Photogrammetry & Robotics Lab

Bundle Adjustment – Part II
Numerics of BA

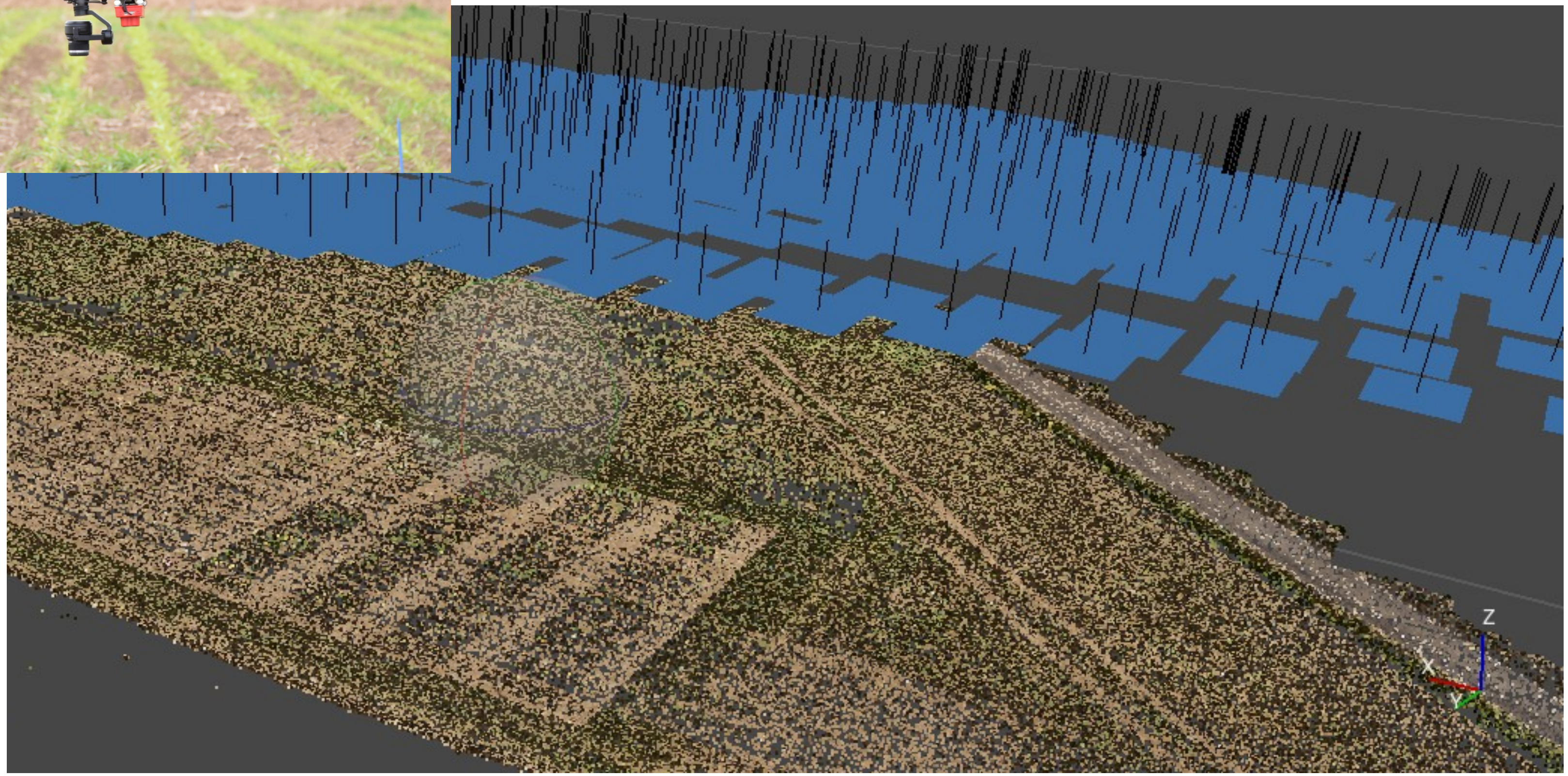
Cyrill Stachniss

5 Minute Preparation for Today

https://www.ipb.uni-bonn.de/5min/

Bundle Adjustment - Part I
Short Reminder

Bundle Adjustment for Aerial Triangulation



Bundle Adjustment for Aerial Triangulation

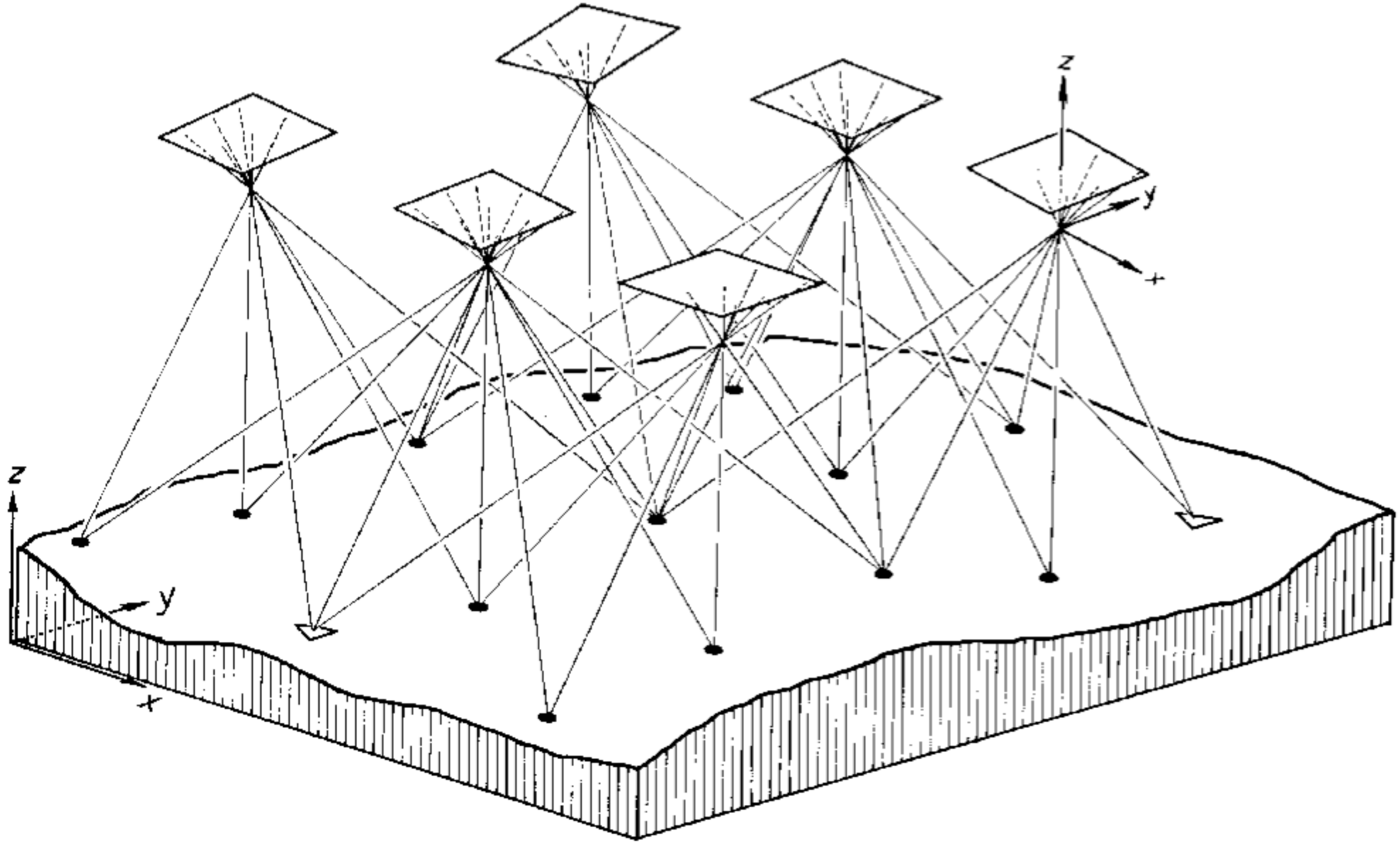


Image courtesy: Ackermann

Bundle Adjustment

Least squares approach to estimating camera poses and 3D points

Key idea:

- Start with an initial guess
- Project the estimated 3D points into the estimated camera images
- Compare locations of the projected 3D points with measured (2D) ones
- Adjust to minimize error in the images

Reprojection Error

BA is a non-linear least squares approach

project. params

img pix.

distortion

a="arbitrary frame"

$$ {}^{a}\mathbf{x}'_{ij} + {}^{a}\widehat{\mathbf{v}}_{x'_{ij}} = \widehat{\lambda}_{ij}\ {}^{a}\widehat{\mathsf{P}}_{j}(\boldsymbol{x}_{ij}, \boldsymbol{p}, \boldsymbol{q})\ \widehat{\mathbf{X}}_{i} $$

ij = point i observed in image j

corrections

scale factor

projection matrix (w/ non-lin. calib.)

3D point

$$ \text{with } \Sigma_{x_{ij}x_{ij}} \quad i = 1, ..., I_j; j = 1, ..., J $$

uncertainty in the image coordinates

#points in image j

#images

Unknown Parameters

- Non-linear least squares approach

$$ {}^{a}\mathbf{x}'_{ij} + {}^{a}\widehat{\mathbf{v}}_{x'_{ij}} = \widehat{\lambda}_{ij}\ {}^{a}\widehat{\mathsf{P}}_{j}(\boldsymbol{x}_{ij}, \boldsymbol{p}, \boldsymbol{q})\ \widehat{\mathbf{X}}_{i} $$

Unknowns:

- 3D locations of new points $\widehat{\mathbf{X}}_{i}$
- 1D scale factor $\widehat{\lambda}_{ij}$
- 6D exterior orientation
- 5D projection parameters (interior o.)
- Non-linear distortion parameters $\boldsymbol{q}$

Eliminating the Scale Factors

We can eliminate the per-point scale factor by using Euclidian coordinates (instead of homogenous coordinates)

$$ {}^{a}\mathbf{x}'_{ij} + {}^{a}\widehat{\mathbf{v}}_{x'_{ij}} = \widehat{\lambda}_{ij}\ {}^{a}\widehat{\mathsf{P}}_{j}(\boldsymbol{x}_{ij}, \boldsymbol{p}, \boldsymbol{q})\ \widehat{\mathbf{X}}_{i} $$

$$ {}^{a}\boldsymbol{x}'_{ij} + {}^{a}\widehat{\boldsymbol{v}}_{x'_{ij}} = \frac{{}^{a}\widehat{\mathsf{P}_{1:2}}_{j}(\boldsymbol{x}_{ij}, \boldsymbol{p}, \boldsymbol{q})\ \widehat{\mathbf{X}}_{i}}{{}^{a}\widehat{\mathsf{P}_{3}}_{j}(\boldsymbol{x}_{ij}, \boldsymbol{p}, \boldsymbol{q})\ \widehat{\mathbf{X}}_{i}} $$

Example: ~13M unknowns reduce to ~3M unknowns

Setting Up and Solving the System of Normal Equations

- Standard procedure…
- With unknowns $\boldsymbol{x}$ and observations $\boldsymbol{l}$
- Setup the normal equations

$$A^{\mathsf{T}}\Sigma^{-1}A\,\underbrace{\Delta\boldsymbol{x}}_{\text{unknowns}} = A^{\mathsf{T}}\Sigma^{-1}\underbrace{\Delta\boldsymbol{l}}_{\text{observations}}$$

- This yields the estimate

$$\underbrace{\widehat{\Delta\boldsymbol{x}}}_{\text{unknowns}} = (A^{\mathsf{T}}\Sigma^{-1}A)^{-1}A^{\mathsf{T}}\Sigma^{-1}\underbrace{\Delta\boldsymbol{l}}_{\text{observations}}$$

Part II
Numeric of the
Bundle Adjustment

We Cannot Solve the Linear System of BA in a Straightforward Manner

Why?

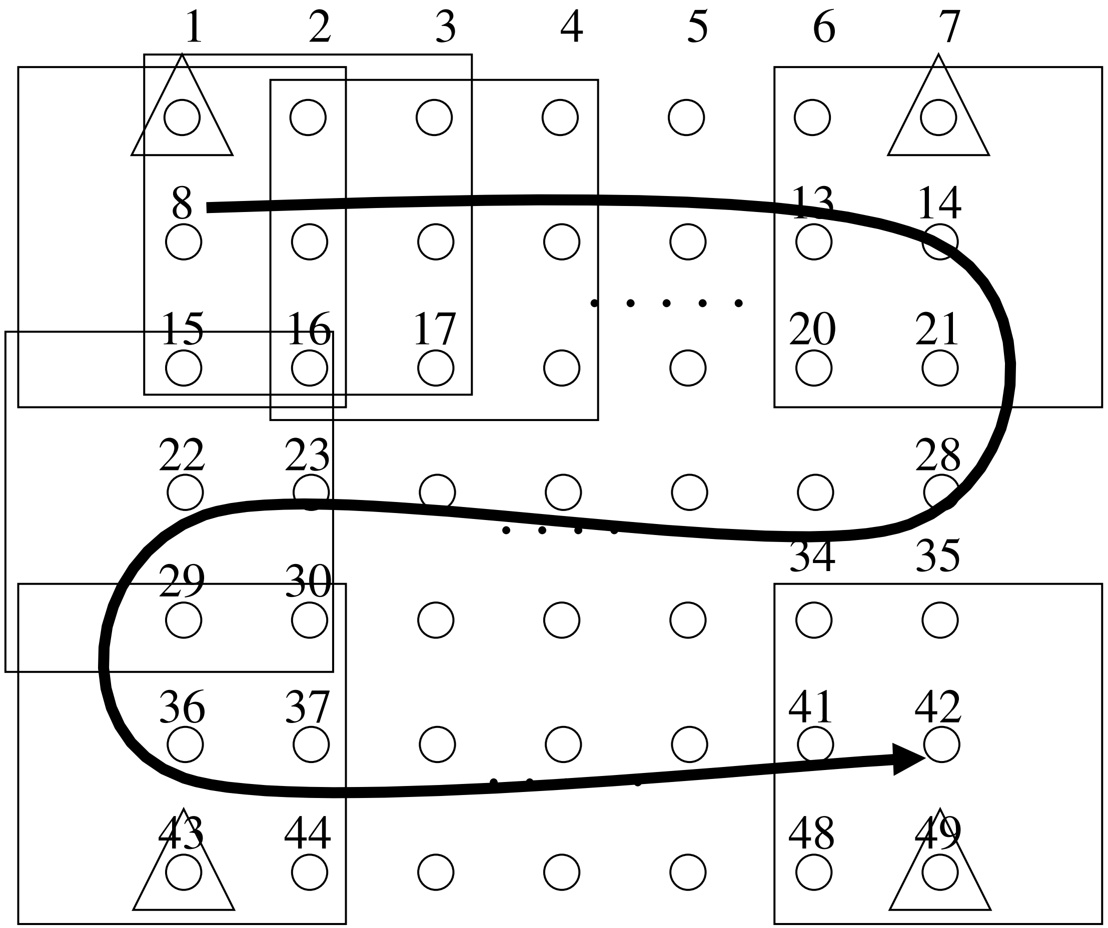
We Cannot Solve the LS in a Straightforward Manner

The linear system becomes **too large**

Example

- 20k images, 18 points per image
- Every point is observed on avg. 3 times
- 120k points = 360k location parameters
- 120k orientation parameters (6x20k)
- 480k parameters from 720k observations
- **Jacobian matrix: ~3.5×10^{11} elements**
- **Normal equation: ~2.3×10^{11} elements**

Let's Study a Small Example to Understand the Structure



Setup

- 3 stripes
- 7 images per stripe
- 60%/20% overlap
- 21 images
- 49 points
- 4 full CPs
- 45 new points

Image courtesy: Förstner

Configuration

- 6 images with 6 points
- 15 images with 9 points
- 171 images points yield 171 x2=**342 observat.**
- 45 new points yield 45 x3=**135 unknowns**
- 21 x6=**126 unknowns** orientation parameters
- In sum **261 unknowns**

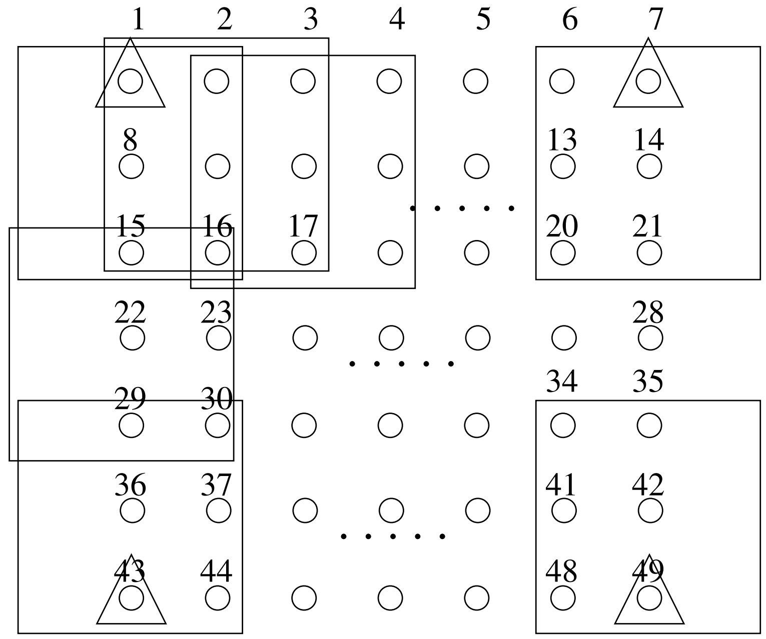


Image courtesy: Förstner

Coefficient Matrix

- We have $\Delta \boldsymbol{l} + \boldsymbol{v} \quad = \quad A \Delta \boldsymbol{x}$
- Let us split up $\Delta \boldsymbol{x}$

$$\Delta \boldsymbol{x} = \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix}$$

3D point coordinates

6D orientation parameters (cams)

- This leads to

$$\Delta \boldsymbol{l} + \boldsymbol{v} = \underbrace{[C \; B]}_{A} \underbrace{\begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix}}_{\Delta \boldsymbol{x}} = C \Delta \boldsymbol{k} + B \Delta \boldsymbol{t}$$

Coefficient Matrix

- We have $\Delta \boldsymbol{l} + \boldsymbol{v} = [C \ B] \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} = C \Delta \boldsymbol{k} + B \Delta \boldsymbol{t}$
- Thus, for every error equation

$$\Delta \boldsymbol{l}_{ij} + \boldsymbol{v}_{ij} = \underbrace{A_{ij}^{\mathsf{T}}}_{2 \times U} \Delta \boldsymbol{x} = \underbrace{C_{ij}^{\mathsf{T}}}_{2 \times 3} \Delta \boldsymbol{k}_i + \underbrace{B_{ij}^{\mathsf{T}}}_{2 \times 6} \Delta \boldsymbol{t}_j$$

- Coefficient matrix

$$A = \begin{bmatrix} A_{2,1}^{\mathsf{T}} \\ \cdots \\ A_{ij}^{\mathsf{T}} \\ \cdots \\ A_{48,21}^{\mathsf{T}} \end{bmatrix}$$

Coefficient Matrix

- Structure

$$A = \begin{bmatrix} A_{2,1}^{\mathsf{T}} \\ \dots \\ A_{ij}^{\mathsf{T}} \\ \dots \\ A_{48,21}^{\mathsf{T}} \end{bmatrix}$$

- with

$$\underset{2\times U}{A_{ij}^{\mathsf{T}}} = [0, \dots, 0, \underset{2\times 3}{C_{ij}^{\mathsf{T}}}, 0, \dots, 0 \mid 0, \dots, 0, \underset{2\times 6}{B_{ij}^{\mathsf{T}}}, 0, \dots, 0]$$

What follows from this structure?

Coefficient Matrix

- Structure

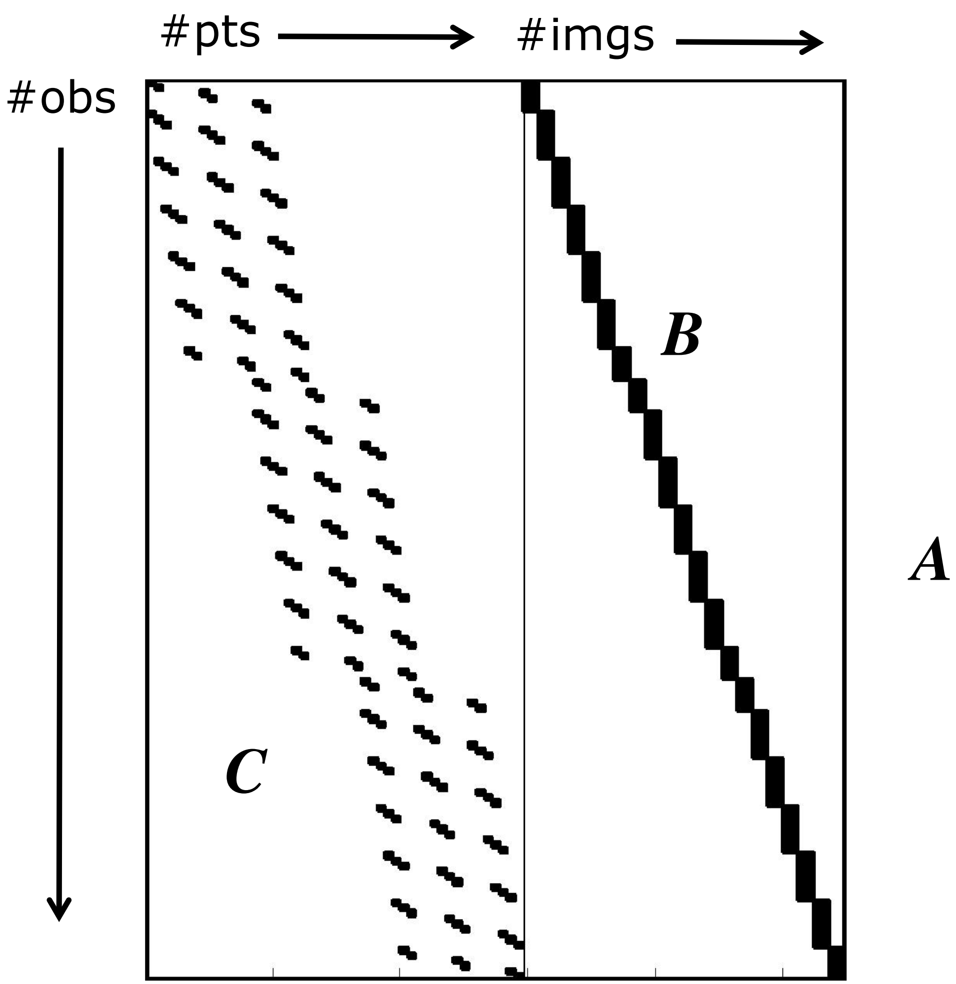
$$A = \begin{bmatrix} A_{2,1}^\top \\ \cdots \\ A_{ij}^\top \\ \cdots \\ A_{48,21}^\top \end{bmatrix}$$

- with

$$\underset{2\times U}{A_{ij}^\top} = [0, \ldots, 0, \underset{2\times 3}{C_{ij}^\top}, 0, \ldots, 0 \mid 0, \ldots, 0, \underset{2\times 6}{B_{ij}^\top}, 0, \ldots, 0]$$

- **Sparse matrix**: mostly 0 entries

Coefficient Matrix



For 3 stripes with
7 images per stripe

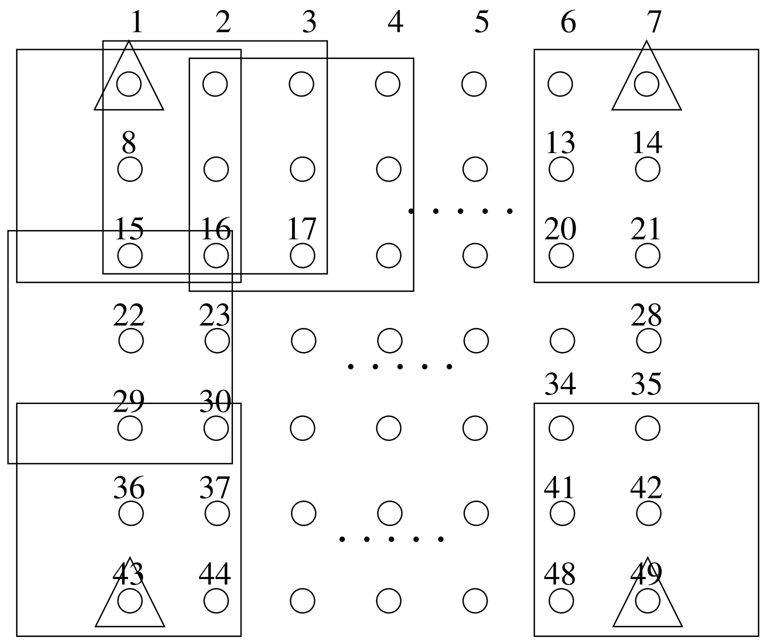
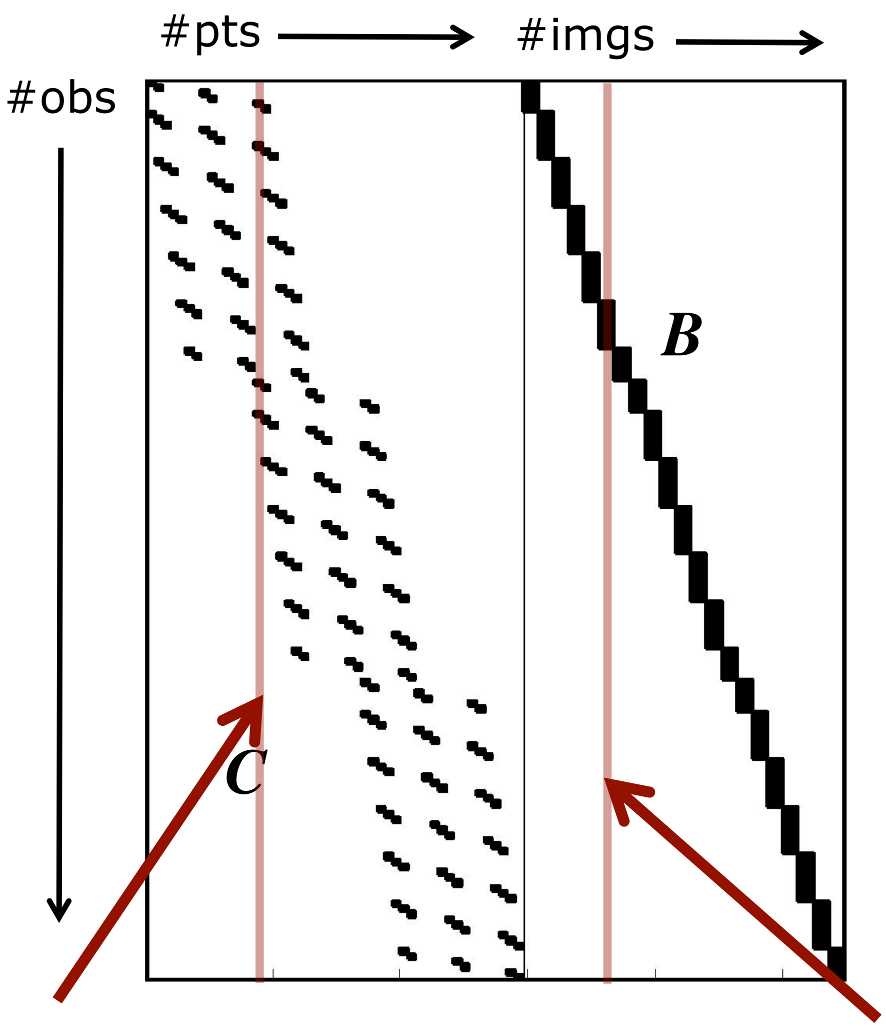
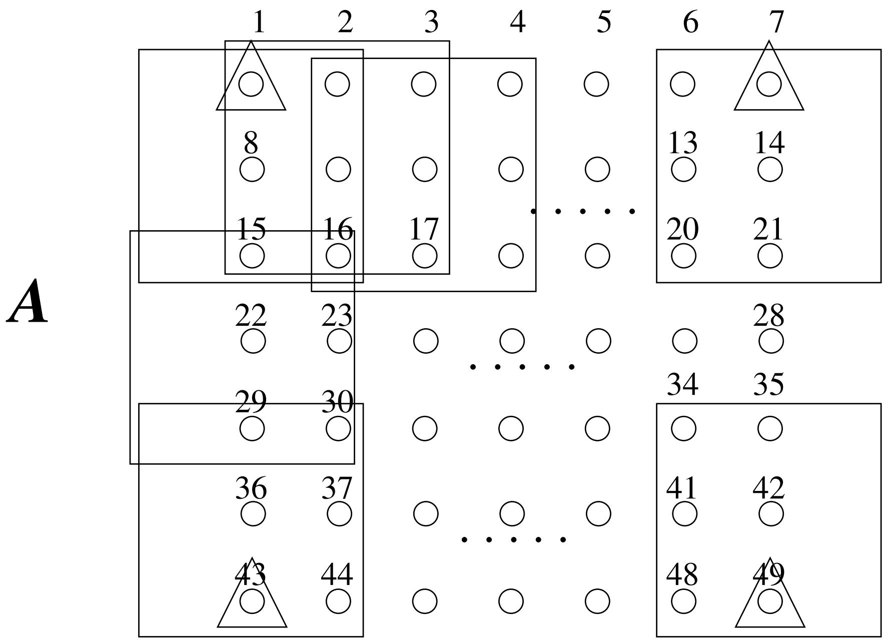


Image courtesy: Förstner

Coefficient Matrix



For 3 stripes with
7 images per stripe

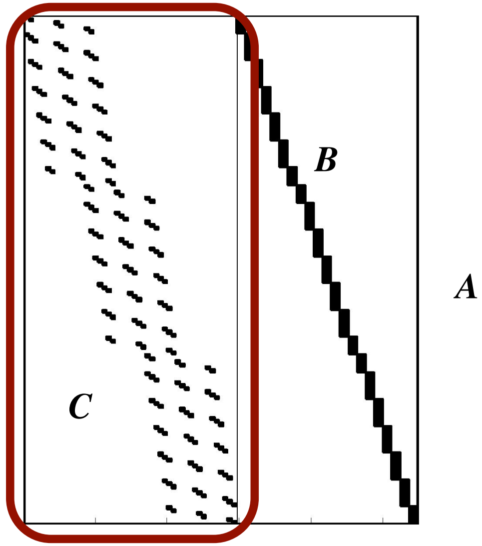


number of times a point is observed

number of observations (points) per image

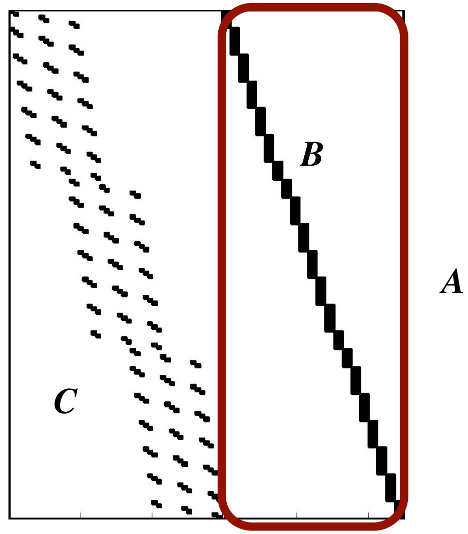
Properties of C

- The matrix C consists of 2x3 sub-matrices C_{ij}^{T}
- The sub-matrices connect image point $\boldsymbol{x}'_{ij}$ with $\boldsymbol{X}_i$



- 1 non-zero 2x3 matrix per "row" in C
- The number of non-zero 2x3 matrices per "column" is the number of times the points $\boldsymbol{X}_i$ has been observed

Properties of B



- The matrix B consists of 2x6 sub-matrices B_{ij}^{T}
- The sub-matrices connect and image point $\boldsymbol{x}'_{ij}$ and the j^{th} camera orientation
- 1 non-zero 2x6 matrix per “row” in B
- The number of non-zero 2x6 matrices per “column” is the number of image points $\boldsymbol{x}'_{ij}$ in the j^{th} image

Submatrices B and C for the Normal Case (s. Photo 1 – P3P)

- The sub-matrices of B and C are the result of the linearization (Jacobians)
- See Photogrammetry I (P3P):

$$B_{ij}^{\mathsf{T}} = \begin{bmatrix} -\frac{c_j}{Z_i - Z_{Oj}} & 0 & -\frac{x'_i}{Z_i - Z_{Oj}} & \frac{x'_i y'_i}{c_j} & -c_j\left(1 + \frac{x_i'^2}{c_j^2}\right) & y'_i \\ 0 & -\frac{c_j}{Z_i - Z_{Oj}} & -\frac{y'_i}{Z_i - Z_{Oj}} & c_j\left(1 + \frac{y_i'^2}{c_j^2}\right) & -\frac{x'_i y'_i}{c_j} & -x'_i \end{bmatrix}$$

- and

$$C_{ij}^{\mathsf{T}} = \begin{bmatrix} \frac{c_j}{Z_i - Z_{Oj}} & 0 & \frac{x'_i}{Z_i - Z_{Oj}} \\ 0 & \frac{c_j}{Z_i - Z_{Oj}} & \frac{y'_i}{Z_i - Z_{Oj}} \end{bmatrix}$$

Submatrices B and C
for the General Case

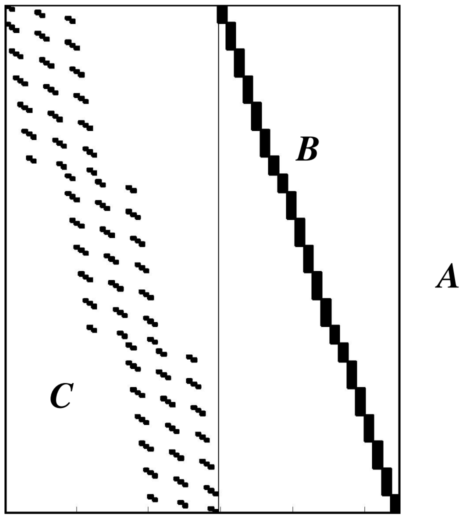
- Computing the Jacobians for the general case is more demanding
- In practice, one uses math tools

Two common ways:

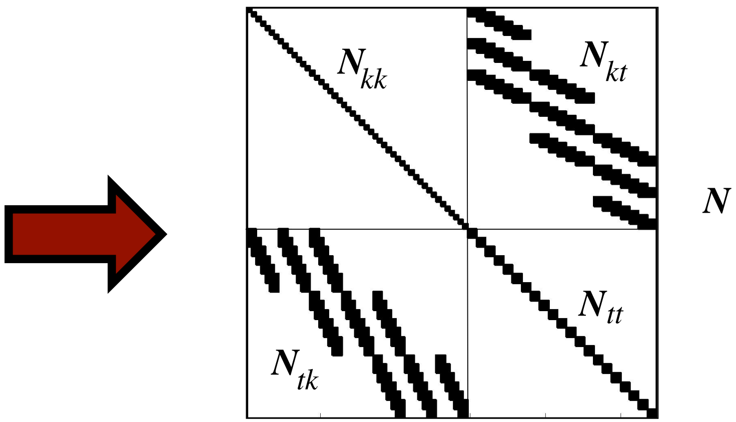
- #1: Compute Jacobians analytically
- #2: Compute Jacobians numerically (done fully automatically)

Normal Matrix

We also obtain a sparse normal matrix



A



$A^{\mathsf{T}} \Sigma_{ll}^{-1} A$

Image courtesy: Förstner

Coefficient Matrix Example

For 7 stripes with 15 images per stripe

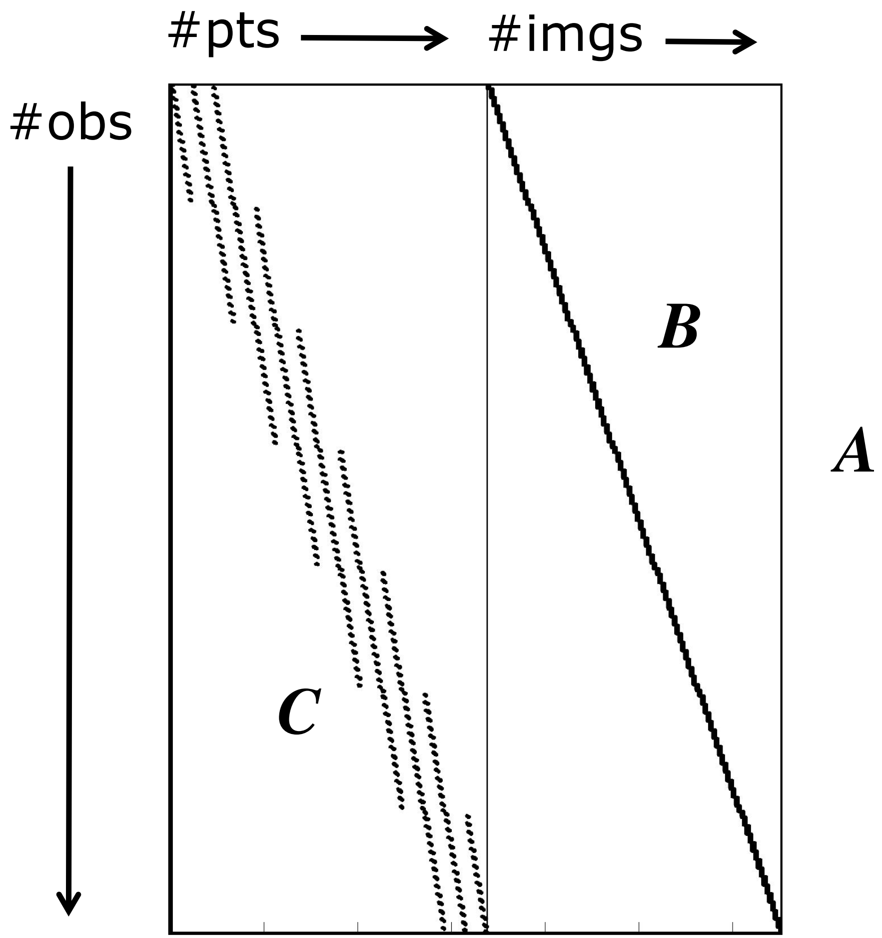
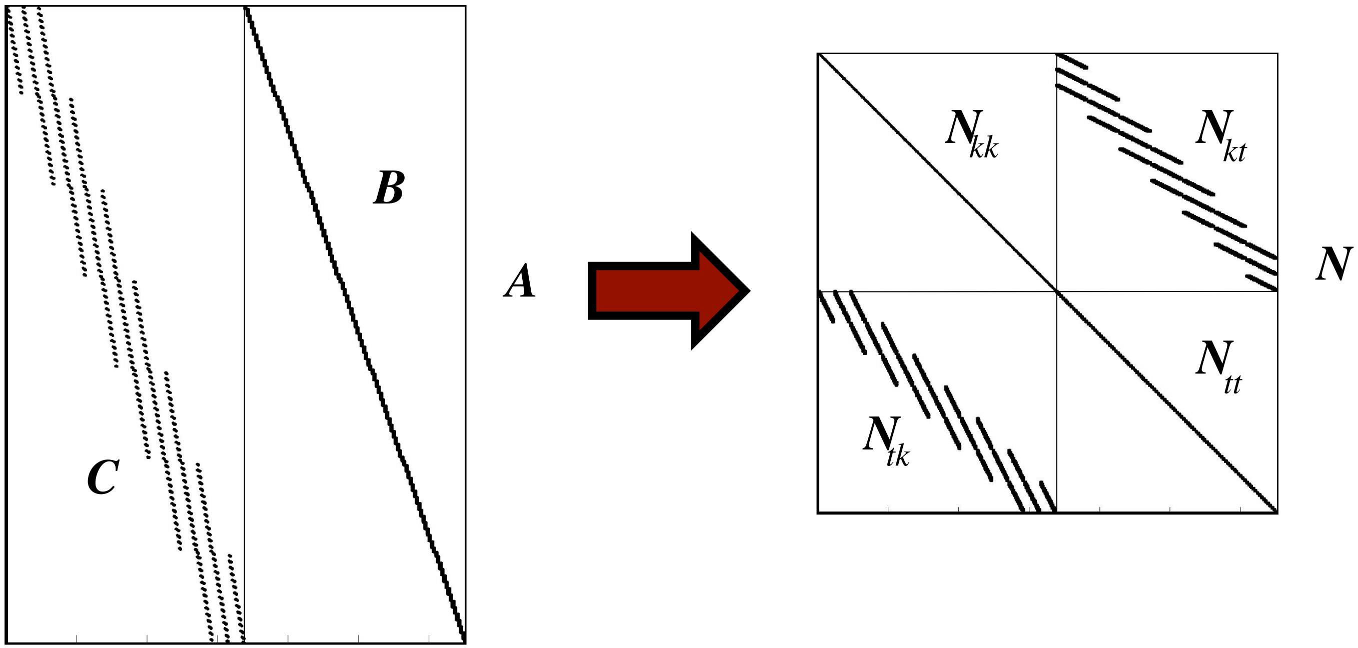


Image courtesy: Förstner

Normal Matrix

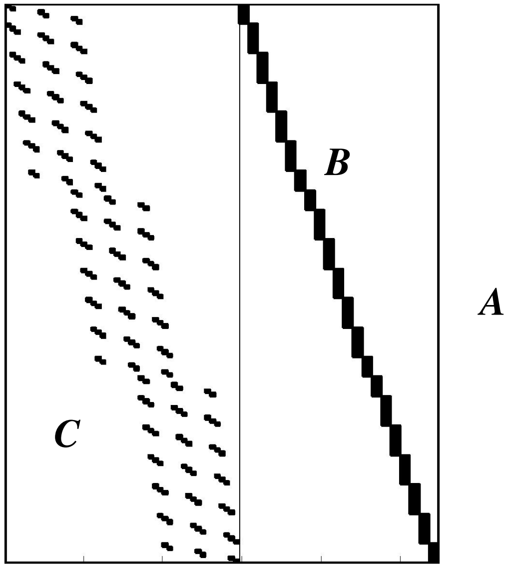
Again, we obtain a sparse normal matrix

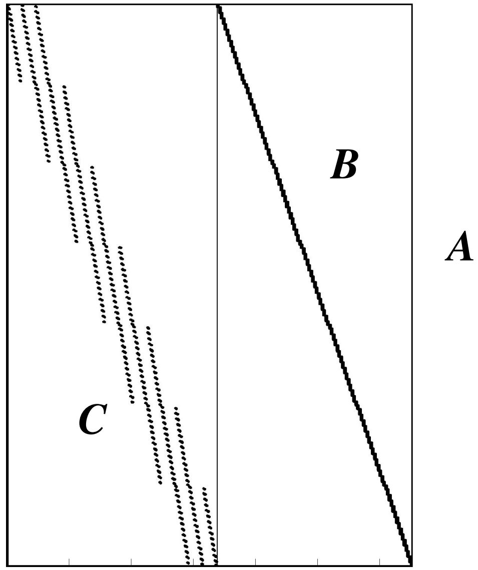


$$A \qquad\qquad A^{\mathsf{T}} \Sigma_{ll}^{-1} A$$

Image courtesy: Förstner

Sparse Coefficient Matrix





A: 3.3% non-zero

N: 9.8% non-zero

A: 0.7% non-zero

N: 2.2% non-zero

Image courtesy: Förstner

Normal Matrix

- We assume a block-diagonal cov. matrix for the observations $\Sigma_{ll} = \mathrm{Diag}\left(\underset{2\times 2}{\Sigma_{ij}}\right)$
- We obtain the normal matrix

$$N = A^\mathsf{T}\Sigma_{ll}^{-1}A = \begin{bmatrix} C^\mathsf{T} \\ B^\mathsf{T} \end{bmatrix} \Sigma_{ll}^{-1}[C, B]$$

$$= \begin{bmatrix} C^\mathsf{T}\Sigma_{ll}^{-1}C & C^\mathsf{T}\Sigma_{ll}^{-1}B \\ B^\mathsf{T}\Sigma_{ll}^{-1}C & B^\mathsf{T}\Sigma_{ll}^{-1}B \end{bmatrix} = \begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix}$$

Normal Matrix

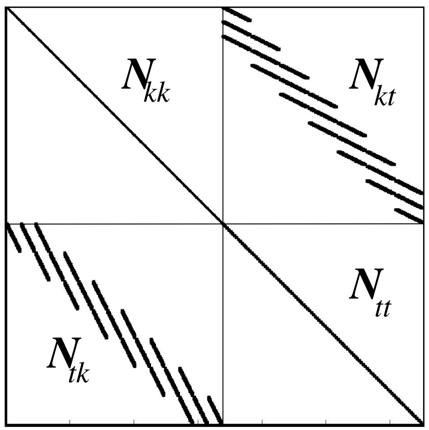
block-diagonal
with 3x3 blocks

sparse matrix that
reveals the connections
of images and points;
consist of 3x6 blocks

$$N = \begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix}$$

block-diagonal
with 6x6 blocks

Example:



Normal Matrix

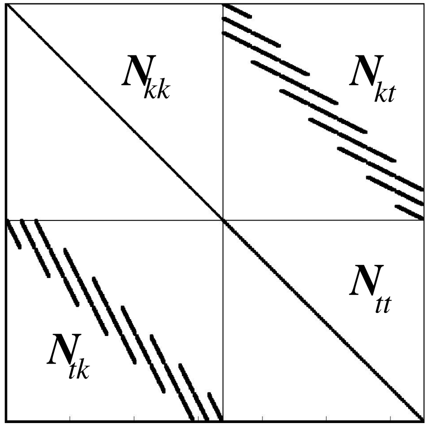
$$N = \begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix}$$

$$\underset{3N_P \times 3N_P}{N_{kk}} = \mathrm{Diag}\left(N_{k_i k_i}\right)$$

$$\underset{6N_B \times 6N_B}{N_{tt}} = \mathrm{Diag}\left(N_{t_j t_j}\right)$$

$$N_{k_i t_j} = C_{ij} \Sigma_{ij}^{-1} B_{ij}^{\mathsf{T}}$$

Example:



$$N_{k_i k_i} = \sum_{j \in \mathcal{B}_i} C_{ij} \Sigma_{ij}^{-1} C_{ij}^{\mathsf{T}}$$

all images in which point i is observed

$$N_{t_j t_j} = \sum_{i \in \mathcal{P}_j} B_{ij} \Sigma_{ij}^{-1} B_{ij}^{\mathsf{T}}$$

all points that are observed in image j

Orientation Parameters Only

- If we want to compute the orientation parameters only, we proceed:

$$
\begin{aligned}
N\Delta\boldsymbol{x} &= \boldsymbol{h} \\
\begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{k} \\ \Delta\boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{h}_k \\ \boldsymbol{h}_t \end{bmatrix}
\end{aligned}
$$

Orientation Parameters Only

- If we want to compute the orientation parameters only, we proceed:

$$
\begin{aligned}
N \Delta \boldsymbol{x} &= \boldsymbol{h} \\
\begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{h}_k \\ \boldsymbol{h}_t \end{bmatrix} \\
\begin{bmatrix} N_{kk}^{-1} & 0 \\ -N_{tk} N_{kk}^{-1} & I \end{bmatrix} \begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} N_{kk}^{-1} & 0 \\ -N_{tk} N_{kk}^{-1} & I \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_k \\ \boldsymbol{h}_t \end{bmatrix}
\end{aligned}
$$

Orientation Parameters Only

- If we want to compute the orientation parameters only, we proceed:

$$
\begin{aligned}
N\Delta \boldsymbol{x} &= \boldsymbol{h} \\
\begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{h}_k \\ \boldsymbol{h}_t \end{bmatrix} \\
\begin{bmatrix} N_{kk}^{-1} & 0 \\ -N_{tk}N_{kk}^{-1} & I \end{bmatrix} \begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} N_{kk}^{-1} & 0 \\ -N_{tk}N_{kk}^{-1} & I \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_k \\ \boldsymbol{h}_t \end{bmatrix} \\
\begin{bmatrix} I & N_{kk}^{-1}N_{kt} \\ 0 & N_{tt} - N_{tk}N_{kk}^{-1}N_{kt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} N_{kk}^{-1}\boldsymbol{h}_k \\ \boldsymbol{h}_t - N_{tk}N_{kk}^{-1}\boldsymbol{h}_k \end{bmatrix}
\end{aligned}
$$

Orientation Parameters Only

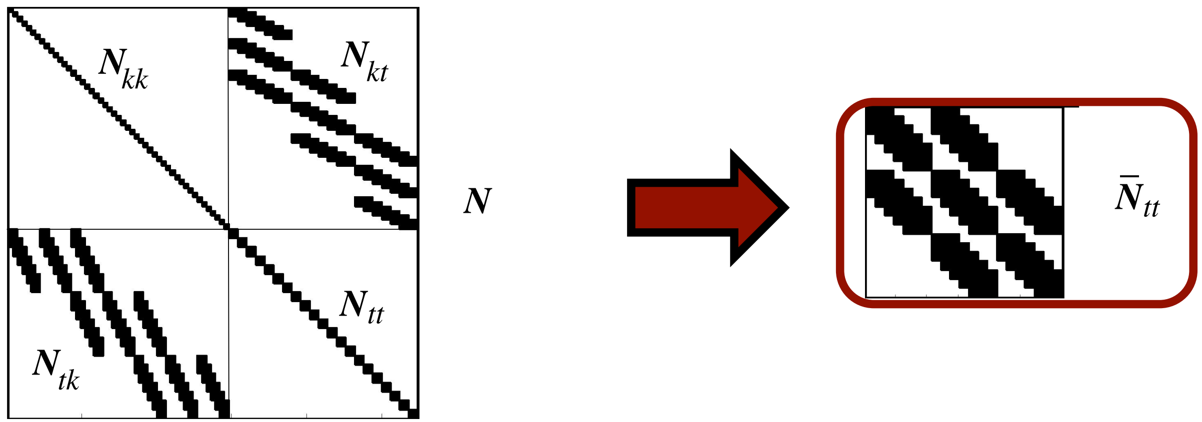
- If we want to compute the orientation parameters only, we proceed:

$$
\begin{aligned}
N \Delta \boldsymbol{x} &= \boldsymbol{h} \\
\begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{h}_k \\ \boldsymbol{h}_t \end{bmatrix} \\
\begin{bmatrix} N_{kk}^{-1} & 0 \\ -N_{tk} N_{kk}^{-1} & I \end{bmatrix} \begin{bmatrix} N_{kk} & N_{kt} \\ N_{tk} & N_{tt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} N_{kk}^{-1} & 0 \\ -N_{tk} N_{kk}^{-1} & I \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_k \\ \boldsymbol{h}_t \end{bmatrix} \\
\begin{bmatrix} I & N_{kk}^{-1} N_{kt} \\ 0 & N_{tt} - N_{tk} N_{kk}^{-1} N_{kt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} &= \begin{bmatrix} N_{kk}^{-1} \boldsymbol{h}_k \\ \boldsymbol{h}_t - N_{tk} N_{kk}^{-1} \boldsymbol{h}_k \end{bmatrix}
\end{aligned}
$$

$$
\overline{N}_{tt} \Delta \boldsymbol{t} = \overline{\boldsymbol{h}}_t
$$

Reduced Normal System

- The reduced normal system $\overline{N}_{tt}\Delta \boldsymbol{t} = \overline{\boldsymbol{h}}_t$ is independent of the number of points



- The reduced system is still sparse
- Here, it is a 126x126 matrix (square matrix, size #obs by #obs)

(Reduced) Normal Matrix

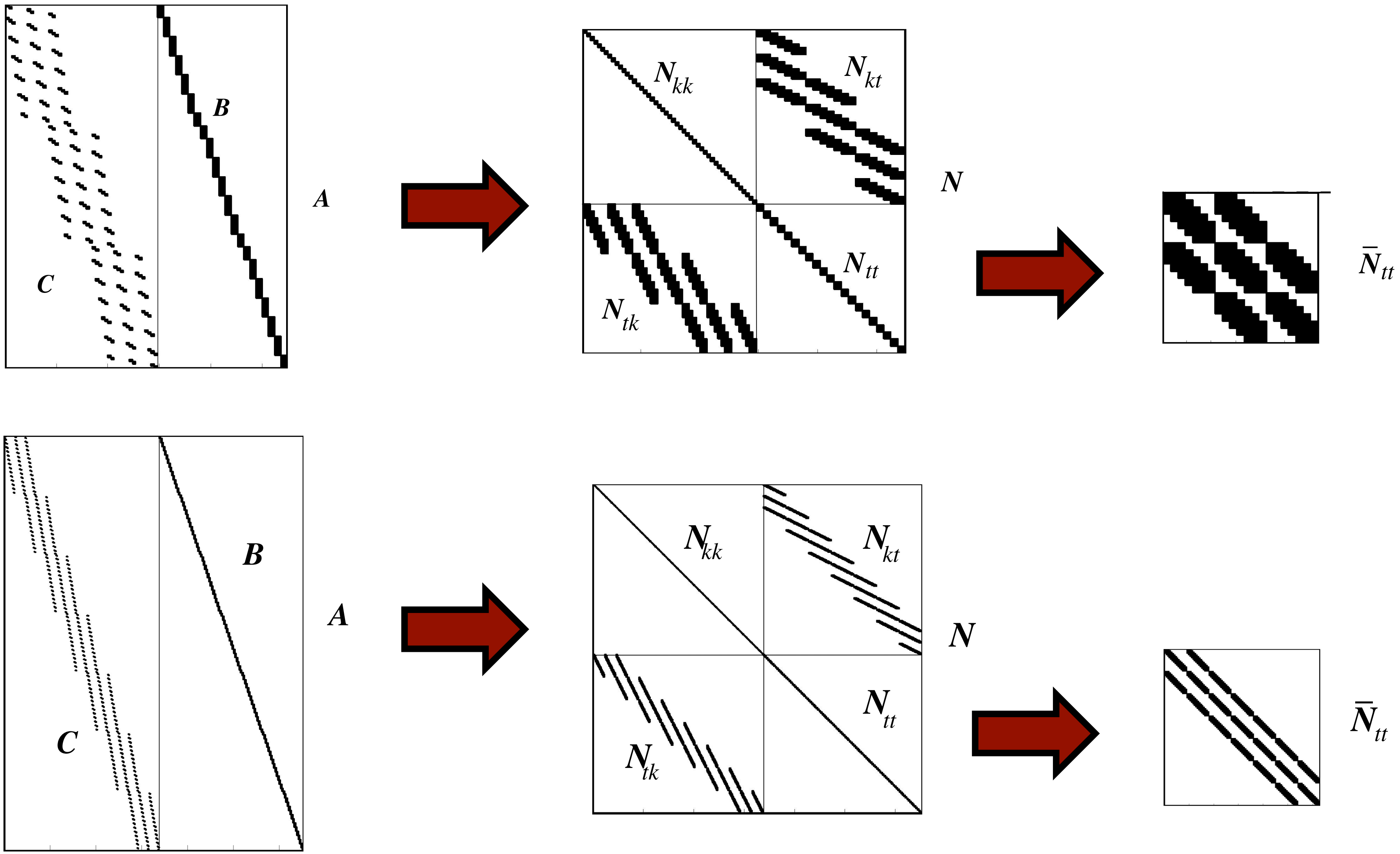


Image courtesy: Förstner

Reduced Normal System

- The reduced normal system is

$$\overline{N}_{tt} \Delta \boldsymbol{t} = \overline{\boldsymbol{h}}_t$$

- with

$$\overline{N}_{tt} = N_{tt} - N_{tk} \boxed{N_{kk}^{-1}} N_{kt} \qquad \overline{\boldsymbol{h}}_t = \boldsymbol{h}_t - N_{tk} \boxed{N_{kk}^{-1}} \boldsymbol{h}_k$$

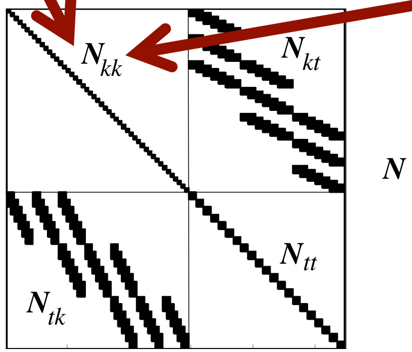
Reduced Normal System

- The reduced normal system is

$$\overline{N}_{tt}\Delta \boldsymbol{t} = \overline{\boldsymbol{h}}_t$$

- with

$$\overline{N}_{tt} = N_{tt} - N_{tk}N_{kk}^{-1}N_{kt} \qquad \overline{\boldsymbol{h}}_t = \boldsymbol{h}_t - N_{tk}N_{kk}^{-1}\boldsymbol{h}_k$$



easy to invert!

- Solve $\overline{N}_{tt}\Delta \boldsymbol{t} = \overline{\boldsymbol{h}}_t$ using a sparse solver

Obtaining 3D Points Given Δt

- We had

$$\begin{bmatrix} I & N_{kk}^{-1} N_{kt} \\ 0 & N_{tt} - N_{tk} N_{kk}^{-1} N_{kt} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{k} \\ \Delta \boldsymbol{t} \end{bmatrix} = \begin{bmatrix} N_{kk}^{-1} \boldsymbol{h}_k \\ \boldsymbol{h}_t - N_{tk} N_{kk}^{-1} \boldsymbol{h}_k \end{bmatrix}$$

- and solved for Δt
- This directly leads to

$$\Delta \boldsymbol{k} + N_{kk}^{-1} N_{kt} \Delta \boldsymbol{t} = N_{kk}^{-1} \boldsymbol{h}_k$$

- Thus, we can can compute the point coordinates given the orientations Δt :

$$\Delta \boldsymbol{k} = N_{kk}^{-1} (\boldsymbol{h}_k - N_{kt} \Delta \boldsymbol{t})$$

Building the Normal Equation

- The full Jacobian/coefficient matrix A does not need to be constructed explicitly
- We directly construct N by

$$N_{k_i k_i} = \sum_{j \in \mathcal{B}_i} C_{ij} \Sigma_{ij}^{-1} C_{ij}^\top \qquad N_{k_i t_j} = C_{ij} \Sigma_{ij}^{-1} B_{ij}^\top$$

$$N_{t_j t_j} = \sum_{i \in \mathcal{P}_j} B_{ij} \Sigma_{ij}^{-1} B_{ij}^\top$$

- and construct the reduced system

$$\overline{N}_{tt} = N_{tt} - N_{tk} N_{kk}^{-1} N_{kt} \qquad \overline{\boldsymbol{h}}_t = \boldsymbol{h}_t - N_{tk} N_{kk}^{-1} \boldsymbol{h}_k$$

- Solve it with a sparse solver
- Compute the points coordinates

$$\Delta \boldsymbol{k} = N_{kk}^{-1} (\boldsymbol{h}_k - N_{kt} \Delta \boldsymbol{t})$$

BA Without Control Points

- In case no control points are provided, the reference frame is not defined
- BA will only be able to correct the geometry up to a similarity transform

Problems?

BA Without Control Points

- In case no control points are provided, the reference frame is not defined
- BA will only be able to correct the geometry up to a similarity transform
- Normal equations with rank deficiency of 7
- Gauge-freedom
- We have to specify a datum

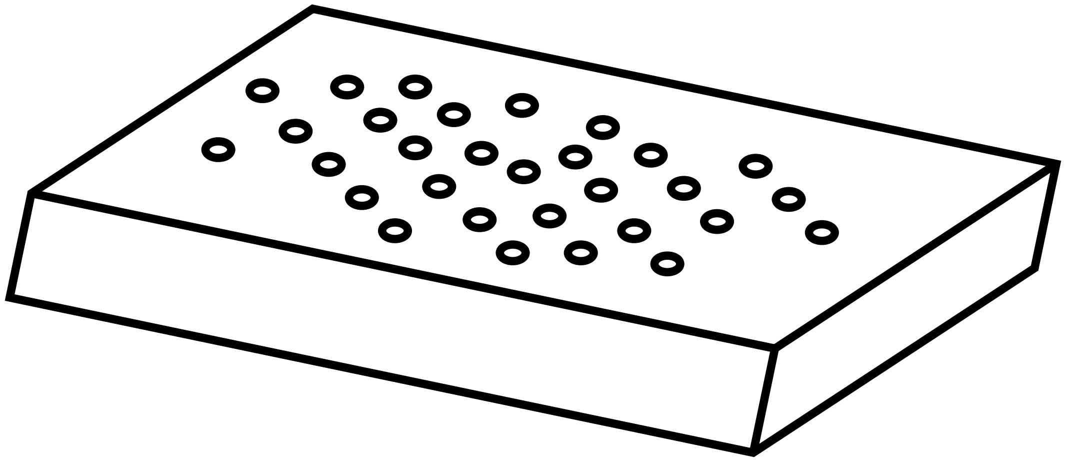
Datum Without Control Points

- Datum through additional constraints

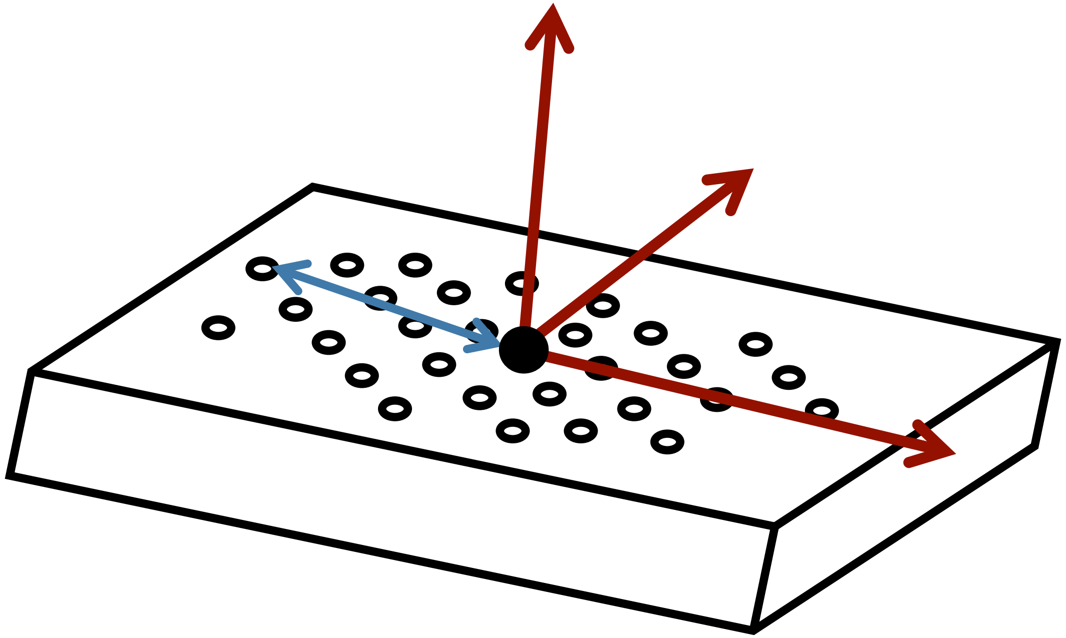
Constraints:

- The center of mass of the 3D points should not change (translation)
- No change in the main directions (rotation)
- No change in the average distance to the center of mass (scale)

Reference Frame



Reference Frame



Constraints

- Constraint can be expressed through a constraint matrix H with $H\widehat{\boldsymbol{x}} = \mathbf{0}$
- The Jacobian H is added and thus considered in the error minimization

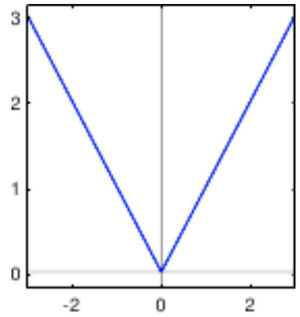
Constraints

- We use $H^{\mathsf{T}} W \widehat{\Delta \boldsymbol{x}}_k = \mathbf{0}$
- Weight matrix W for the points
- W is often the identity or an indicator function (to deactivate certain points)

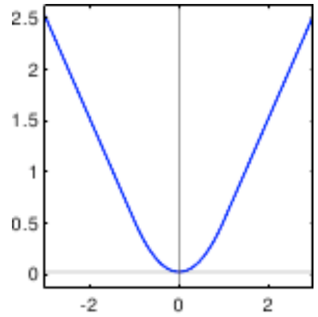
$$\begin{array}{cc}
 & \begin{array}{ccc} \text{translation} \qquad & \qquad \text{rotation} \qquad & \qquad \text{scale} \end{array} \\
\underset{3N\times 7}{H} = & \left[\begin{array}{ccc|ccc|c}
1 & 0 & 0 & 0 & -Z_1 & Y_1 & X_1 \\
0 & 1 & 0 & Z_1 & 0 & -X_1 & Y_1 \\
0 & 0 & 1 & -Y_1 & X_1 & 0 & Z_1 \\
\hline
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\hline
1 & 0 & 0 & 0 & -Z_n & Y_n & X_n \\
0 & 1 & 0 & Z_n & 0 & -X_n & Y_n \\
0 & 0 & 1 & -Y_n & X_n & 0 & Z_n \\
\hline
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\hline
1 & 0 & 0 & 0 & -Z_N & Y_N & X_N \\
0 & 1 & 0 & Z_N & 0 & -X_N & Y_N \\
0 & 0 & 1 & -Y_N & X_N & 0 & Z_N
\end{array}\right]
\end{array}$$

A Remark on Outliers

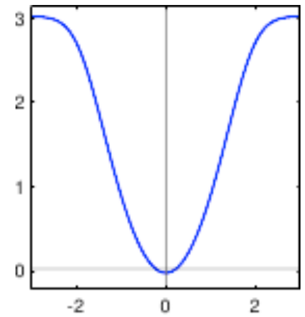
- See BA – Part 1 lecture on how to reduce the risk of outlier observations
- Furthermore, we use robust kernels
- Instead of using a Gaussian noise model, consider a robustified version
- Reduce "penalty" far away from 0



L1 norm



Huber

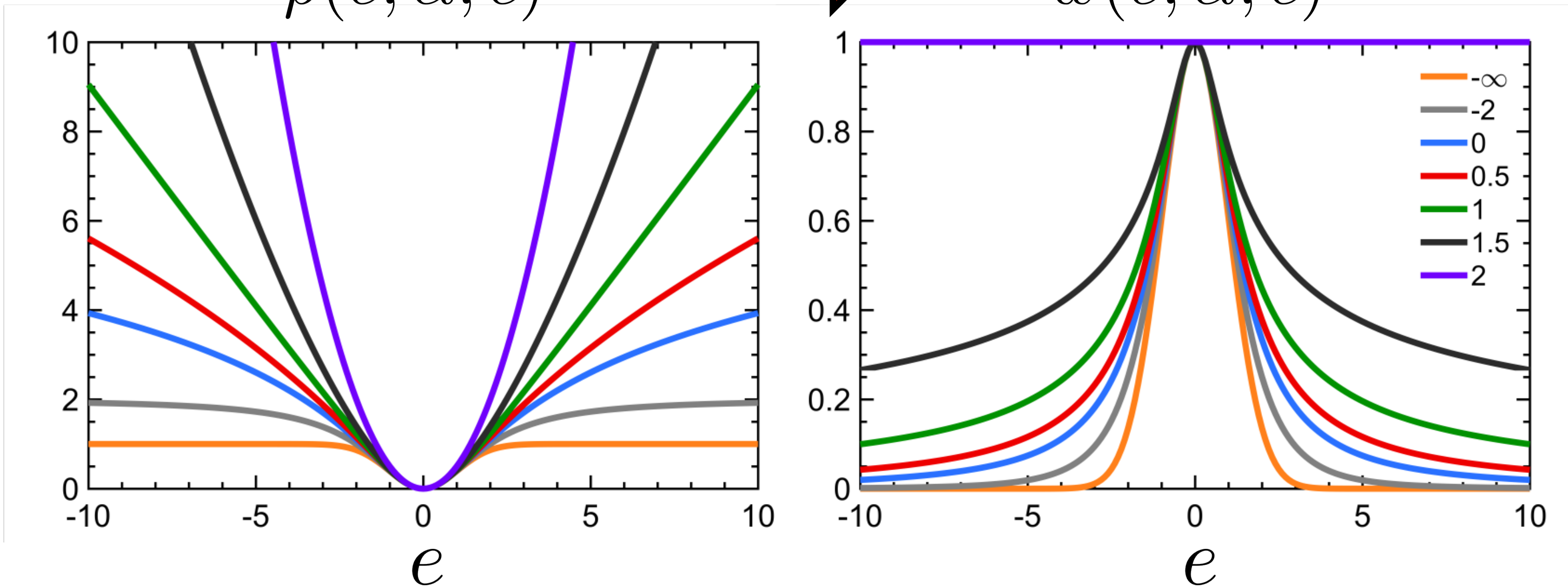


Blake-Zisserman

Robust Kernels

A robust kernel leads to weighted LS

$\rho(e, \alpha, c)$ $\Rightarrow$ $w(e, \alpha, c)$



Source: Barron, A General and Adaptive Robust Loss Function, CVPR 2019

BA Numerics Summary

- Bundle Adjustment = least squares solution to relative and absolute orientation considering uncertainties
- We have to solve a large system
- BA leads to sparse matrices
- Using sparse solvers is key
- Often sequential solution of orientation parameters first and then the point coordinates

Slide Information

- The slides have been created by Cyrill Stachniss as part of the photogrammetry and robotics courses.
- I tried to acknowledge all people from whom I used images or videos. In case I made a mistake or missed someone, please let me know.
- The photogrammetry material heavily relies on the very well written lecture notes by Wolfgang Förstner and the Photogrammetric Computer Vision book by Förstner & Wrobel.
- Parts of the robotics material stems from the great Probabilistic Robotics book by Thrun, Burgard and Fox.
- If you are a university lecturer, feel free to use the course material. If you adapt the course material, please make sure that you keep the acknowledgements to others and please acknowledge me as well. To satisfy my own curiosity, please send me email notice if you use my slides.

Cyrill Stachniss, cyrill.stachniss@igg.uni-bonn.de, 2014