# Computer Vision

## 3. Estimating 2D Transformations

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

# Syllabus

| | | |
|---|---|---|
| Mon. 10.4. | (1) | 0. Introduction |
| | | 1. Projective Geometry in 2D: a. The Projective Plane |
| Mon. 17.4. | — | — *Easter Monday* — |
| Mon. 24.4. | (2) | 1. Projective Geometry in 2D: b. Projective Transformations |
| Mon. 1.5. | — | — *Labor Day* — |
| Mon. 8.5. | (3) | 2. Projective Geometry in 3D: a. Projective Space |
| Mon. 15.5. | (4) | 2. Projective Geometry in 3D: b. Quadrics, Transformations |
| Mon. 22.5. | (5) | 3. Estimating 2D Transformations: a. Direct Linear Transformation |
| Mon. 29.5. | (6) | 3. Estimating 2D Transformations: b. Iterative Minimization |
| Mon. 5.6. | — | — *Pentecoste Day* — |
| Mon. 12.6. | (7) | 4. Interest Points: a. Edges and Corners |
| Mon. 19.6. | (8) | 4. Interest Points: b. Image Patches |
| Mon. 26.6. | ( 9) | 5. Simulataneous Localization and Mapping: a. Camera Models |
| Mon. 3.7. | (10) | 5. Simulataneous Localization and Mapping: b. Triangulation |

# Outline

# Objects to estimate from data

- a 2D projectivity
- a 3D to 2D projection (camera)
- the Fundamental Matrix
- the Trifocal Tensor

Data:

- $N$ pairs $x_n$,$x'_n$ of corresponding points in two images $(n = 1, \ldots, N)$

Note: The Trifocal Tensor represents a relation between three images and thus requires $N$ **triples** of corresponding points $x_n$,$x'_n$, $x''_n$ in **three images** $(n = 1, \ldots, N)$.

# Outline

# From Corresponding Points to Linear Equations (1/2)

Inhomogeneous coordinates:

$$
\begin{aligned}
x'_n &\stackrel{!}{=} \hat{x}'_n := Hx_n, \quad n = 1, \ldots, N \\
&= \begin{pmatrix} x_n^T & 0^T & 0^T \\ 0^T & x_n^T & 0^T \\ 0^T & 0^T & x_n^T \end{pmatrix} h, \quad h := \mathrm{vect}(H) := \begin{pmatrix} H_{1,1} \\ H_{1,2} \\ H_{1,3} \\ H_{2,1} \\ \vdots \\ H_{3,3} \end{pmatrix}
\end{aligned}
$$

# From Corresponding Points to Linear Equations (1/2)

Inhomogeneous coordinates:

$$
\begin{aligned}
x'_n &\stackrel{!}{=} \hat{x}'_n := Hx_n, \quad n = 1, \ldots, N \\
&= \begin{pmatrix} x_n^T & 0^T & 0^T \\ 0^T & x_n^T & 0^T \\ 0^T & 0^T & x_n^T \end{pmatrix} h, \quad h := \mathrm{vect}(H) := \begin{pmatrix} H_{1,1} \\ H_{1,2} \\ H_{1,3} \\ H_{2,1} \\ \vdots \\ H_{3,3} \end{pmatrix}
\end{aligned}
$$

Homogeneous coordinates:

$$
\begin{aligned}
x'_{n,i} : x'_{n,j} &= \hat{x}'_{n,i} : \hat{x}'_{n,j}, \quad \forall i, j \in \{1, 2, 3\}, i \neq j \\
x'_{n,i}\hat{x}'_{n,j} - x'_{n,j}\hat{x}'_{n,i} &= 0, \quad \text{and one equation is linear dependent} \\
\rightsquigarrow 0 &\stackrel{!}{=} \underbrace{\begin{pmatrix} 0^T & -x'_{n,3}x_n^T & x'_{n,2}x_n^T \\ x'_{n,3}x_n^T & 0^T & -x'_{n,1}x_n^T \end{pmatrix}}_{=:A(x_n, x'_n)} h
\end{aligned}
$$

# From Corresponding Points to Linear Equations (2/2)

$$A(x_n, x'_n)h \stackrel{!}{=} 0, \quad n = 1, \ldots, N$$

$$\underbrace{\begin{pmatrix} A(x_1, x'_1) \\ A(x_2, x'_2) \\ \vdots \\ A(x_N, x'_N) \end{pmatrix}}_{=:A(x_{1:N}, x'_{1:N})} h = 0$$

- to estimate a general projectivity we need 4 points (8 equations, 8 dof)
- we are looking for non-trivial solutions $h \neq 0$.

# More than 4 Points & Noise: Overdetermined

- For $N > 4$ points and **exact coordinates**,
  the system $Ah = 0$ still has rank 8 and a non-trivial solution $h \neq 0$.
- But for $N > 4$ points and **noisy coordinates**,
  the system $Ah = 0$ is overdetermined and (in general) has only the trivial solution $h = 0$.

Relax the objective $Ah = 0$ to (**constrained least squares**)

$$\arg\min_{h:||h||=1} ||Ah|| = \arg\min_{h} \frac{||Ah||}{||h||}$$

$$= \text{(normed) eigenvector to smallest eigenvalue}$$

and solve via SVD:

$$A^T A = USU^T, \quad S = \text{diag}(s_1, \ldots, s_9), s_i \geq s_{i+1} \forall i, UU^T = I$$

$$h := U_{9,1:9}$$

# Degenerate Configurations: Underdetermined

- If three of the four points are collinear (in both images), $A$ will have rank $< 8$ and thus $h$ underdetermined, and thus there is no unique solution for $h$.

**Degenerate Configuration:**
Corresponding points that do not uniquely determine a transformation (in a particular class of transformations).

# Direct Linear Transformation Algorithm (DLT)

1: **procedure**
EST-2D-PROJECTIVITY-DLT$(x_1, x'_1, x_2, x'_2, \ldots, x_N, x'_N \in \mathbb{P}^2)$

2: $$A := \begin{pmatrix} A(x_1, x'_1) \\ A(x_2, x'_2) \\ \vdots \\ A(x_N, x'_N) \end{pmatrix} = \begin{pmatrix} 0^T & -x'_{1,3}x_1^T & x'_{1,2}x_1^T \\ x'_{1,3}x_1^T & 0^T & -x'_{1,1}x_1^T \\ 0^T & -x'_{2,3}x_2^T & x'_{2,2}x_2^T \\ x'_{2,3}x_2^T & 0^T & -x'_{2,1}x_2^T \\ \vdots & & \\ 0^T & -x'_{N,3}x_N^T & x'_{N,2}x_N^T \\ x'_{N,3}x_N^T & 0^T & -x'_{N,1}x_N^T \end{pmatrix}$$

3: $(U, S) := \mathrm{SVD}(A^T A)$

4: $h := U_{9,1:9}$

5: **return** $H := \begin{pmatrix} h_{1:3}^T \\ h_{4:6}^T \\ h_{7:9}^T \end{pmatrix}$

Note: Do not use this unnormalized version of DLT, but the one in section 3.

# Outline

# Algebraic Distance

- the loss minimized by DLT, represented as distance between
  - $x'$: point in 2nd image
  - $\hat{x}' := Hx$: estimated position of $x'$ by $H$

$$
\begin{aligned}
\ell_{\text{alg}}(H; x, x') &:= ||A(x', x)h||^2 \\
&= || \begin{pmatrix} 0^T & -x_3' x^T & x_2' x^T \\ x_3' x^T & 0^T & -x_1' x^T \end{pmatrix} h||^2 \\
&= || \begin{pmatrix} -x_3' \hat{x}_2' + x_2' \hat{x}_3' \\ x_3' \hat{x}_1' - x_1' \hat{x}_3' \end{pmatrix} ||^2 \\
&= d_{\text{alg}}(x', \hat{x}')^2
\end{aligned}
$$

with

$$
d_{\text{alg}}(x, y) := \sqrt{a_1^2 + a_2^2}, \quad (a_1, a_2, a_3)^T = x \times y
$$

# Geometric Distances: Transfer Errors

**Transfer Error in One Image** (2nd image):

$$\ell_{\text{trans1}}(H; x, x') := d(x', Hx)^2 = d(x', \hat{x}')^2$$

with Euclidean distance in inhomogeneous coordinates

$$d(x, y) := \sqrt{(x_1/x_3 - y_1/y_3)^2 + (x_2/x_3 - y_2/y_3)^2}$$

# Geometric Distances: Transfer Errors

**Transfer Error in One Image** (2nd image):

$$\ell_{\text{trans1}}(H; x, x') := d(x', Hx)^2 = d(x', \hat{x}')^2$$

with Euclidean distance in inhomogeneous coordinates

$$d(x, y) := \sqrt{(x_1/x_3 - y_1/y_3)^2 + (x_2/x_3 - y_2/y_3)^2}$$
$$= \frac{1}{|x_3||y_3|} d_{\text{alg}}(x, y)$$

- DLT/algebraic error equals geometric error for affine transformations ($x_3 = y_3 = 1$)

# Geometric Distances: Transfer Errors

**Transfer Error in One Image** (2nd image):

$$\ell_{\text{trans1}}(H; x, x') := d(x', Hx)^2 = d(x', \hat{x}')^2$$

with Euclidean distance in inhomogeneous coordinates

$$\begin{aligned} d(x, y) :=& \sqrt{(x_1/x_3 - y_1/y_3)^2 + (x_2/x_3 - y_2/y_3)^2} \\ =& \frac{1}{|x_3||y_3|} d_{\text{alg}}(x, y) \end{aligned}$$

- DLT/algebraic error equals geometric error for affine transformations ($x_3 = y_3 = 1$)

**Symmetric Transfer Error**:

$$\begin{aligned} \ell_{\text{strans}}(H; x, x') :=& d(x, H^{-1}x')^2 + d(x', Hx)^2 \\ =& d(x, \hat{x})^2 + d(x', \hat{x}')^2, \quad \hat{x} := H^{-1}x' \end{aligned}$$

# Transfer Errors: Probabilistic Interpretation

Assume

- measurements $x_n$ in the 1st image are noise-free,
- measurements $x'_n$ in the 2nd image are distributed Gaussian around true values $Hx_n$:

$$p(x'_n \mid Hx_n, \sigma^2) = \frac{1}{2\pi\sigma^2} e^{-d(x'_n, Hx_n)^2/(2\sigma^2)}$$

**log-likelihood** for Transfer Error in One Image:

$$\begin{aligned} p(H \mid x_{1:N}, x'_{1:N}) &= \frac{p(x_{1:N}, x'_{1:N} \mid H)p(H)}{p(x_{1:N}, x'_{1:N})} && \text{Bayes} \\ &\propto p(x_{1:N}, x'_{1:N} \mid H)p(H) \propto && p(x'_{1:N} \mid H, x_{1:N})p(H) \\ &= p(H) \prod_{n=1}^{N} p(x'_n \mid H, x_n) \propto && \prod_{n=1}^{N} p(x'_n \mid H, x_n) \\ \log p(H \mid x_{1:N}, x'_{1:N}) &\propto -\sum_{n=1}^{N} d(x'_n, Hx_n)^2 && = \text{transfer error} \end{aligned}$$

# Reprojection Error

- additionally to projectivity $H$, also find noise-free / perfectly matching pairs $\hat{x}, \hat{x}'$:

$$\text{minimize } \ell_{rep}(H, \hat{x}_1, \hat{x}'_1, \ldots, \hat{x}_N, \hat{x}'_N) := \sum_{n=1}^{N} d(x_n, \hat{x}_n)^2 + d(x'_n, \hat{x}'_n)^2$$

w.r.t.

$$\hat{x}'_n = H\hat{x}_n, \quad n = 1, \ldots, N$$

over

$$H, \hat{x}_1, \hat{x}'_1, \ldots, \hat{x}_N, \hat{x}'_N$$

**Reprojection Error**:

$$\ell_{\text{rep}}(H, \hat{x}, \hat{x}'; x, x') := d(x, \hat{x})^2 + d(x', \hat{x}')^2, \quad \text{with } \hat{x}' = H\hat{x}$$

- analogue probabilistic interpretation:
  - measurements $x, x'$ are Gaussian around true values $\hat{x}, \hat{x}'$

# Outline

# Are Solutions Invariant under Transformations?

- Given corresponding points $x_n, x'_n$,
  a method such as DLT will find a projectivity $H$.

# Are Solutions Invariant under Transformations?

- Given corresponding points $x_n, x'_n$,
  a method such as DLT will find a projectivity $H$.
- Now assume
  - the first image is transformed by projectivity $T$,
  - the second image is transformed by projectivity $T'$

  before we apply the estimation method.
  - Corresponding points now will be $\tilde{x}_n := Tx_n, \tilde{x}'_n := T'x'_n$
- Let $\tilde{H}$ be the projectivity estimated by the method applied to $\tilde{x}_n, \tilde{x}'_n$.
- Is it guaranteed that $H$ and $\tilde{H}$ are "the same" (equivalent) ?

$$\tilde{H} \stackrel{?}{=} T'HT^{-1}$$

# Are Solutions Invariant under Transformations?

- Given corresponding points $x_n, x'_n$,
  a method such as DLT will find a projectivity $H$.
- Now assume
  - the first image is transformed by projectivity $T$,
  - the second image is transformed by projectivity $T'$

  before we apply the estimation method.
  - Corresponding points now will be $\tilde{x}_n := Tx_n, \tilde{x}'_n := T'x'_n$
- Let $\tilde{H}$ be the projectivity estimated by the method applied to $\tilde{x}_n, \tilde{x}'_n$.
- Is it guaranteed that $H$ and $\tilde{H}$ are "the same" (equivalent) ?

$$\tilde{H} \stackrel{?}{=} T'HT^{-1}$$

- This may depend on the class of projectivities allowed for $T, T'$.
  - at least invariance under similarities would be useful !

# DLT is not Invariant under Similarities

- If $T'$ is a similarity transformation with scale factor $s$ and $T$ any projectivity, then one can show

$$||\tilde{A}\tilde{h}|| = s||Ah||$$

Note: $\tilde{A} := A(\tilde{x}_\cdot, \tilde{x}'_\cdot), \tilde{h} := \text{vect}(\tilde{H})$

# DLT is not Invariant under Similarities

- If $T'$ is a similarity transformation with scale factor $s$ and $T$ any projectivity, then one can show

$$||\tilde{A}\tilde{h}|| = s||Ah||$$

- But solutions $H$ and $\tilde{H}$ will not be equivalent nevertheless, as DLT minimizes under constraint $||h|| = 1$ and this constraint is not scaled with $s$ !
- So DLT is not invariant under similarity transforms.

Note: $\tilde{A} := A(\tilde{x}_{\cdot}, \tilde{x}'_{\cdot}), \tilde{h} := \text{vect}(\tilde{H})$

# Transfer/Reprojection Errors are Invariant under Similarities

- If $T'$ is Euclidean:

$$\begin{aligned} d(\tilde{x}'_n, \tilde{H}\tilde{x}_n)^2 =& d(T'x'_n, T'HT^{-1}Tx_n)^2 \\ =& x'^T_n T'^T T'HT^{-1}Tx_n = x'_n H x_n = d(x'_n, Hx_n)^2 \end{aligned}$$

- If $T'$ is a similarity with scale factor $s$:

$$\begin{aligned} d(\tilde{x}'_n, \tilde{H}\tilde{x}_n)^2 =& d(T'x'_n, T'HT^{-1}Tx_n)^2 \\ =& x'_n T'^T T'HT^{-1}Tx_n = x'_n s^2 H x_n = s^2 d(x'_n, Hx_n)^2 \end{aligned}$$

- Error is just scaled, so attains minimum at same position.
  $\rightsquigarrow$ Transfer/Reprojection Errors are invariant under similarities.

# DLT with Normalization

- Image coordinates of corresponding points are usually finite:
  $x = (x_1, x_2, 1)^T$,
  thus have different scale $(100, 100, 1)$ when measured in pixels.
- Therefore, entries in $A(x, x')$ will have largely different scale:

$$A(x, x') = \begin{pmatrix} 0^T & -x'_3 x^T & x'_2 x^T \\ x'_3 x^T & 0^T & -x'_1 x^T \end{pmatrix} = \begin{pmatrix} 0^T & -x^T & x'_2 x^T \\ x^T & 0^T & -x'_1 x^T \end{pmatrix}$$

  - some in 100s $(x^T)$, some in 10.000s $(x'_2 x^T, -x'_1 x^T)$

# DLT with Normalization

- normalize $x_{1:N}$:

$$\tilde{x}_{1:N} := \text{normalize}(x_{1:N}) := \left(\frac{x_n - \mu(x_{1:N})}{\tau(x_{1:N})/\sqrt{2}}\right)_{n=1,\ldots,N},$$

with

$$\mu(x_{1:N}) := \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \text{centroid/mean}$$

$$\tau(x_{1:N}) := \frac{1}{N}\sum_{n=1}^{N} d(x_n, \mu(x_{1:N})) \qquad \text{avg. distance to centroid}$$

- afterwards:

$$\mu(\tilde{x}_{1:N}) = 0, \quad \tau(\tilde{x}_{1:N}) = \sqrt{2}$$

- Normalization is a similarity transform:

$$T := T_{\text{norm}}(x_{1:N}) := \begin{pmatrix} \sqrt{2}/\tau(x_{1:N})I & -\mu(x_{1:N})\sqrt{2}/\tau(x_{1:N}) \\ 0 & 1 \end{pmatrix}$$

# DLT with Normalization / Algorithm

**procedure**
EST-2D-PROJECTIVITY-DLTN$(x_1, x'_1, x_2, x'_2, \ldots, x_N, x'_N \in \mathbb{P}^2)$

$$T := T_{\text{norm}}(x_{1:N}) := \begin{pmatrix} \sqrt{2}/\tau(x_{1:N})I & -\mu(x_{1:N})\sqrt{2}/\tau(x_{1:N}) \\ 0 & 1 \end{pmatrix}$$

$$T' := T_{\text{norm}}(x'_{1:N}) := \begin{pmatrix} \sqrt{2}/\tau(x'_{1:N})I & -\mu(x'_{1:N})\sqrt{2}/\tau(x'_{1:N}) \\ 0 & 1 \end{pmatrix}$$

$\tilde{x}_n := Tx_n \quad \forall n = 1, \ldots, N$ ▷ normalize $x_n$

$\tilde{x}'_n := T'x'_n \quad \forall n = 1, \ldots, N$ ▷ normalize $x'_n$

$\tilde{H} :=$ est-2d-projectivity-dlt$(\tilde{x}_1, \tilde{x}'_1, \tilde{x}_2, \tilde{x}'_2, \ldots, \tilde{x}_N, \tilde{x}'_N)$

$H := T'^{-1}\tilde{H}T$ ▷ unnormalize $\tilde{H}$

**return** $H$

# Outline

# Types of Problems

- The transformation estimation problem for the
  - **algebraic distance/loss** can be cast into a single
  - **linear system of equations (DLTn)**.
- The transformation estimation problem for the
  - **transfer distance/loss** as well as for the
  - **reconstruction loss** is more complicated and has to be handled by an explicit
  - **iterative minimization procedure**.

# Minimization **Objectives** $f : \mathbb{R}^M \to \mathbb{R}$

a) transfer distance in one image:

$$\text{minimize } f(H) := \sum_{n=1}^{N} d(x'_n, Hx_n)^2$$

b) symmetric transfer distance:

$$\text{minimize } f(H) := \sum_{n=1}^{N} d(x'_n, Hx_n)^2 + d(x_n, H^{-1}x'_n)^2$$

c) reconstruction loss:

$$\text{minimize } f(H, \hat{x}_{1:N}) := \sum_{n=1}^{N} d(x_n, \hat{x}_n)^2 + d(x'_n, H\hat{x}_n)^2$$

- $x_n, x'_n$ are constants, $H, \hat{x}_{1:N}$ variables
- a), b) have $M := 9$ **parameters** / variables
  - as $H$ as only 8 dof, the objective is slightly **overparametrized**
- c) has $M := 2N + 9$ **parameters** / variables
  - allowing only finite points for $\hat{x}_n$

# Objectives of type $f = e^T e$ (1/3)

All three objectives $f$ are $L_2$ norms of (parametrized) vectors, i.e. can be written as

$$f(x) = e(x)^T e(x), \quad h : \mathbb{R}^M \to \mathbb{R}^N$$

a) transfer distance in one image:

$$\begin{aligned} \text{minimize } f(H) &:= \sum_{n=1}^{N} d(x'_n, Hx_n)^2 \\ &= e(H)^T e(H), \end{aligned}$$

$$e(H) := \begin{pmatrix} x'_{1,1}/x'_{1,3} - (Hx_1)_1/(Hx_1)_3 \\ x'_{1,2}/x'_{1,3} - (Hx_1)_2/(Hx_1)_3 \\ \vdots \\ x'_{N,1}/x'_{N,3} - (Hx_N)_1/(Hx_N)_3 \\ x'_{N,2}/x'_{N,3} - (Hx_N)_2/(Hx_N)_3 \end{pmatrix}$$

# Objectives of type $f = e^T e$ (2/3)

b) symmetric transfer distance:

$$\text{minimize } f(H) := \sum_{n=1}^{N} d(x'_n, Hx_n)^2 + d(x_n, H^{-1}x'_n)^2 = e(H)^T e(H),$$

$$e(H) := \begin{pmatrix} x'_{1,1}/x'_{1,3} - (Hx_1)_1/(Hx_1)_3 \\ x'_{1,2}/x'_{1,3} - (Hx_1)_2/(Hx_1)_3 \\ \vdots \\ x'_{N,1}/x'_{N,3} - (Hx_N)_1/(Hx_N)_3 \\ x'_{N,2}/x'_{N,3} - (Hx_N)_2/(Hx_N)_3 \\ x_{1,1}/x_{1,3} - (H^{-1}x'_1)_1/(H^{-1}x'_1)_3 \\ x_{1,2}/x_{1,3} - (H^{-1}x'_1)_2/(H^{-1}x'_1)_3 \\ \vdots \\ x_{N,1}/x_{N,3} - (H^{-1}x'_N)_1/(H^{-1}x'_N)_3 \\ x_{N,2}/x_{N,3} - (H^{-1}x'_N)_2/(H^{-1}x'_N)_3 \end{pmatrix}$$

# Objectives of type $f = e^T e$ (3/3)

c) reconstruction loss:

$$\text{minimize } f(H, \hat{x}_{1:N}) := \sum_{n=1}^{N} d(x_n, \hat{x}_n)^2 + d(x'_n, H\hat{x}_n)^2 = e(H)^T e(H),$$

$$e(H) := \begin{pmatrix} x'_{1,1}/x'_{1,3} - (H\hat{x}_1)_1/(H\hat{x}_1)_3 \\ x'_{1,2}/x'_{1,3} - (H\hat{x}_1)_2/(H\hat{x}_1)_3 \\ \vdots \\ x'_{N,1}/x'_{N,3} - (H\hat{x}_N)_1/(H\hat{x}_N)_3 \\ x'_{N,2}/x'_{N,3} - (H\hat{x}_N)_2/(H\hat{x}_N)_3 \\ x_{1,1}/x_{1,3} - \hat{x}_{1,1} \\ x_{1,2}/x_{1,3} - \hat{x}_{1,2} \\ \vdots \\ x_{N,1}/x_{N,3} - \hat{x}_{N,1} \\ x_{N,2}/x_{N,3} - \hat{x}_{N,2} \end{pmatrix}$$

# Minimizing $f$ (I): Gradient Descent

To minimize $f : \mathbb{R}^M \to \mathbb{R}$ over $x \in \mathbb{R}^M$ **Gradient Descent**

1. starts at a random **starting point** $x_0 \in \mathbb{R}^M$

$$t := 0, \quad x^{(t)} := x_0$$

# Minimizing $f$ (I): Gradient Descent

To minimize $f : \mathbb{R}^M \to \mathbb{R}$ over $x \in \mathbb{R}^M$ **Gradient Descent**

1. starts at a random **starting point** $x_0 \in \mathbb{R}^M$

$$t := 0, \quad x^{(t)} := x_0$$

2. computes as **descent direction** $d^{(t)}$ **at** $x^{(t)}$
   — direction where $f$ decreases —
   the **gradient of** $f$:

$$d^{(t)} := -g^{(t)} := -\nabla_x f|_{x^{(t)}} := -(\frac{\partial f}{\partial x_m}(x^{(t)}))_{m=1,\ldots,M}$$

# Minimizing $f$ (I): Gradient Descent

To minimize $f : \mathbb{R}^M \to \mathbb{R}$ over $x \in \mathbb{R}^M$ **Gradient Descent**

1. starts at a random **starting point** $x_0 \in \mathbb{R}^M$

$$t := 0, \quad x^{(t)} := x_0$$

2. computes as **descent direction** $d^{(t)}$ **at** $x^{(t)}$
   — direction where $f$ decreases —
   the **gradient of** $f$:

$$d^{(t)} := -g^{(t)} := -\nabla_x f|_{x^{(t)}} := -(\frac{\partial f}{\partial x_m}(x^{(t)}))_{m=1,\ldots,M}$$

3. moves into the descent direction:

$$x^{(t+1)} := x^{(t)} + d$$

# Minimizing $f$ (I): Gradient Descent

To minimize $f : \mathbb{R}^M \to \mathbb{R}$ over $x \in \mathbb{R}^M$ **Gradient Descent**

1. starts at a random **starting point** $x_0 \in \mathbb{R}^M$

$$t := 0, \quad x^{(t)} := x_0$$

2. computes as **descent direction** $d^{(t)}$ **at** $x^{(t)}$
   — direction where $f$ decreases —
   the **gradient of** $f$:

$$d^{(t)} := -g^{(t)} := -\nabla_x f|_{x^{(t)}} := -(\frac{\partial f}{\partial x_m}(x^{(t)}))_{m=1,\ldots,M}$$

3. moves into the descent direction:

$$x^{(t+1)} := x^{(t)} + d$$

Beware:

- $f$ decreases only in the neighborhood of $x^{(t)}$
- A full gradient step may be too large and **not** leading to a decrease !

# Minimizing $f$ (I): Gradient Descent w. Steplength Control

To minimize $f : \mathbb{R}^M \to \mathbb{R}$ over $x \in \mathbb{R}^M$ **Gradient Descent**

1. starts at a random **starting point** $x_0 \in \mathbb{R}^M$

$$t := 0, \quad x^{(t)} := x_0$$

2. computes as **descent direction** $d^{(t)}$ **at** $x^{(t)}$
   — direction where $f$ decreases —
   the **gradient of** $f$:

$$d^{(t)} := -g^{(t)} := -\nabla_x f|_{x^{(t)}} := -(\frac{\partial f}{\partial x_m}(x^{(t)}))_{m=1,\ldots,M}$$

3. finds a steplength $\alpha \in \mathbb{R}^+$ so that $f$ actually decreases:

$$\alpha := \max\{\alpha := 2^{-k} \mid k = 0, 1, 2, \ldots, f(x + \alpha d) < f(x)\}$$

4. moves a step into the descent direction:

$$x^{(t+1)} := x^{(t)} + \alpha d$$

# Minimizing $f$ (I): Gradient Descent / Algorithm

**procedure** MIN-GD($f : \mathbb{R}^M \to \mathbb{R}, x_0 \in \mathbb{R}^M, \nabla_x f : \mathbb{R}^M \to \mathbb{R}^M, \epsilon \in \mathbb{R}^+$)
    $x := x_0$
    **do**
        $d := -\nabla_x f|_x$
        $\alpha := 1$
        **while** $f(x + \alpha d) \geq f(x)$ **do**
            $\alpha := \alpha/2$
        $x := x + \alpha d$
    **while** $||d|| > \epsilon$
    **return** $x$

# Minimizing $f$ (II): Newton

The Newton algorithm computes a better descent direction:

- approximate $f$ by the **quadratic Taylor expansion at $x^{(t)}$**:

$$f(x+d) \approx \tilde{f}(d) := f(x^{(t)}) + \nabla_x f|_{x^{(t)}}^T d + \frac{1}{2} d^T \nabla_x^2 f|_{x^{(t)}}^T d$$
$$= f(x^{(t)}) + g_{x^{(t)}}^T d + \frac{1}{2} d^T H_{x^{(t)}} d$$

where

$$\nabla_x^2 f|_x := H_x := \left(\frac{\partial^2 f}{\partial x_m \partial x_k}\right)_{m,k=1,\ldots,M}$$ **Hessian of $f$**

- the approximation attains its minimum at

$$0 \stackrel{!}{=} \nabla_d \tilde{f}(d) = g_{x^{(t)}} + H_{x^{(t)}} d$$
$$H_{x^{(t)}} d = -g_{x^{(t)}}$$ **normal equations**

- solve this linear system of equations to find descent direction

# Minimizing $f$ (II): Newton / Algorithm

**procedure** MIN-NEWTON($f : \mathbb{R}^M \to \mathbb{R}, x_0 \in \mathbb{R}^M,$
$\nabla_x f : \mathbb{R}^M \to \mathbb{R}^M, \nabla_x^2 f : \mathbb{R}^M \to \mathbb{R}^{M \times M}, \epsilon \in \mathbb{R}^+$)
  $x := x_0$
  **do**
    $g := \nabla_x f|_x$
    $H := \nabla_x^2 f|_x$
    $d := \text{solve}_d(Hd = -g)$
    $\alpha := 1$
    **while** $f(x + \alpha d) \geq f(x)$ **do**
      $\alpha := \alpha/2$
    $x := x + \alpha d$
  **while** $||d|| > \epsilon$
  **return** $x$

# Minimizing $f = e^T e$ (I): Gauss-Newton

Gauss-Newton is

- a **specialization of the Newton algorithm**
- for **objectives of type** $f(x) = e(x)^T e(x)$
- that **approximates the Hessian**:

# Minimizing $f = e^T e$ (I): Gauss-Newton

Gauss-Newton is

- a **specialization of the Newton algorithm**
- for **objectives of type** $f(x) = e(x)^T e(x)$
- that **approximates the Hessian**:

$$\nabla_x f|_x = 2\nabla_x e|_x^T e(x)$$

# Minimizing $f = e^T e$ (I): Gauss-Newton

Gauss-Newton is

- a **specialization of the Newton algorithm**
- for **objectives of type** $f(x) = e(x)^T e(x)$
- that **approximates the Hessian**:

$$\nabla_x f|_x = 2\nabla_x e|_x^T e(x)$$
$$\nabla_x^2 f|_x = 2\nabla_x e|_x^T \nabla_x e|_x + 2\nabla_x^2 e|_x^T e(x)$$

# Minimizing $f = e^T e$ (I): Gauss-Newton

Gauss-Newton is

- a **specialization of the Newton algorithm**
- for **objectives of type** $f(x) = e(x)^T e(x)$
- that **approximates the Hessian**:

$$\nabla_x f|_x = 2\nabla_x e|_x^T e(x)$$
$$\nabla_x^2 f|_x = 2\nabla_x e|_x^T \nabla_x e|_x + 2\nabla_x^2 e|_x^T e(x)$$

Now approximate $e$ by a linear Taylor expansion, i.e.

$$\nabla_x^2 e|_x \approx 0$$
$$\rightsquigarrow \quad \nabla_x^2 f|_x \approx 2\nabla_x e|_x^T \nabla_x e|_x$$

# Minimizing $f = e^T e$ (I): Gauss-Newton

Gauss-Newton is

- a **specialization of the Newton algorithm**
- for **objectives of type** $f(x) = e(x)^T e(x)$
- that **approximates the Hessian**:

$$\nabla_x f|_x = 2\nabla_x e|_x^T e(x)$$
$$\nabla_x^2 f|_x = 2\nabla_x e|_x^T \nabla_x e|_x + 2\nabla_x^2 e|_x^T e(x)$$

Now approximate $e$ by a linear Taylor expansion, i.e.

$$\nabla_x^2 e|_x \approx 0$$
$$\rightsquigarrow \quad \nabla_x^2 f|_x \approx 2\nabla_x e|_x^T \nabla_x e|_x$$

- all we need is the gradient of $e$ !

# Minimizing $f = e^T e$ (I): Gauss-Newton / Algorithm

**procedure** MIN-GAUSS-NEWTON($e : \mathbb{R}^M \to \mathbb{R}^N, x_0 \in \mathbb{R}^M, \nabla_x e : \mathbb{R}^M \to \mathbb{R}^{N \times M}, \epsilon \in \mathbb{R}^+$)
    $x := x_0$
    **do**
        $J := \nabla_x e|_x$
        $g := J^T e(x)$
        $H := J^T J$
        $d := \text{solve}_d(Hd = -g)$
        $\alpha := 1$
        **while** $e(x + \alpha d)^T e(x + \alpha d) \geq e(x)^T e(x)$ **do**
            $\alpha := \alpha/2$
        $x := x + \alpha d$
    **while** $||d|| > \epsilon$
    **return** $x$

# Minimizing $f = e^T e$ (II): Levenberg-Marquardt

- slight variation of the Gauss-Newton method

$$J^T J\, d = -g \qquad \text{Gauss-Newton Normal Eq.}$$

$$(J^T J + \lambda I)\, d = -g \qquad \text{Levenberg-Marquardt Normal Eq.}$$

# Minimizing $f = e^T e$ (II): Levenberg-Marquardt

- slight variation of the Gauss-Newton method

$$J^T J\, d = -g \qquad \text{Gauss-Newton Normal Eq.}$$

$$(J^T J + \lambda I)\, d = -g \qquad \text{Levenberg-Marquardt Normal Eq.}$$

- if new objective value is worse, try again with larger $\lambda$
  - for large $\lambda$: equivalent to Gradient descent with small stepsize $1/\lambda$

$$(J^T J + \lambda I) \approx \lambda I, \qquad (J^T J + \lambda I)\, d = -g \qquad \rightsquigarrow d = -\frac{1}{\lambda} g$$

# Minimizing $f = e^T e$ (II): Levenberg-Marquardt

- slight variation of the Gauss-Newton method

$$J^T J\, d = -g \qquad \text{Gauss-Newton Normal Eq.}$$
$$(J^T J + \lambda I)\, d = -g \qquad \text{Levenberg-Marquardt Normal Eq.}$$

- if new objective value is worse, try again with larger $\lambda$
  - for large $\lambda$: equivalent to Gradient descent with small stepsize $1/\lambda$

$$(J^T J + \lambda I) \approx \lambda I, \qquad (J^T J + \lambda I)\, d = -g \qquad \rightsquigarrow d = -\frac{1}{\lambda} g$$

- once new objective value is smaller, accept and decrease $\lambda$
  - for small $\lambda$: equivalent to Gauss-Newton with (large) stepsize 1

# Minimizing $f = e^T e$ (I): Levenberg-Marquardt / Algorithm

**procedure** MIN-LEVENBERG-MARQUARDT($e : \mathbb{R}^M \to \mathbb{R}^N, x_0 \in \mathbb{R}^M, \nabla_x e : \mathbb{R}^M \to \mathbb{R}^{N \times M}, \epsilon \in \mathbb{R}^+$)
    $x := x_0$
    $\lambda := 1$
    **do**
        $J := \nabla_x e|_x$
        $g := J^T e(x)$
        $\lambda := (\lambda/10)/10$
        **do**
            $H := J^T J + \lambda I$
            $d := \text{solve}_d(Hd = -g)$
            $\lambda := 10\lambda$
        **while** $e(x+d)^T e(x+d) \geq e(x)^T e(x)$
        $x := x + d$
    **while** $||d|| > \epsilon$
    **return** $x$

# Example: Reconstruction Loss (1/2)

$$e(H) := \begin{pmatrix} x'_{1,1}/x'_{1,3} - (H\hat{x}_1)_1/(H\hat{x}_1)_3 \\ x'_{1,2}/x'_{1,3} - (H\hat{x}_1)_2/(H\hat{x}_1)_3 \\ \vdots \\ x'_{N,1}/x'_{N,3} - (H\hat{x}_N)_1/(H\hat{x}_N)_3 \\ x'_{N,2}/x'_{N,3} - (H\hat{x}_N)_2/(H\hat{x}_N)_3 \\ x_{1,1}/x_{1,3} - \hat{x}_{1,1} \\ x_{1,2}/x_{1,3} - \hat{x}_{1,2} \\ \vdots \\ x_{N,1}/x_{N,3} - \hat{x}_{N,1} \\ x_{N,2}/x_{N,3} - \hat{x}_{N,2} \end{pmatrix} = \mathrm{vect}\left(\begin{pmatrix} e^1_{1:N,1:2} \\ e^2_{1:N,1:2} \end{pmatrix}\right)$$

with

$$e^1_{n,i} := x'_{n,i}/x'_{n,3} - (H\hat{x}_n)_i/(H\hat{x}_n)_3$$

$$e^2_{n,i} := x_{n,i}/x_{n,3} - \hat{x}_{n,i}$$

# Example: Reconstruction Loss (2/2)

$$e^1_{n,i} := \frac{x'_{n,i}}{x'_{n,3}} - \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3}$$

$$e^2_{n,i} := x_{n,i}/x_{n,3} - \hat{x}_{n,i}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^1_{n,i}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^2_{n,i}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^1_{n,i}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^2_{n,i}$$

Note: $(H\hat{x}_n)_i = \sum_{j=1}^{3} H_{i,j}\hat{x}_{n,j}$.

# Example: Reconstruction Loss (2/2)

$$e^1_{n,i} := \frac{x'_{n,i}}{x'_{n,3}} - \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3}$$

$$e^2_{n,i} := x_{n,i}/x_{n,3} - \hat{x}_{n,i}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^1_{n,i} = \begin{cases} -\frac{H_{i,\tilde{i}}}{(H\hat{x}_n)_3} + \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3^2} H_{3,\tilde{i}}, & \text{if } \tilde{n} = n \\ 0, & \text{else} \end{cases}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^2_{n,i}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^1_{n,i}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^2_{n,i}$$

Note: $(H\hat{x}_n)_i = \sum_{j=1}^{3} H_{i,j}\hat{x}_{n,j}$.

# Example: Reconstruction Loss (2/2)

$$e^1_{n,i} := \frac{x'_{n,i}}{x'_{n,3}} - \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3}$$

$$e^2_{n,i} := x_{n,i}/x_{n,3} - \hat{x}_{n,i}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^1_{n,i} = \begin{cases} -\frac{H_{i,\tilde{i}}}{(H\hat{x}_n)_3} + \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3^2} H_{3,\tilde{i}}, & \text{if } \tilde{n} = n \\ 0, & \text{else} \end{cases}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^2_{n,i} = \begin{cases} -1, & \text{if } \tilde{n} = n, \tilde{i} = i \\ 0, & \text{else} \end{cases}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^1_{n,i}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^2_{n,i}$$

Note: $(H\hat{x}_n)_i = \sum_{j=1}^{3} H_{i,j}\hat{x}_{n,j}$.

# Example: Reconstruction Loss (2/2)

$$e^1_{n,i} := \frac{x'_{n,i}}{x'_{n,3}} - \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3}$$

$$e^2_{n,i} := x_{n,i}/x_{n,3} - \hat{x}_{n,i}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^1_{n,i} = \begin{cases} -\frac{H_{i,\tilde{i}}}{(H\hat{x}_n)_3} + \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3^2} H_{3,\tilde{i}}, & \text{if } \tilde{n} = n \\ 0, & \text{else} \end{cases}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^2_{n,i} = \begin{cases} -1, & \text{if } \tilde{n} = n, \tilde{i} = i \\ 0, & \text{else} \end{cases}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^1_{n,i} = -\delta(\tilde{i} = i) \frac{\hat{x}_{n,\tilde{j}}}{(H\hat{x}_n)_3} + \delta(\tilde{i} = 3) \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3^2} \hat{x}_{n,3}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^2_{n,i}$$

Note: $(H\hat{x}_n)_i = \sum_{j=1}^{3} H_{i,j} \hat{x}_{n,j}$.

# Example: Reconstruction Loss (2/2)

$$e^1_{n,i} := \frac{x'_{n,i}}{x'_{n,3}} - \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3}$$

$$e^2_{n,i} := x_{n,i}/x_{n,3} - \hat{x}_{n,i}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^1_{n,i} = \begin{cases} -\frac{H_{i,\tilde{i}}}{(H\hat{x}_n)_3} + \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3^2} H_{3,\tilde{i}}, & \text{if } \tilde{n} = n \\ 0, & \text{else} \end{cases}$$

$$\nabla_{\hat{x}_{\tilde{n},\tilde{i}}} e^2_{n,i} = \begin{cases} -1, & \text{if } \tilde{n} = n, \tilde{i} = i \\ 0, & \text{else} \end{cases}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^1_{n,i} = -\delta(\tilde{i} = i) \frac{\hat{x}_{n,\tilde{j}}}{(H\hat{x}_n)_3} + \delta(\tilde{i} = 3) \frac{(H\hat{x}_n)_i}{(H\hat{x}_n)_3^2} \hat{x}_{n,3}$$

$$\nabla_{H_{\tilde{i},\tilde{j}}} e^2_{n,i} = 0$$

Note: $(H\hat{x}_n)_i = \sum_{j=1}^{3} H_{i,j} \hat{x}_{n,j}$.

# Example: Comparison of Different Methods


a

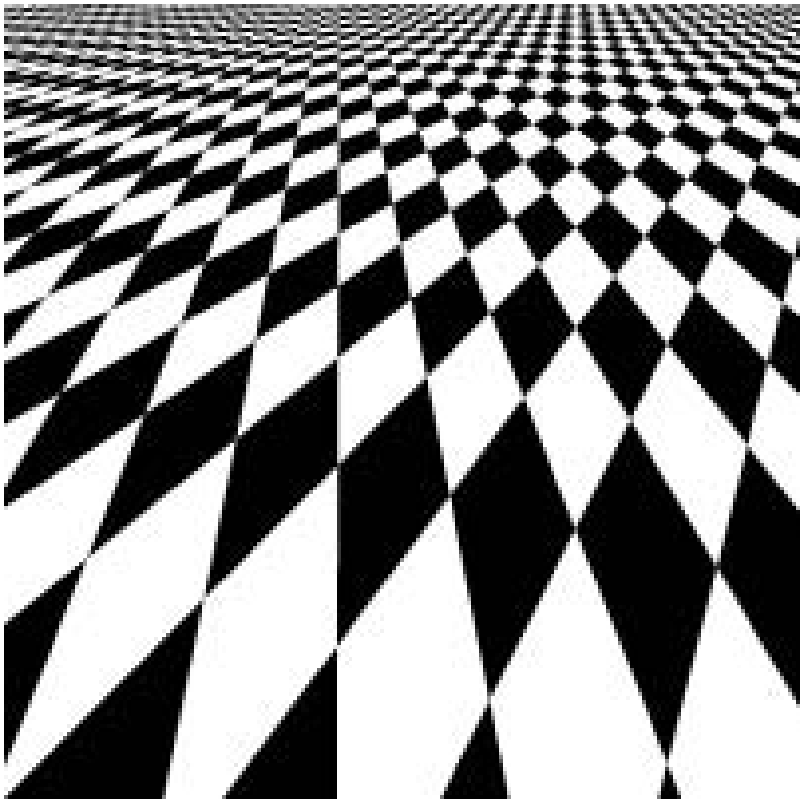

b


c

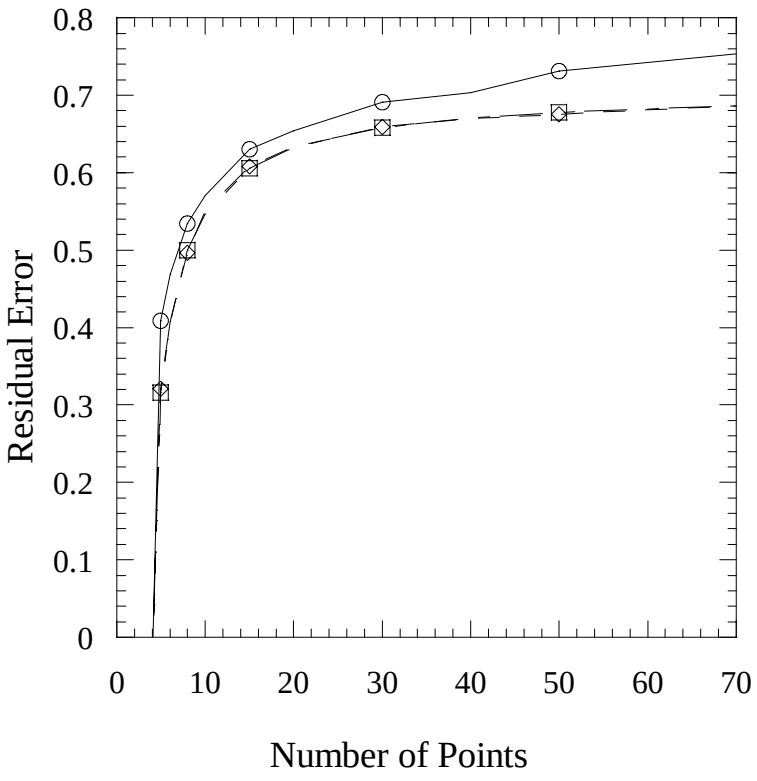
| | residual error in pixels | |
|---|---|---|
| method | pair a,b | pair a,c |
| DLT unnormalized | 0.4080 | 26.2056 |
| DLT normalized | 0.4078 | 0.6602 |
| Transfer distance in one image | 0.4077 | 0.6602 |
| Reconstruction loss | 0.4078 | 0.6602 |
| affine | 6.0095 | 2.8481 |

[HZ04, p. 115]

# Example: Comparison of Different Methods



a

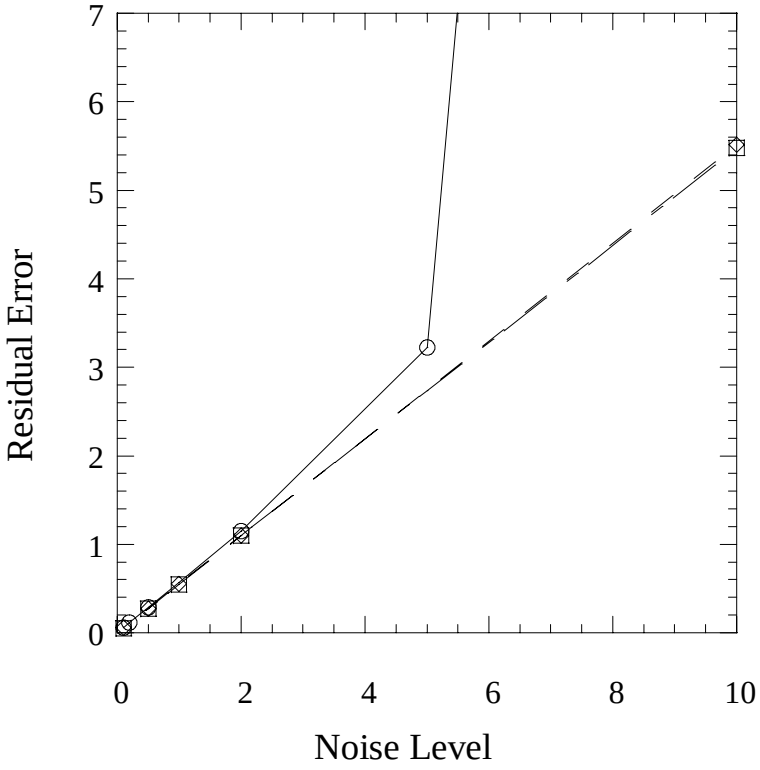



b

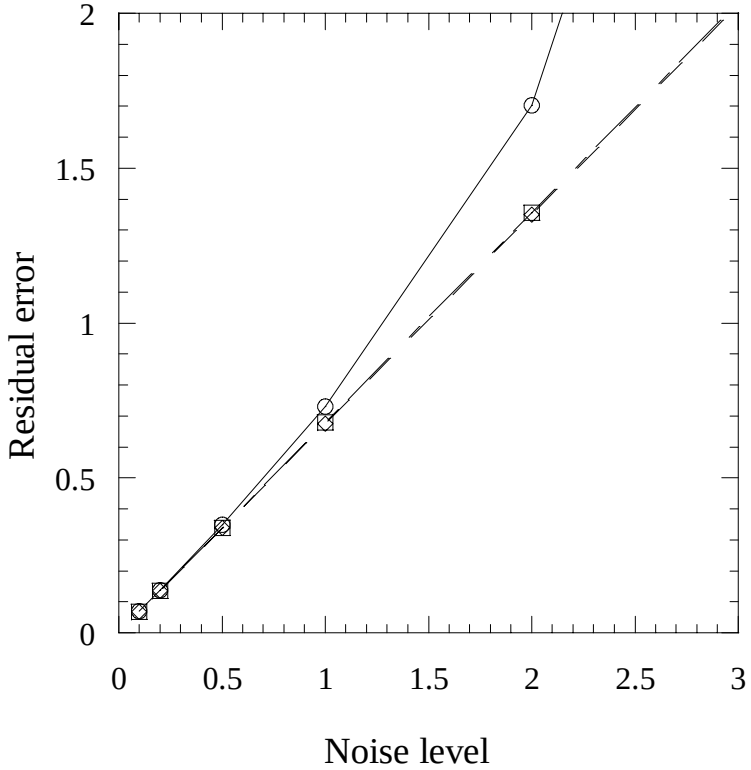
Note: solid: DLTn, dashed: reconstruction loss

[HZ04, p. 116]

# Example: Comparison of Different Methods




c



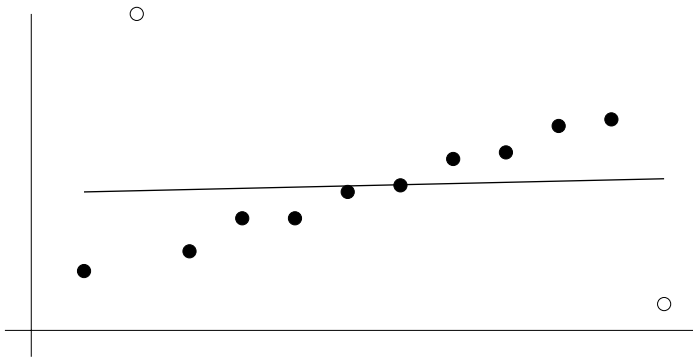

d

Note: solid: DLTn, dashed: reconstruction loss; c) 10 points, d) 50 points [HZ04, p. 116]

# Outline

# Outliers and Robust Estimation

- When estimating a transformation from pairs of corresponding points, having these correspondences estimated from data themselves, we expect **noise**: **wrong correspondences**.
- Wrong correspondences could be not just a little bit off, but **way off**: **outliers**.
- Some losses, esp. least squares, are **sensitive to outliers**:



a

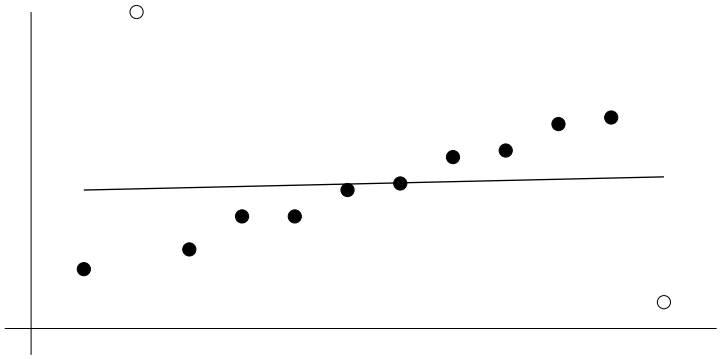
- **Robust estimation:** estimation that is less sensitive to outliers.

[HZ04, p. 117]

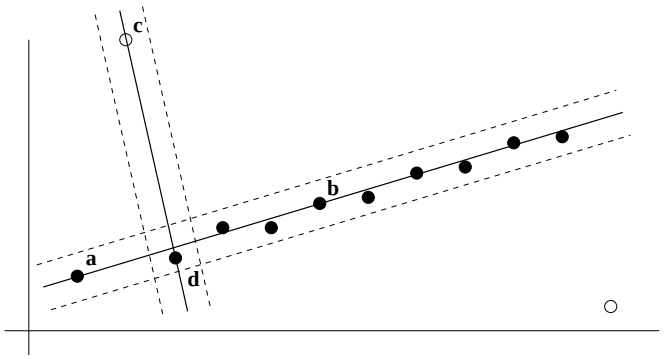
# Random Sample Consensus (RANSAC)

idea:

1. draw iteratively random samples of data points
   - many and small enough so that some will have no outliers with high probability
2. estimate the model from such a sample
3. grade the samples by the **support** of their models
   - support: number of well-explained points, i.e., points with a small error under the model (inliers)
4. reestimate the model on the support of the best sample



a




b

[HZ04, p. 117]

# Model Estimation Terminology

- RANSAC works like a wrapper around any estimation method.
- examples:
  - estimating a transformation from point correspondences
  - estimating a line (a linear model) from 2d points
- model estimation terminology:

$\mathcal{X}$ **data space**, e.g. $\mathbb{R}^2$

$\mathcal{D} \subseteq \mathcal{X}$ **dataset**, e.g. $\mathcal{D} = \{x_1, \ldots, x_N\}$

$f(\theta \mid \mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ell(x, \theta)$ **objective**

$\ell : \mathcal{X} \times \Theta \to \mathbb{R}$ **loss/error**, e.g. $\ell(\begin{pmatrix} x \\ y \end{pmatrix}; \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}) := (y - (\theta_1 + \theta_2 x))^2$

$\Theta$ **(model) parameter space**, e.g. $\mathbb{R}^2$

$a : \mathcal{P}(\mathcal{X}) \to \Theta$ **estimation method**, e.g. gradient descent

aiming at $a(\mathcal{D}) \approx \arg\min_{\theta \in \Theta} f(\theta \mid \mathcal{D})$

# RANSAC Algorithm

**procedure** EST-RANSAC($\mathcal{D}, \ell, a;\ N' \in \mathbb{N}, T \in \mathbb{N}, \ell_{\text{max}} \in \mathbb{R}, \text{sup}_{\text{min}} \in \mathbb{N}$)

  $\mathcal{S}_{\text{best}} := \emptyset$

  **for** $t = 1, \ldots, T$ or until $|\mathcal{S}| \geq \text{sup}_{\text{min}}$ **do**

    $\mathcal{D}' \sim \mathcal{D}$ of size $N'$ ▷ draw a sample

    $\hat{\theta} := a(\mathcal{D}')$ ▷ estimate the model

    $\mathcal{S} := \{x \in \mathcal{D} \mid \ell(x, \hat{\theta}) < \ell_{\text{max}}\}$ ▷ compute support

    **if** $|\mathcal{S}| > |\mathcal{S}_{\text{best}}|$ **then**

      $\mathcal{S}_{\text{best}} := \mathcal{S}$

  $\hat{\theta} := a(\mathcal{S}_{\text{best}})$ ▷ reestimate the model

  **return** $\hat{\theta}$

# What is a good **sample size** $N'$?

- often the minimum number to get a unique solution is used.

# What is a good **maximal support loss** $\ell_{\text{max}}$?

- for squared distance/L2 loss: $\ell(x, x') := (x - x')^2$
- assume Gaussian noise: $x_{\text{obs}} \sim \mathcal{N}(x_{\text{true}}, \Sigma)$,
  - isotrop noise
  - but no noise in some directions
    - e.g., points on a line: noise only orthogonal to the line

$\rightsquigarrow$ $\Sigma = USU^T$, $S = \text{diag}(s_1, s_2), s_i \in \{\sigma^2, 0\}, UU^T = I$

$\rightsquigarrow$ $\ell(x_{\text{obs}}, x_{\text{true}}) \sim \sigma^2 \chi^2_m$, $\quad m := \text{rank}(S)$ degrees of freedom

- inlier: $\ell(x_{\text{obs}}, x_{\text{true}}) < \ell_{\text{max}}$ with probability $\alpha$
  $\ell_{\text{max}} := \sigma^2 \text{CDF}^{-1}_{\chi^2_m}(\alpha)$

| $m$ | model | $\ell_{\text{max}}(\alpha = 0.95)$ |
|---|---|---|
| 1 | line, fundamtental matrix | $3.84\sigma^2$ |
| 2 | projectivity, camera matrix | $5.99\sigma^2$ |
| 3 | trifocal tensor | $7.81\sigma^2$ |

# What is a good **sample frequency** $T$?

- find $T$ s.t. at least one of the samples contains no outliers with high probability $\alpha := 0.99$.
- denote $p(x \text{ is an outlier}) = \epsilon$:

$$p(\mathcal{D}' \text{ contains no outliers}) = (1-\epsilon)^{N'}$$

$$p(\text{at least one } \mathcal{D}' \text{ contains no outliers}) = 1 - (1-(1-\epsilon)^{N'})^T \stackrel{!}{=} \alpha$$

$$\rightsquigarrow \quad T = \frac{1-\alpha}{1-(1-\epsilon)^{N'}}$$

# What is a good **sample frequency** $T$?

- find $T$ s.t. at least one of the samples contains no outliers with high probability $\alpha := 0.99$.
- denote $p(x \text{ is an outlier}) = \epsilon$:

$$p(\mathcal{D}' \text{ contains no outliers}) = (1-\epsilon)^{N'}$$

$$p(\text{at least one } \mathcal{D}' \text{ contains no outliers}) = 1 - (1 - (1-\epsilon)^{N'})^T \stackrel{!}{=} \alpha$$

$$\rightsquigarrow \quad T = \frac{1-\alpha}{1-(1-\epsilon)^{N'}}$$

| | $\epsilon = p(x \text{ is an outlier})$ | | | | | |
|---|---|---|---|---|---|---|
| $N'$ | 5% | 10% | 20% | 30% | 40% | 50% |
| 2 | 2 | 3 | 5 | 7 | 11 | 17 |
| 3 | 3 | 4 | 7 | 11 | 19 | 35 |
| 4 | 3 | 5 | 9 | 17 | 34 | 72 |
| 5 | 4 | 6 | 12 | 26 | 57 | 146 |
| 6 | 4 | 7 | 16 | 37 | 97 | 293 |
| 7 | 4 | 8 | 20 | 54 | 163 | 588 |
| 8 | 5 | 9 | 26 | 78 | 272 | 1177 |

# What is a good **sufficient support size $\text{sup}_{\text{min}}$**?

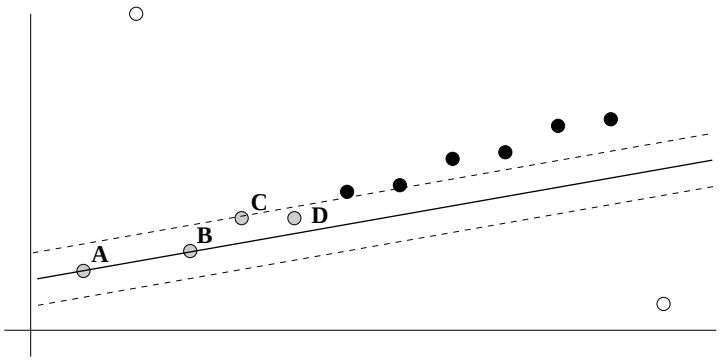
- the sufficient support size is an **early stopping criterion**.
- stop if we have as many inliers as expected:

$$\text{sup}_{\text{min}} = N(1 - \epsilon)$$

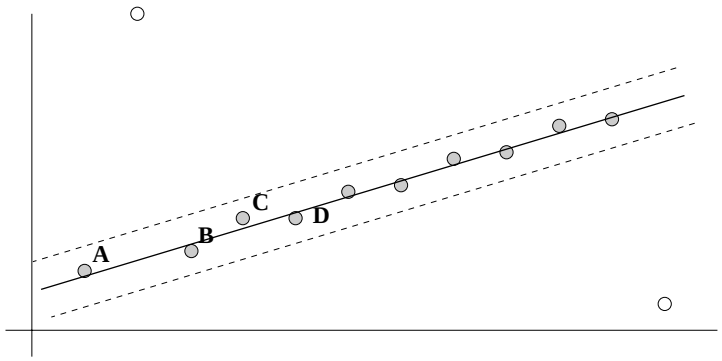
# RANSAC Algorithm / Repeated Reestimation

**procedure**
EST-RANSAC-RERE$(\mathcal{D}, \ell, a;\ N' \in \mathbb{N}, T \in \mathbb{N}, \ell_{\text{max}} \in \mathbb{R}, \text{sup}_{\text{min}} \in \mathbb{N})$
$\vdots$
    $\mathcal{S} := \mathcal{S}_{\text{best}}$
    **do**
        $\mathcal{S}_{\text{final}} := \mathcal{S}$
        $\hat{\theta} := a(\mathcal{S}_{\text{final}})$ ▷ reestimate the model
        $\mathcal{S} := \{x \in \mathcal{D} \mid \ell(x, \hat{\theta}) < \ell_{\text{max}}\}$ ▷ compute support
    **while** $\mathcal{S}_{\text{final}} \neq \mathcal{S}$
    **return** $\hat{\theta}$

# RANSAC: Repeated Reestimation




a




b

a) estimation from initial sample

b) reestimation from sample plus support

[HZ04, p. 121]

# Outline

# Putting it All Together

1. **interest points**:
   compute interest points in each image.

# Putting it All Together

1. **interest points**:
   compute interest points in each image.
2. **putative matches**:
   compute matching pairs of interest points from their proximity and intensity neighborhood.

# Putting it All Together

1. **interest points**:
   compute interest points in each image.
2. **putative matches**:
   compute matching pairs of interest points from their proximity and intensity neighborhood.
3. **simultaneously estimate a projectivity (model) and identify outliers** (robust estimation):

# Putting it All Together

1. **interest points**:
   compute interest points in each image.
2. **putative matches**:
   compute matching pairs of interest points from their proximity and intensity neighborhood.
3. **simultaneously estimate a projectivity (model) and identify outliers** (robust estimation):
   3.1 estimate a projectivity $H$ from several samples of 4 points and keep the one with maximal support/inliers (**RANSAC** using **DLTn**)

# Putting it All Together

1. **interest points**:
   compute interest points in each image.
2. **putative matches**:
   compute matching pairs of interest points from their proximity and intensity neighborhood.
3. **simultaneously estimate a projectivity (model) and identify outliers** (robust estimation):
   3.1 estimate a projectivity $H$ from several samples of 4 points and keep the one with maximal support/inliers (**RANSAC** using **DLTn**)
   3.2 reestimate the projectivity $H$ using the best sample and all its support/inliers (using **Levenberg-Marquardt**; RANSAC final step)

# Putting it All Together

1. **interest points**:
   compute interest points in each image.
2. **putative matches**:
   compute matching pairs of interest points from their proximity and intensity neighborhood.
3. **simultaneously estimate a projectivity (model) and identify outliers** (robust estimation):
   3.1 estimate a projectivity $H$ from several samples of 4 points and keep the one with maximal support/inliers (**RANSAC** using **DLTn**)
   3.2 reestimate the projectivity $H$ using the best sample and all its support/inliers (using **Levenberg-Marquardt**; RANSAC final step)
   3.3 **Guided Matching**: use projectivity $H$ to identify a search region about the transferred points (with relaxed threshold)

# Example

Left and right image:

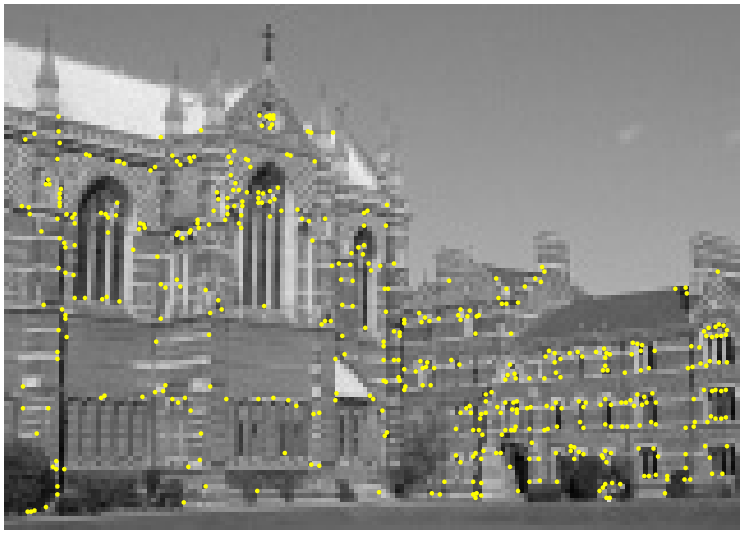
a

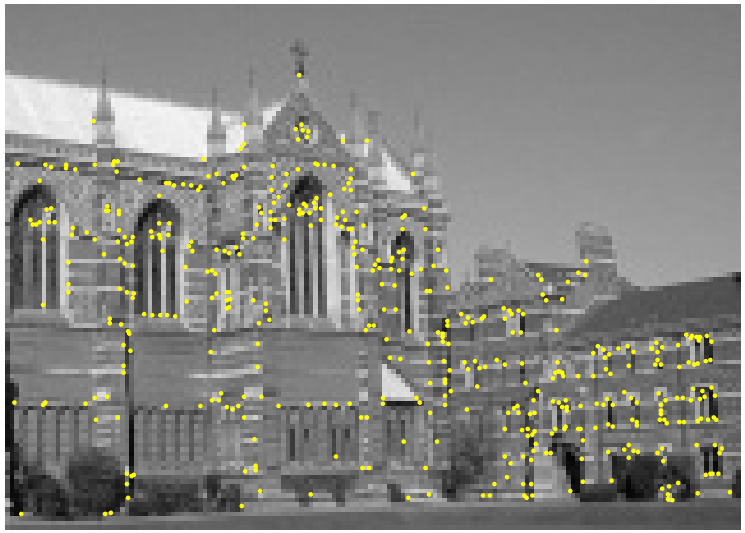
b

[HZ04, p. 126]

# Example

ca. 500+500 interest points ("corners"):





[HZ04, p. 126]

# Example

268 putative matches:

e

[HZ04, p. 126]

# Example

117 outlier — 151 inlier matches:

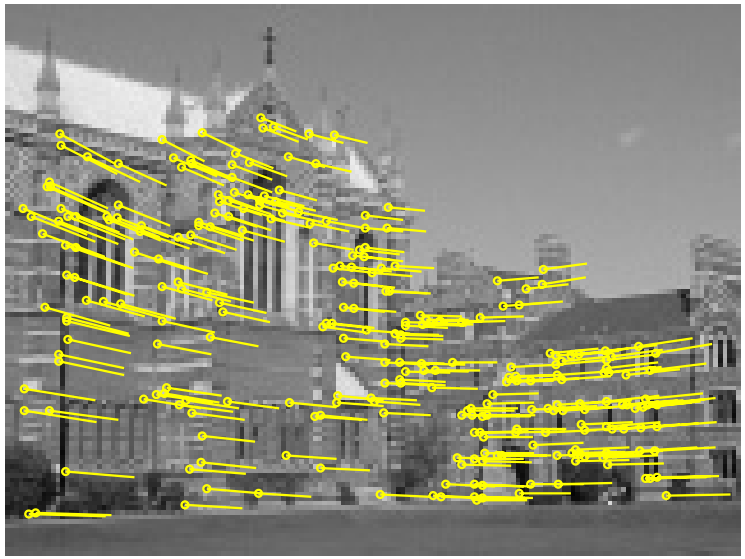
f

g

[HZ04, p. 126]

# Example

262 final matches (after guided matching):



h

[HZ04, p. 126]

# Summary

- ...

# Further Readings

- [HZ04, ch. 4].
- For iterative estimation methods in CV see [HZ04, appendix 6].
- You may also read [HZ04, ch. 5] which will not be covered in the lecture explicitly.

# References

Richard Hartley and Andrew Zisserman.
*Multiple view geometry in computer vision*.
Cambridge university press, 2004.