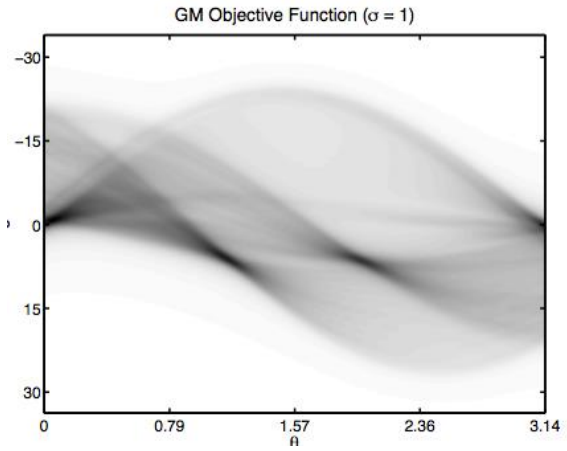

The blue lines result from local non-max suppression of the HT above, then thresholding the results at 10 votes. Results are sensitive to the particular values used for this threshold and the bin sizes. Good values for these parameters often depend on the image data.

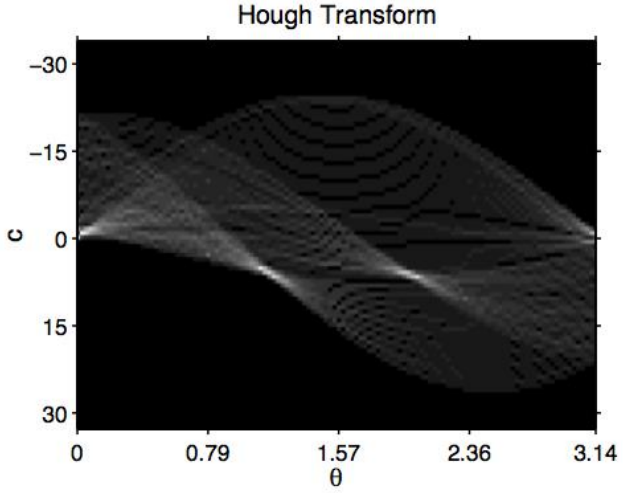
# Hough transform: key issues

The Hough transform is a reasonable approach for simple data, for which clear peaks can be expected. However,

- **Goldilocks' problem:** It is difficult to set appropriate bin sizes:
  - **Too large:** Poor resolution of parameters.
  - **Too small:** Peaks in HT broken into pieces, each with fewer votes.
  - **Just right:** Isolated peaks at appropriate parameter values.
- Poor detector performance (i.e. the trade-off between false positives and false negatives). In noisy and/or cluttered datasets many extraneous bins can have vote counts comparable to, or larger than, the counts for the desired models.
- In practice, fractional votes for neighbouring Hough bins are also included to help smooth the HT and reduce discretization artifacts.
- The number of bins grows exponentially with the dimension of the unkown parameters (eg. $n^d$ for $n$ bins in each of $d$ dimentions).

# Hough transform's relation to M-estimation




The robust objective function $\mathcal{O}(\theta, c)$ is closely related to a smoothed Hough transform (e.g., see the figures on p. 10 and 15).

# Topic 01:

# Robust Estimation Basics

- least squares & total least squares
- robust M-estimators
- optimizing robust objective functions
- proposal generation: Hough transforms
- proposal generation: RANSAC

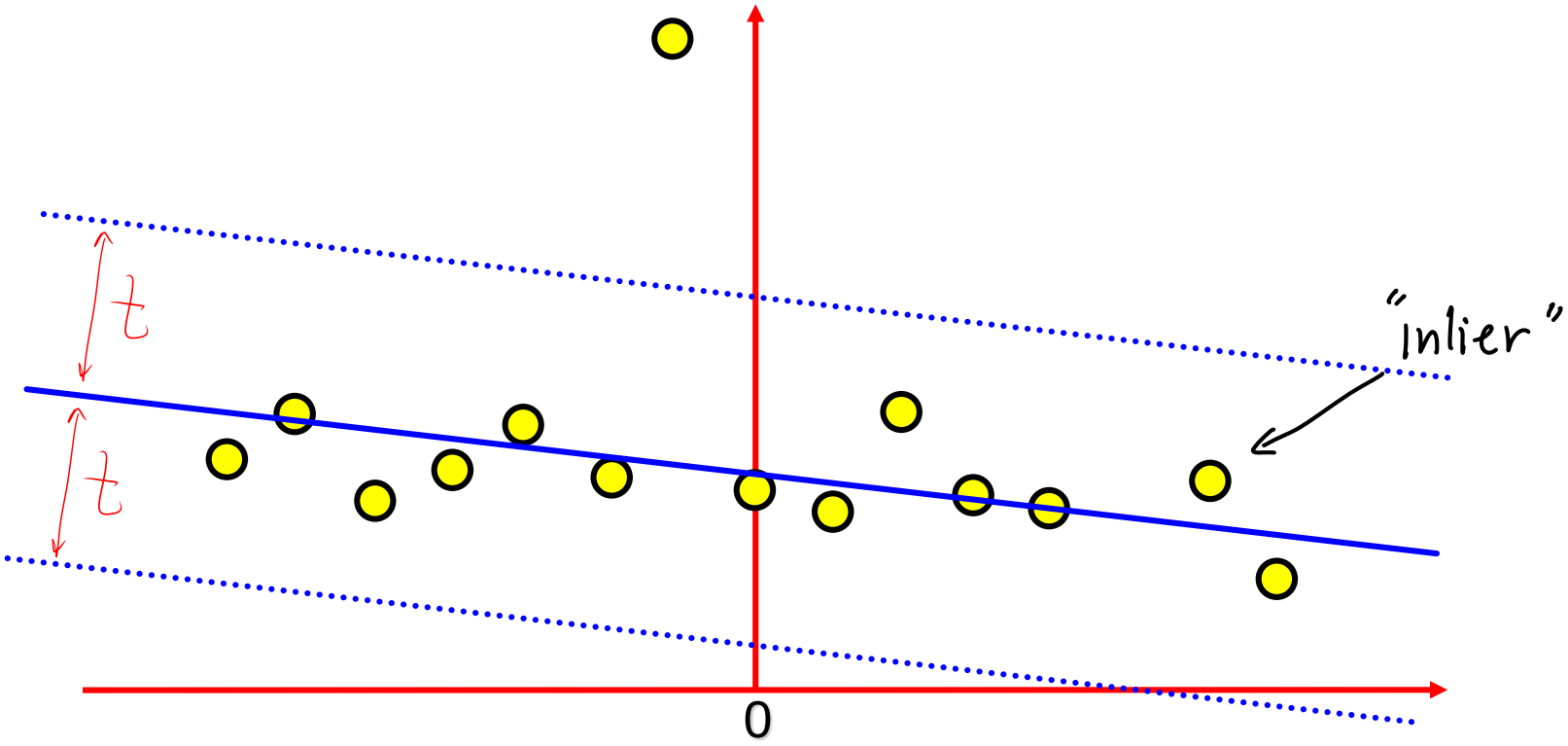
# robust parameter estimation with RANSAC

given:

- n = # model parameters
- p = fraction of inliers
- t = fit threshold
- $p_s$ = success probability

RANdom
SAmple
Consensus

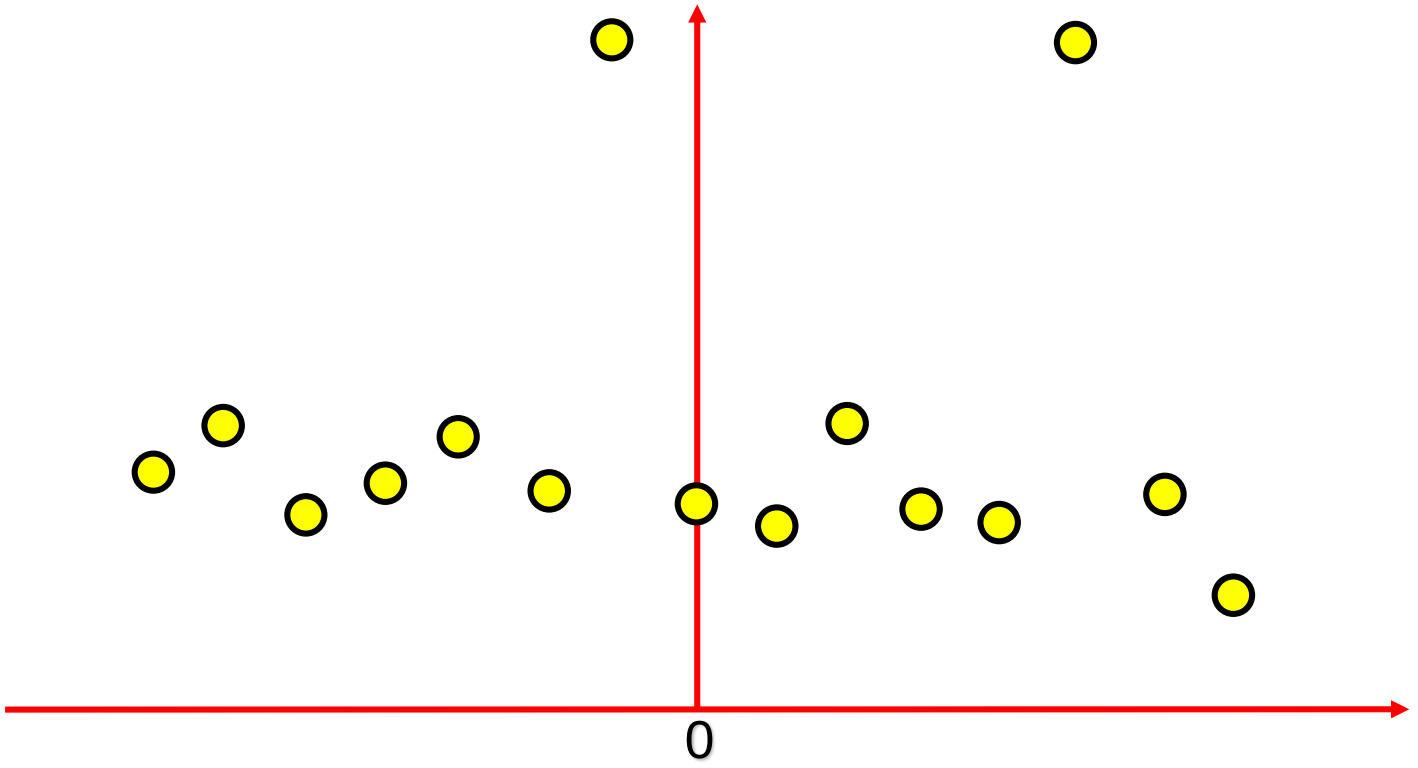
"outlier"



# RANSAC algorithm (Bolles & Fischler, 1981)

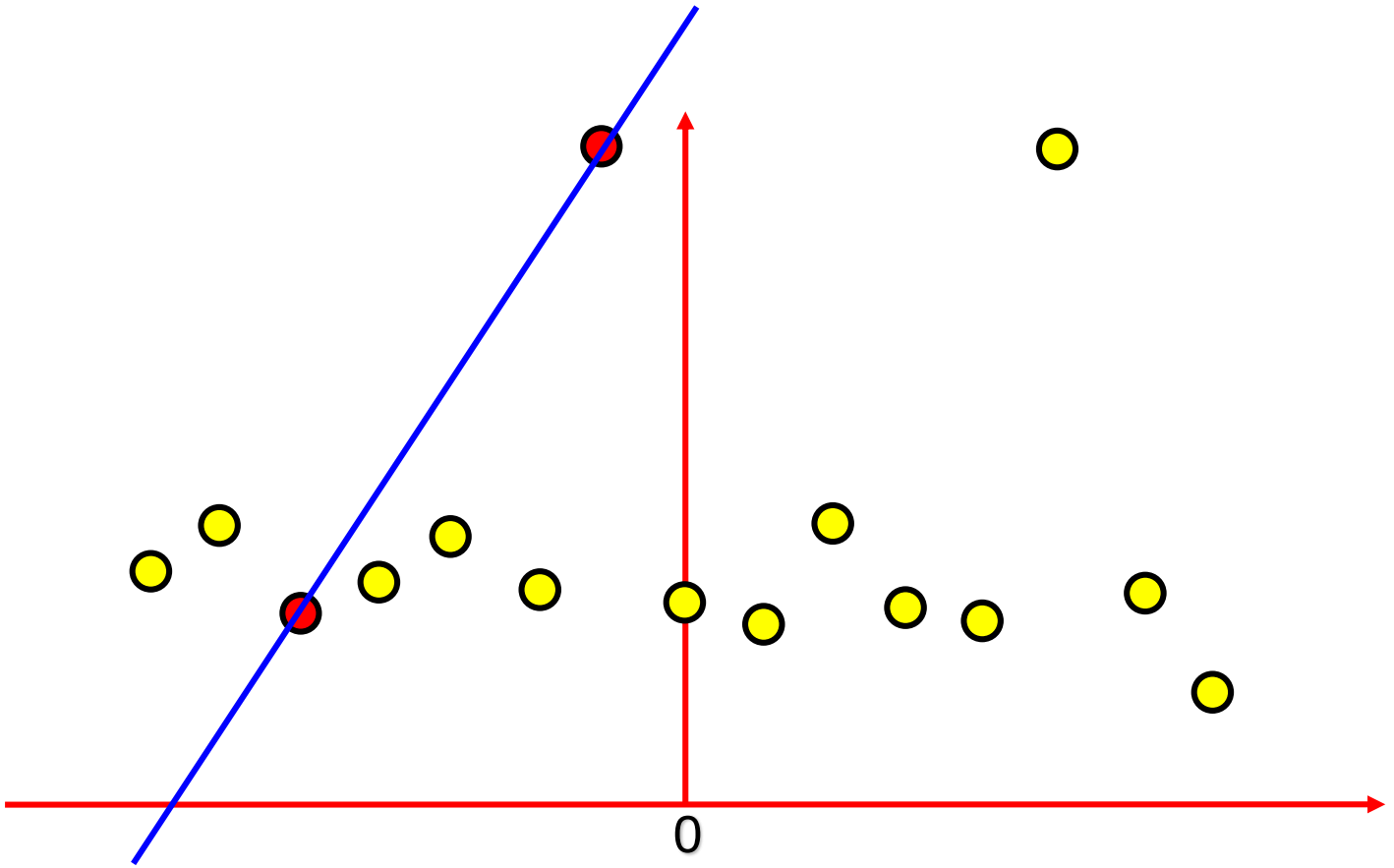
example: line fitting using RANSAC (i.e., n=2)

step 1: randomly choose n measurements



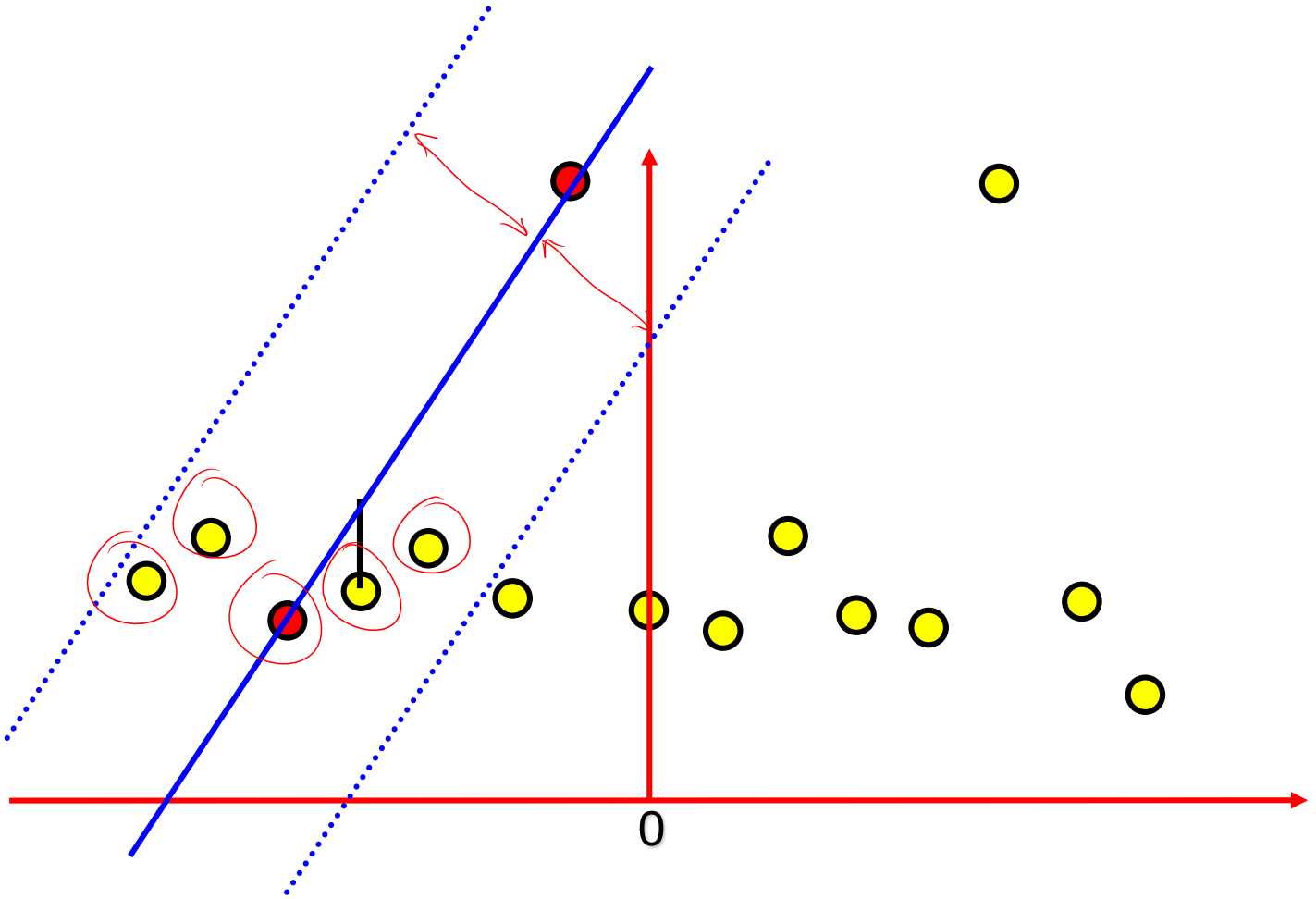
# RANSAC algorithm

step 2: compute the model parameters



# RANSAC algorithm: inlier detection

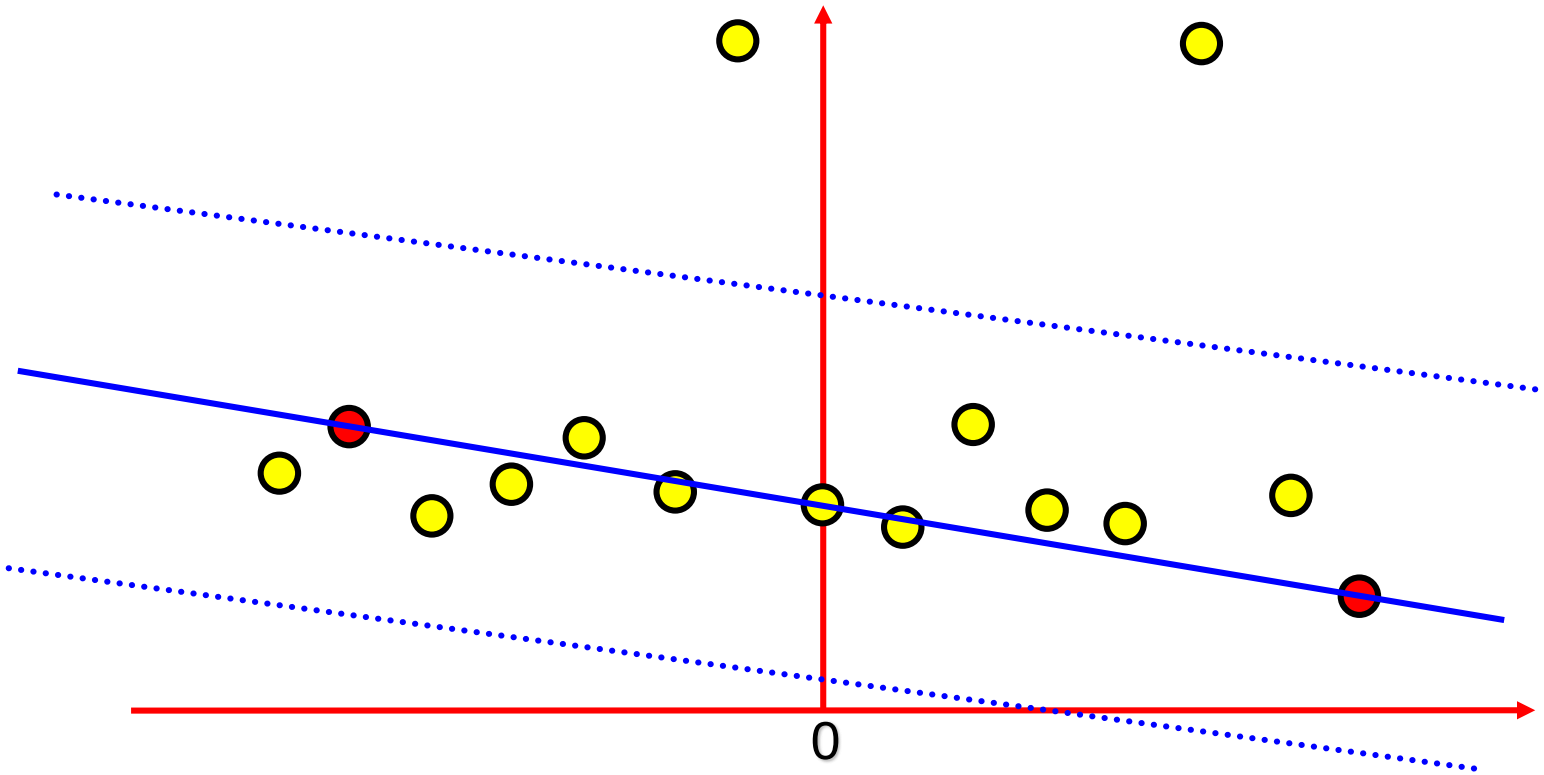
step 3: count measurements with $e_i < t$



# RANSAC algorithm

step 4: repeat K times

step 5: return model having the largest number of inliers

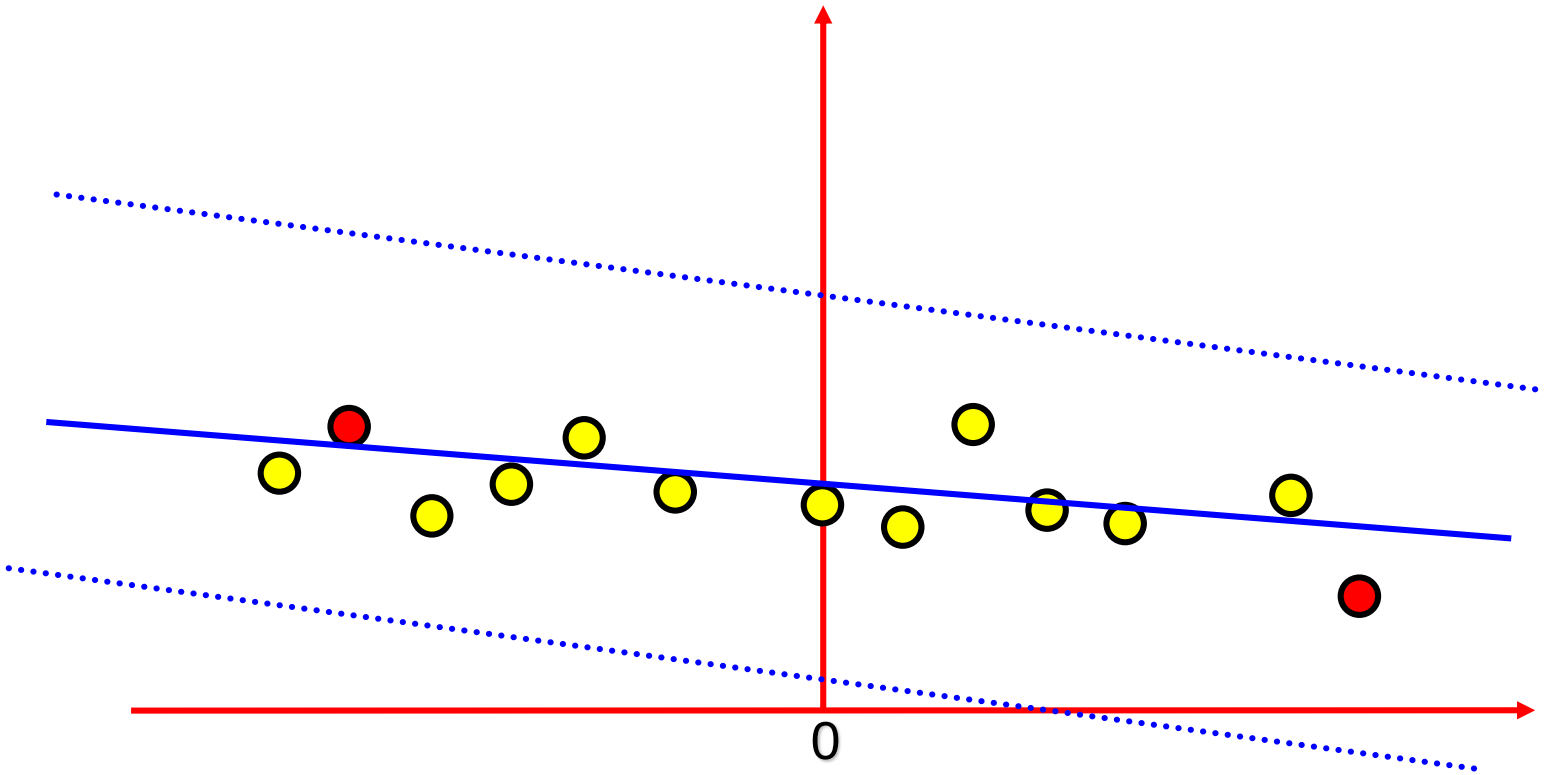


# RANSAC algorithm

step 4: repeat K times

step 5: return model having the largest number of inliers

(optional) compute optimal fit to the inliers using LS or TLS



# RANSAC algorithm: choosing K

given:

- n = # model parameters
- p = fraction of inliers
- t = fit threshold
- $p_s$ = success probability

1. randomly choose n measurements
2. fit n-parameter model
3. count the inliners
4. repeat K times
5. if at least one line with p inliers found, return SUCCESS

- Probability we chose an inlier: $p$
- Probability we chose (n+1) inlier: $p^{n+1}$
- Prob at least 1 outlier chosen: $1-p^{n+1}$
- Prob at least 1 outlier chosen in all K trials: $(1-p^{n+1})^K$
- Failure probability: $(1-p^{n+1})^K$
- Success probability $p_s = 1-(1-p^{n+1})^K$
- By taking logs on both sides

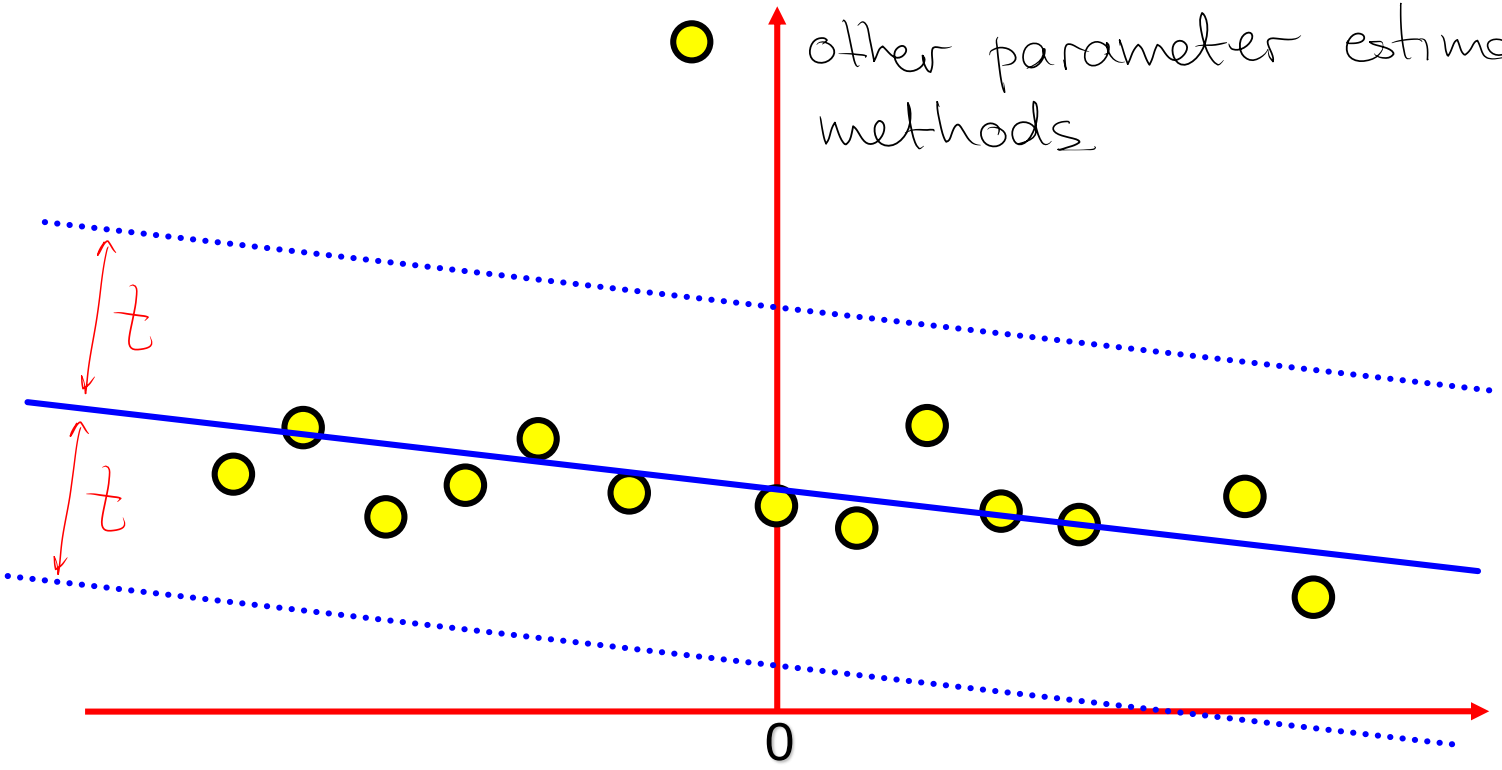
$$K = \frac{\log(1-p_s)}{\log(1-p^{n+1})}$$

# comments on RANSAC

given:

- n = # model parameters
- p = fraction of inliers
- t = fit threshold
- $p_s$ = success probability

- very popular & effective for geometric parameter estimation problems
- threshold hard to set
- can be combined with other parameter estimation methods



# robust estimation summary

- A redescending M-estimator, such as the Geman-McLure estimator, downweights the influence of data with large errors.
- The use of such an estimator leads to a non-linear optimization problem for the model parameters.
- The optimization problem typically has multiple local minima, some of which provide useful estimates while others are extraneous.
- The iteratively reweighted least squares algorithm provides one way to find local minima of the resulting objective function.
- Leverage points can significantly skew the parameter estimates, and can be controlled by limiting the spatial extent of the data being fit. This can also reduce the number of extraneous minima.
- Initial guesses can be generated by randomly sampling the data (see RANSAC). Another approach uses continuation with decreasing $\sigma$ (see deterministic annealing and graduated non-convexity).

# discussion

We have not provided any quantitative evaluation of the results which, almost certainly, would provide invaluable data. Why not?

What should we use for ground truth? Human data? Synthetic data?

How should the results from different line-finder algorithms be compared? What evaluation metric should be used? Different algorithms which use the line finder results can be expected to vary in their sensitivity to different types of errors.

In the end what matters is how cost-effective various algorithms are when included in a working robot. (For an example of this style of assessment, see Doughtery and Bowyer, 1998, Objective evaluation of edge detectors using a formally defined framework.)

Here we simply punt (i.e., accept our current gains and move on).

# Topic 01:

# Advanced methods (later)

- least squares & total least squares
- robust M-estimation
- optimizing robust objective functions
- proposal generation: Hough transforms
- proposal generation: RANSAC
- MLESAC & PEARL
- robust filtering