Computational Optical Imaging - Optique Numerique

-- Calibration and Features --

Winter 2013

Ivo Ihrke

with slides by Thorsten Thormaehlen

Estimating the Camera Matrix

Ivo Ihrke / Winter 2013

3D Calibration target

DLT algorithm

INSTITUT d'OPTIQUE GRADUATE SCHOOL

- Have correspondences: $\mathbf{x} \Leftrightarrow \mathbf{X}$; unkown $\mathtt{A}$
- Projecting a point: $\mathbf{x} = \mathtt{A}\mathbf{X}$
- Performing the perspective division yields

$$x = \frac{a_{11}X + a_{12}Y + a_{13}Z + a_{14}}{a_{31}X + a_{32}Y + a_{33}Z + a_{34}}$$

$$y = \frac{a_{21}X + a_{22}Y + a_{23}Z + a_{24}}{a_{31}X + a_{32}Y + a_{33}Z + a_{34}}$$

$$\begin{aligned} \Rightarrow a_{31}Xx + a_{32}Yx + a_{33}Zx + a_{34}x - a_{11}X - a_{12}Y - a_{13}Z - a_{14} &= 0 \\ a_{31}Xy + a_{32}Yy + a_{33}Zy + a_{34}y - a_{21}X - a_{22}Y - a_{23}Z - a_{24} &= 0 \end{aligned}$$

$$\begin{bmatrix} -X & -Y & -Z & -1 & 0 & 0 & 0 & 0 & Xx & Yx & Zx & x \\ 0 & 0 & 0 & 0 & -X & -Y & -Z & -1 & Xy & Yy & Zy & y \\ & & & & & \vdots & & & & & & \end{bmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ \vdots \end{pmatrix} = \mathtt{B}\mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ \vdots \end{pmatrix}$$

DLT algorithm II

- Solve homogeneous system $\mathbf{Ba} = \mathbf{0}$

  (avoid trivial solution **a**=**0**)

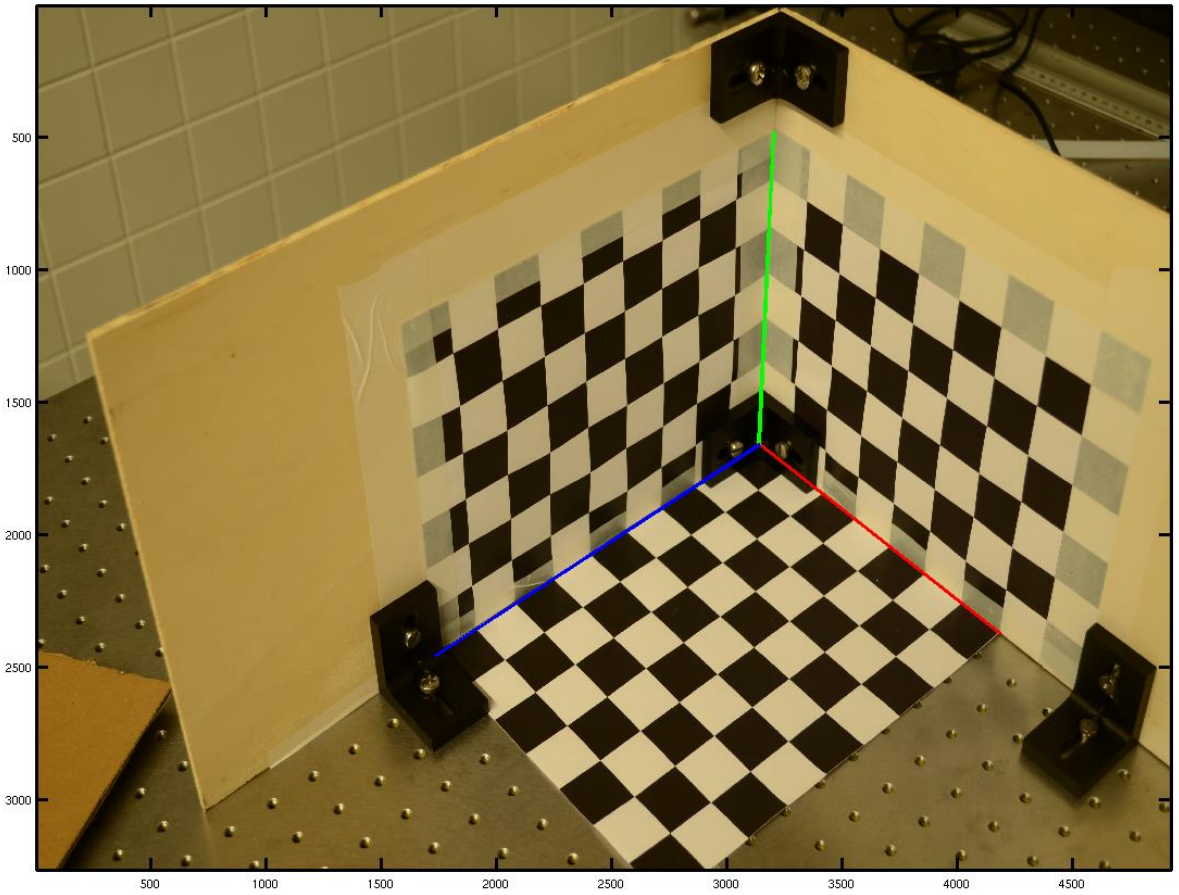
- Perform SVD: $\mathbf{UDV}^{\mathrm{T}} = \mathrm{svd}(\mathbf{B})$

- Extract lower-most row of $\mathbf{V}^{\mathrm{T}}$

  (right-most column of $\mathbf{V}$)

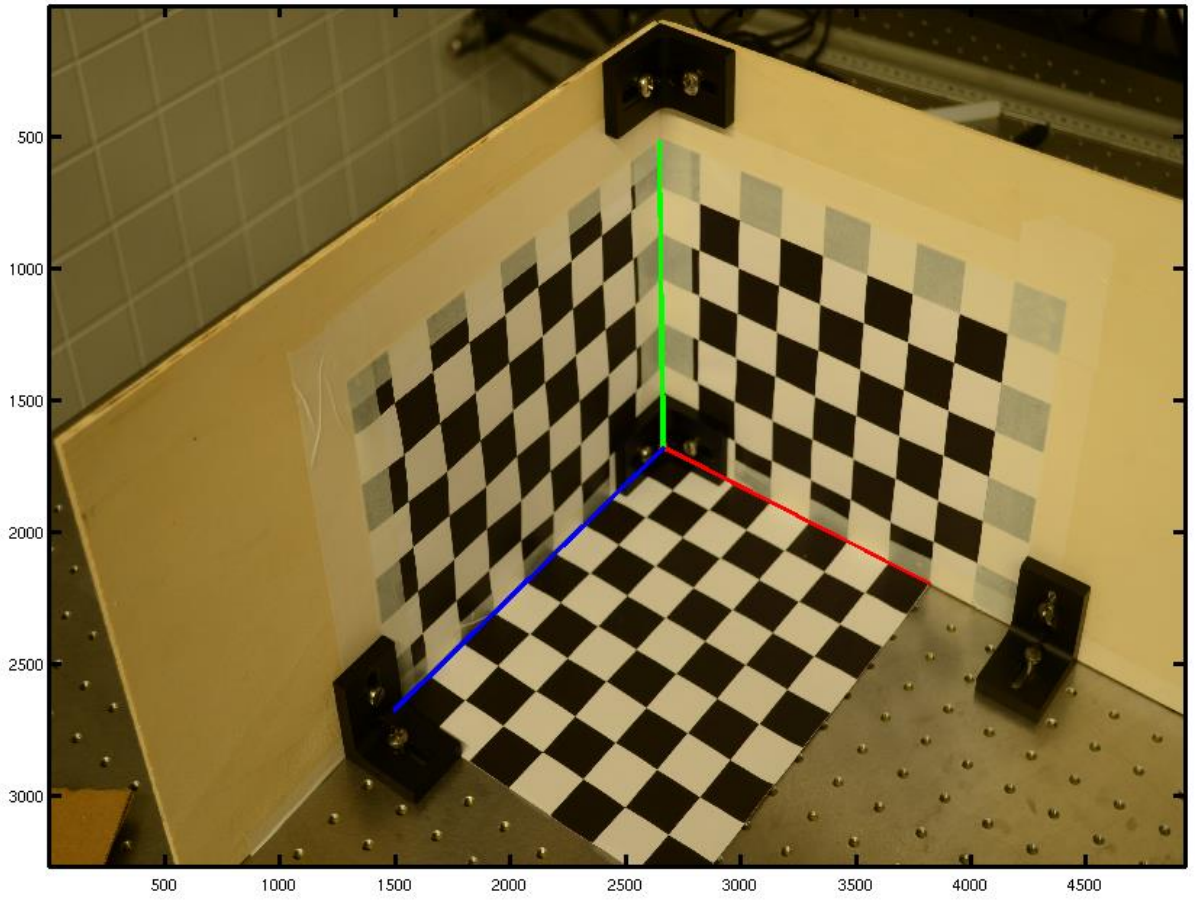
- Yields least-squares optimal approximation to null-space

3D Target – calibrated coordinate system



Plotted 3D points: (0,0,0), (7,0,0), (0,7,0), (0,0,-9)

3D Target – calibrated coordinate system



Plotted 3D points: (0,0,0), (7,0,0), (0,7,0), (0,0,-9)

Things to note:

INSTITUT d'OPTIQUE GRADUATE SCHOOL

- Computer vision coordinate systems are often left-handed
- OpenGL systems are right-handed
- Can simply load A into matrix stack
  - BUT: coordinate system (and clipping) might behave strangely
  - Convert by “flip matrix”


500
1000
1500
2000
2500
3000
500
1000
1500
2000
2500
3000
3500
4000
4500


500
1000
1500
2000
2500
3000
500
1000
1500
2000
2500
3000
3500
4000
4500

Accuracy considerations

- Extract focal length from intrinsic matrix K

  (K11+K22)/2 * pixel size (Nikon D7000: 4.78 um)

- View 1: f = 33.65mm
  - EXIF info: 34.0 mm

$$K_1 = \begin{pmatrix} 7020.09 & 1.09 & 2536.22 \\ 0 & 7061.12 & 1576.92 \\ 0 & 0 & 1.0 \end{pmatrix}$$

- View 2: f = 32.93mm
  - EXIF info: 34.0 mm

$$K_2 = \begin{pmatrix} 6870.53 & -11.58 & 2477.79 \\ 0 & 6908.17 & 1595.35 \\ 0 & 0 & 1.0 \end{pmatrix}$$

Accuracy considerations

- Extract focal length from intrinsic matrix K
- Variance on image measurements:
  - sigma = 2 pixels

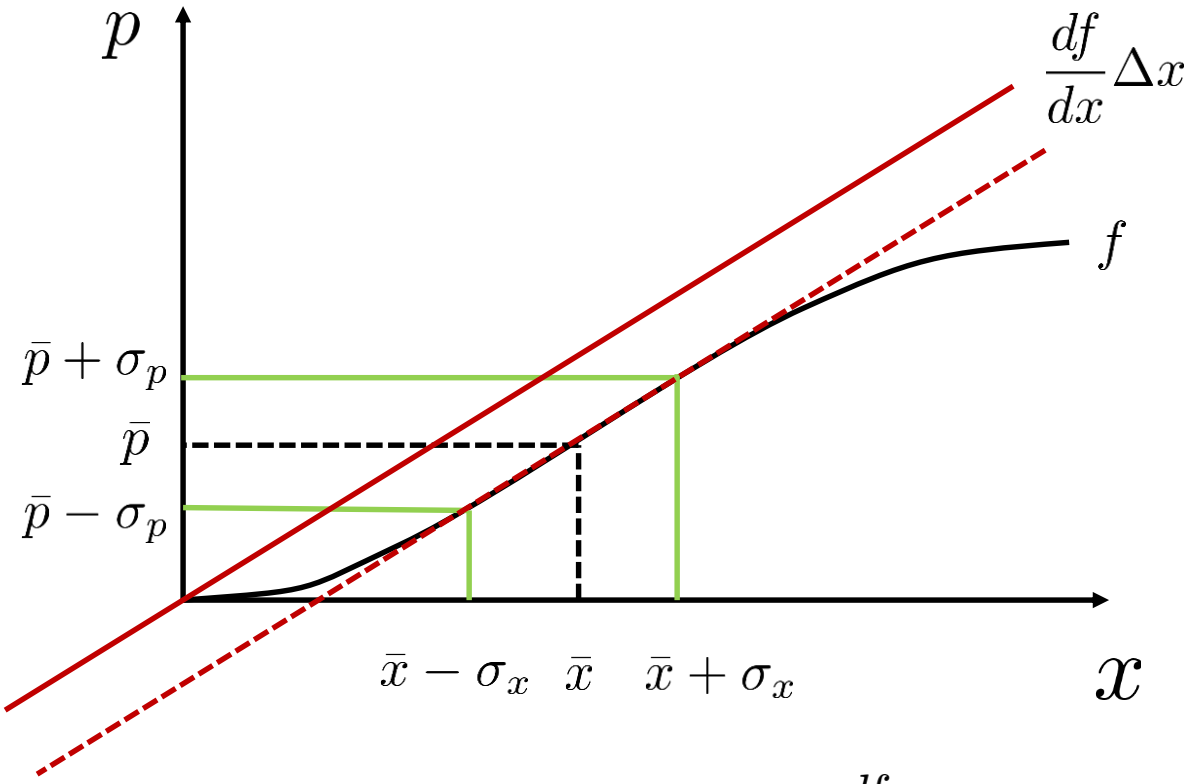
- View 1: f = 33.65 (sigma = 0.67mm)
- EXIF info: 34.0 mm

- View 2: f = 32.93mm (sigma = 0.72mm)
  - EXIF info: 34.0 mm

How to estimate the parameter covariance – analytical - I

- 1D case – mapping $f : x \in \mathbb{R} \mapsto p \in \mathbb{R}$

  from measurement space to parameter space



$$\bar{p} + \Delta_p = f(\bar{x} + \Delta_x) \approx f(\bar{x}) + \frac{df}{dx}\Delta_x + \mathcal{O}(\Delta_x^2)$$

How to estimate the parameter covariance – analytical - II

INSTITUT d'OPTIQUE GRADUATE SCHOOL

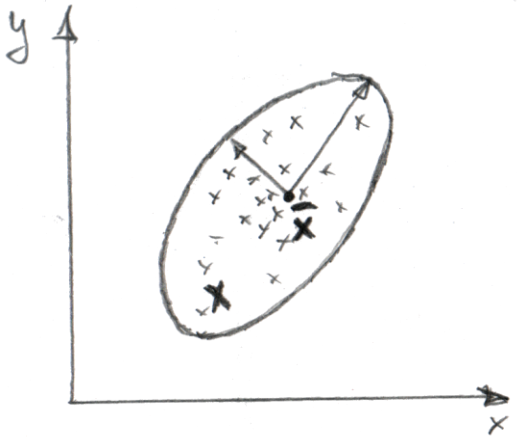
- Taylor expansion $\bar{p} + \sigma_p = f(\bar{x} + \sigma_x) \approx f(\bar{x}) + \frac{df}{dx}\sigma_x + \mathcal{O}(\sigma_x^2)$
  - Valid as long as linearization within $[\bar{x} - 2\sigma_x, \bar{x} + 2\sigma_x]$ holds

$$\Rightarrow \sigma_p \approx \frac{df}{dx}\sigma_x \quad \text{or} \quad \frac{dx}{df}\sigma_p \approx \sigma_x$$

Tool to transfer standard deviation from measurements to model parameters

- Higher Dimensions:
  - Description of variance → covariance matrix

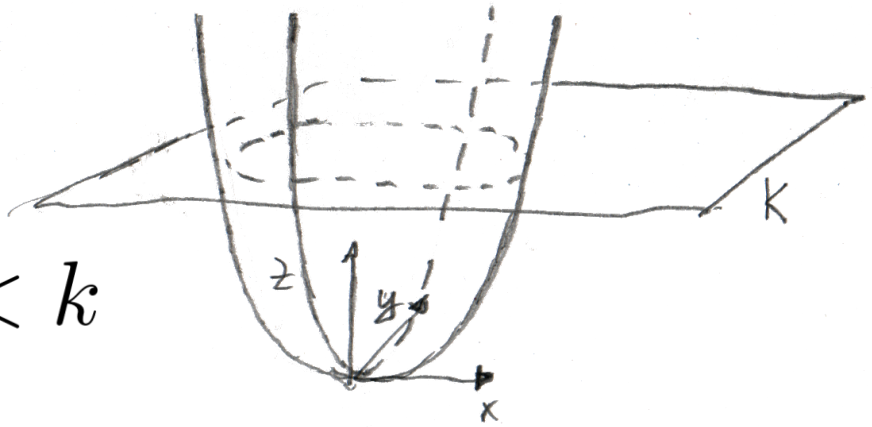
$$\Sigma_{\mathbf{x}} = \begin{pmatrix} \sigma_{x_1}^2 & \cdots & \sigma_{x_1} \cdot \sigma_{x_n} \\ \vdots & \ddots & \vdots \\ \sigma_{x_n} \cdot \sigma_{x_1} & \cdots & \sigma_{x_n}^2 \end{pmatrix}$$



How to interpret the covariance matrix

- Error ellipsoid (quadratic, positive (semi-) definite form)

$$\underbrace{(\mathbf{x}-\bar{\mathbf{x}})^T}_{\Delta\mathbf{x}^T}\Sigma_{\mathbf{x}}^{-1}\underbrace{(\mathbf{x}-\bar{\mathbf{x}})}_{\Delta\mathbf{x}} < k$$



- k determined from χ^2 distribution $(k, 2)$ [Zhang96]
- E.g. contains
  - 70% of the data for $k = 2.41$
  - 95% of the data for $k = 5.99$

How to estimate the parameter covariance – analytical - III

- Higher dimensional Taylor:

$$\bar{\mathbf{x}} + \Delta\mathbf{x} = f^{-1}(\bar{\mathbf{p}} + \Delta\mathbf{p}) \approx f^{-1}(\bar{\mathbf{p}}) + \mathrm{J}_{\mathbf{p}}\Delta\mathbf{p} + \mathcal{O}(\Delta\mathbf{p}^2)$$

with Jacobian matrix

$$\mathrm{J}_{\mathbf{p}} \in \mathbb{R}^{M\times N} := \begin{pmatrix} \frac{\partial x_1}{\partial p_1} & \cdots & \frac{\partial x_1}{\partial p_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_M}{\partial p_1} & \cdots & \frac{\partial x_M}{\partial p_N} \end{pmatrix}$$

$$\Rightarrow \Delta\mathbf{x} \approx \mathrm{J}_{\mathbf{p}}\Delta\mathbf{p}$$

$$\Rightarrow \Delta\mathbf{x}^T\Sigma_{\mathbf{x}}^{-1}\Delta x = \Delta\mathbf{p}^T \overbrace{\mathrm{J}_{\mathbf{p}}{}^T\Sigma_{\mathbf{x}}^{-1}\mathrm{J}_{\mathbf{p}}}^{\Sigma_{\mathbf{p}}^{-1}}\Delta\mathbf{p}$$

$$\Rightarrow \Sigma_{\mathbf{p}} = \left(\mathrm{J}_{\mathbf{p}}{}^T\Sigma_{\mathbf{x}}^{-1}\mathrm{J}_{\mathbf{p}}\right)^{-1}$$

Example: line fit, direct parametrization

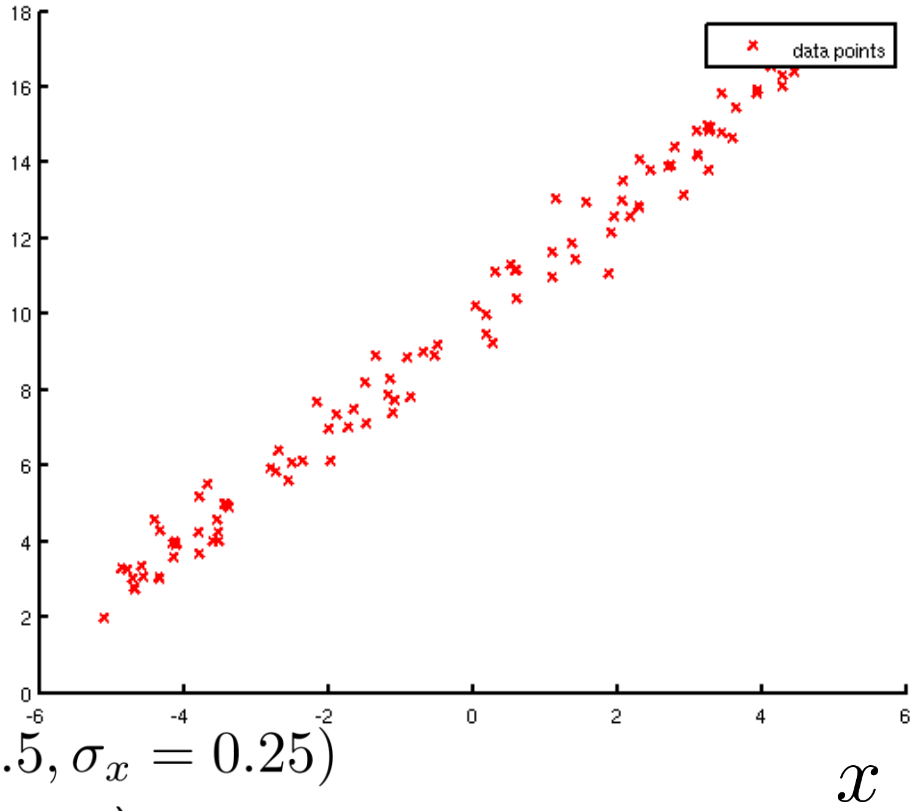
$$y = m \cdot x + n$$

y - measurements

x - measurement positions

$$\mathbf{p} = (m, n)^T$$

-parameter vector



fitting linear system $(N = 100, \sigma_y = 0.5, \sigma_x = 0.25)$

$$\mathbf{Ap} = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{pmatrix} \begin{pmatrix} m \\ n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \mathbf{y}$$

Ground truth values: $m = 1.5,\ n = 10$

Example: line fit, direct parametrization

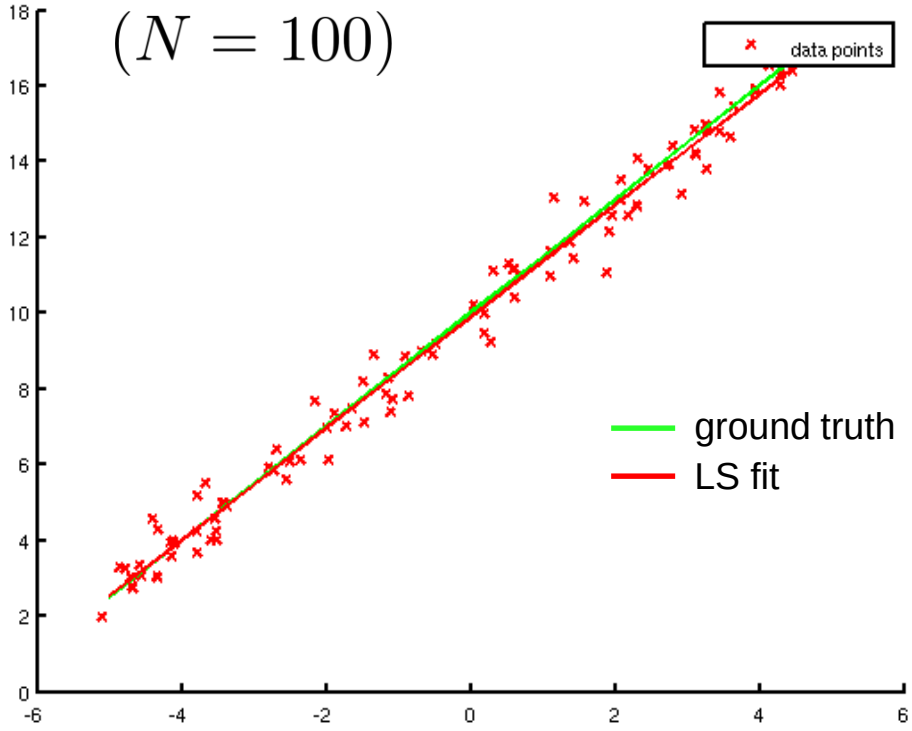
$$y = m \cdot x + n$$

y - measurements

x - measurement positions

$$\mathbf{p} = (m, n)^T$$

-parameter vector



Solve normal equations → least squares solution

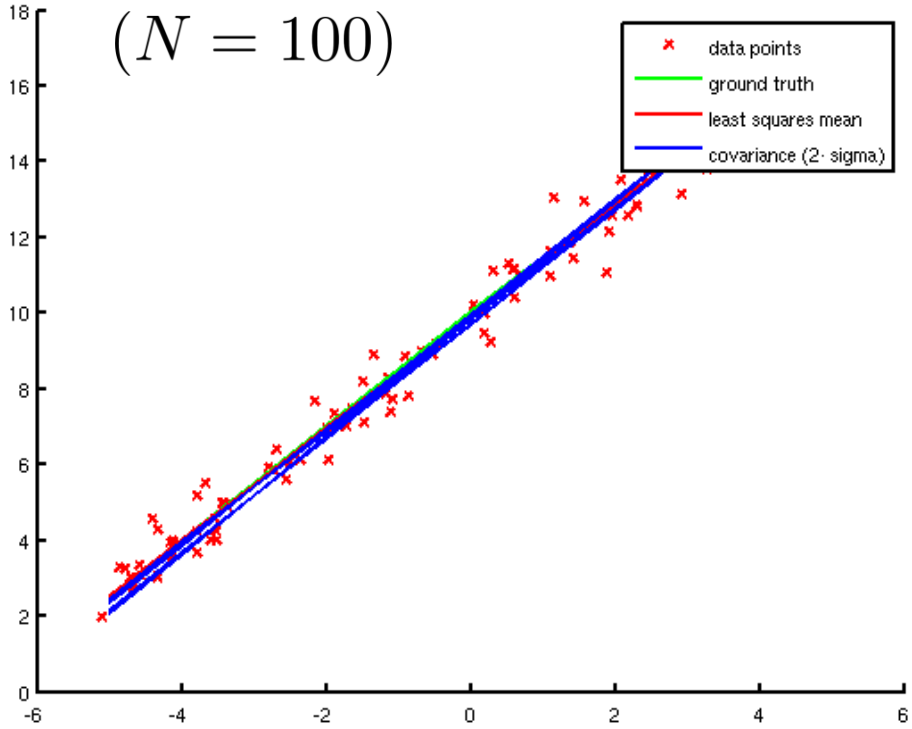
$$\mathbf{p} = (\mathtt{A}^T \mathtt{A}\mathbf{p})^{-1} \mathtt{A}^T \mathbf{y}$$

$$\Rightarrow \mathbf{p} = (1.516, 9.827)^T$$

Example: line fit, direct parametrization

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

y $(N = 100)$



Jacobian w.r.t. measurements:

$$\frac{\partial y}{\partial m} = x, \quad \frac{\partial y}{\partial n} = 1$$

$$\Rightarrow \mathrm{J}_{\mathbf{p}_y} = \begin{pmatrix} x_1 & \dots & x_N \\ 1 & \dots & 1 \end{pmatrix}^T$$

$$\Sigma_{\mathbf{y}} = \begin{pmatrix} \sigma_{y_1}^2 & 0 & \dots & 0 \\ 0 & \sigma_{y_2}^2 & \dots & 0 \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_{y_N}^2 \end{pmatrix} = \begin{pmatrix} 0.5^2 & 0 & \dots & 0 \\ 0 & 0.5^2 & \dots & 0 \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & 0.5^2 \end{pmatrix}$$

$$\Sigma_{\mathbf{p}_y} = \left(\mathrm{J}_{\mathbf{p}_y}{}^T \Sigma_{\mathbf{y}}^{-1} \mathrm{J}_{\mathbf{p}_y}\right)^{-1} = 10^{-3} \cdot \begin{pmatrix} 0.301 & -0.017 \\ -0.017 & 2.501 \end{pmatrix}$$

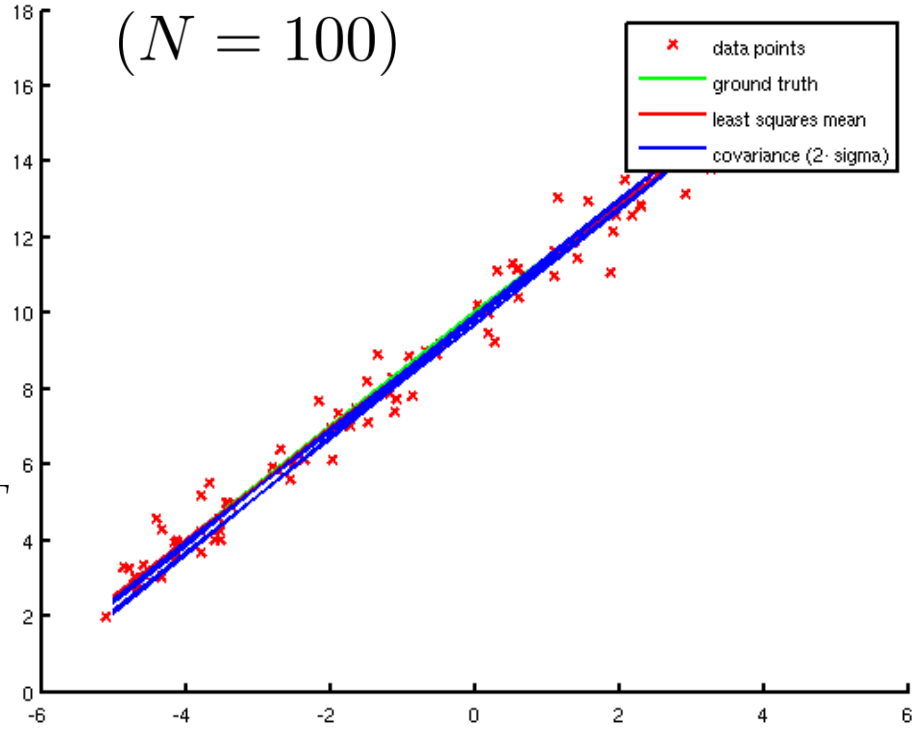
Example: line fit, direct parametrization

$$\frac{y-n}{m} = x$$

Jacobian w.r.t. input uncertainty:

$$\frac{\partial x}{\partial m} = \frac{n-y}{m^2}, \quad \frac{\partial x}{\partial n} = -\frac{1}{m}$$

$$\Rightarrow \mathrm{J}_{\mathbf{p}_x} = \begin{pmatrix} (n-y_1)/m^2 & \dots & (n-y_N)/m^2 \\ -1/m & \dots & -1/m \end{pmatrix}^T$$



$$\Sigma_{\mathbf{x}} = \begin{pmatrix} \sigma_{x_1}^2 & 0 & \dots & 0 \\ 0 & \sigma_{x_2}^2 & \dots & 0 \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_{x_N}^2 \end{pmatrix} = \begin{pmatrix} 0.25^2 & 0 & \dots & 0 \\ 0 & 0.25^2 & \dots & 0 \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & 0.25^2 \end{pmatrix}$$

$$\Sigma_{\mathbf{p}_x} = \left(\mathrm{J}_{\mathbf{p}_x}{}^T \Sigma_{\mathbf{x}}^{-1} \mathrm{J}_{\mathbf{p}_x}\right)^{-1} = 10^{-3} \cdot \begin{pmatrix} 0.116 & 0.005 \\ 0.005 & 1.376 \end{pmatrix}$$

Combining covariances from different sources

- If the sources are independent, the covariance matrices add
  - Line example – error due to incorrectly known x *and* noisy measurements:

$$\Sigma_{\mathbf{p}} = \Sigma_{\mathbf{p}_x} + \Sigma_{\mathbf{p}_y}$$

  - Line example: $\Sigma_{\mathbf{p}} = 10^{-3} \cdot \begin{pmatrix} 0.408 & -0.04 \\ -0.04 & 3.794 \end{pmatrix}$
    - Covariances are negligible

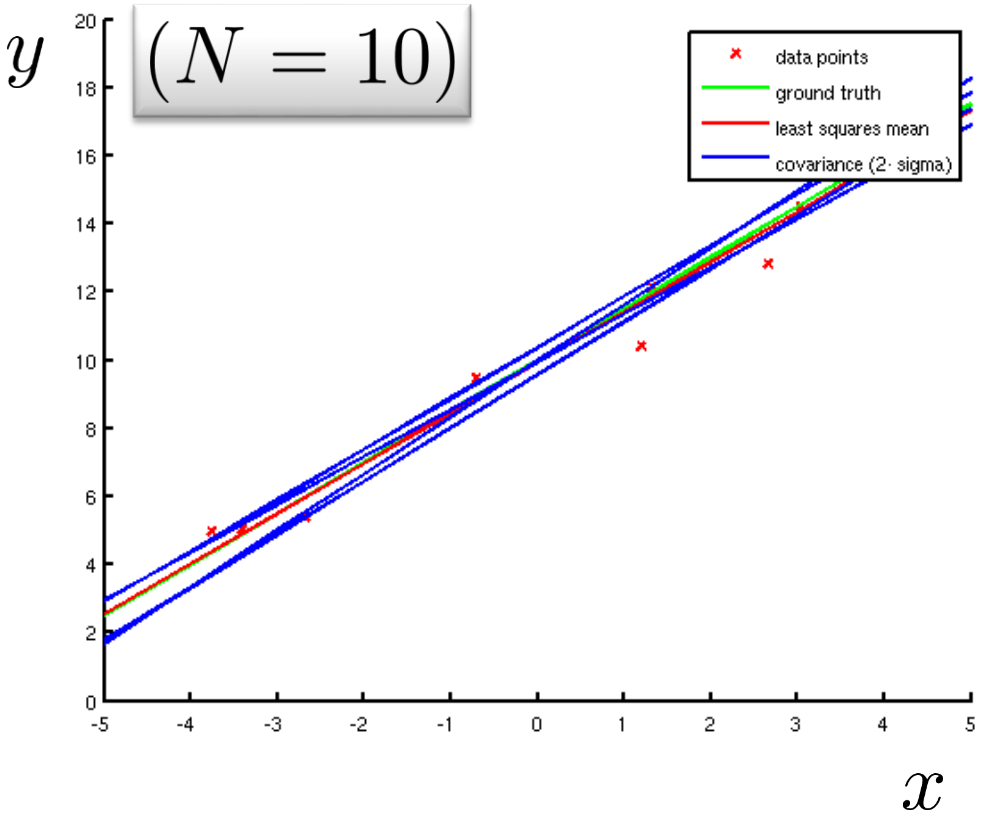
$$\sigma_m = \sqrt{\Sigma_{\mathbf{p}}^{1,1}}, \quad \sigma_n = \sqrt{\Sigma_{\mathbf{p}}^{2,2}}$$

$$\Rightarrow \mathbf{p} = (1.516 \pm 0.0408, 9.827 \pm 0.123)^T$$

with 95% certainty $(\mathbf{p} \pm (2 \cdot \sigma_m, 2 \cdot \sigma_n)^T)$

Less points – larger variance

$$\Sigma_{\mathbf{p}} = 10^{-2} \cdot \begin{pmatrix} 0.441 & -0.277 \\ -0.277 & 4.037 \end{pmatrix}$$



$$\Rightarrow \mathbf{p} = (1.516 \pm 0.133, 9.827 \pm 0.402)^T$$

with 95% certainty $\left(\mathbf{p} \pm (2 \cdot \sigma_m, 2 \cdot \sigma_n)^T\right)$

Monte-Carlo Estimation

- Idea: generate synthetic measurements around true measurement and analyze sensitivity of solution
  - Collect all parameter estimates from perturbed measurements and estimate their statistics
- Generating data:
  - Line example:
    - Perturbing the measurements $\mathbf{y}' = \mathbf{y} + \mathcal{N}_{\Sigma_\mathbf{y}}$
    - Perturbing the input positions $\mathbf{x}' = \mathbf{x} + \mathcal{N}_{\Sigma_\mathbf{x}}$

Monte-Carlo Estimation

- Generating noise according to some covariance matrix

$$\mathcal{N}_\Sigma = \mathcal{N}(0, \mathbf{1}) \cdot \mathrm{chol}(\Sigma)$$

Over-Parameterized Models

INSTITUT d'OPTIQUE GRADUATE SCHOOL

- In practice: (linear) estimates often over-parameterized
  - Example line equation: $a \cdot x + b \cdot y + c = 0$
  - Line is parameterized by $\mathbf{p} = (a, b, c)^T$

- Problem: $\lambda\mathbf{p} = (\lambda a, \lambda b, \lambda c)^T$

$\Rightarrow \lambda(a \cdot x + b \cdot y + c) = 0$ is the same equation

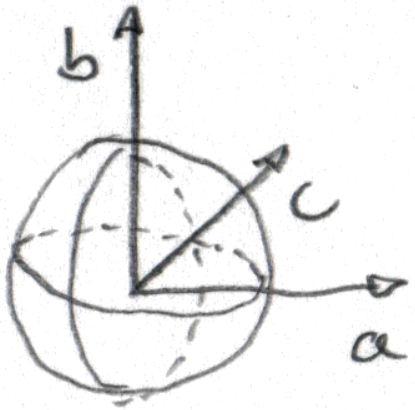
$\Rightarrow {\mathbf{J}_\mathbf{p}}^T \Sigma_\mathbf{x}^{-1} \mathbf{J}_\mathbf{p}$ is ill-conditioned (not full rank) and cannot be inverted

Over-Parameterized Models

- Idea: invert model in lower-dimensional space where it is full-rank and transfer coordinates

- Example: normalized coordinates: $||\mathbf{p}|| = 1$
  - Describes a manifold (the 2-sphere S^2)

    in 3D Euclidean parameter space

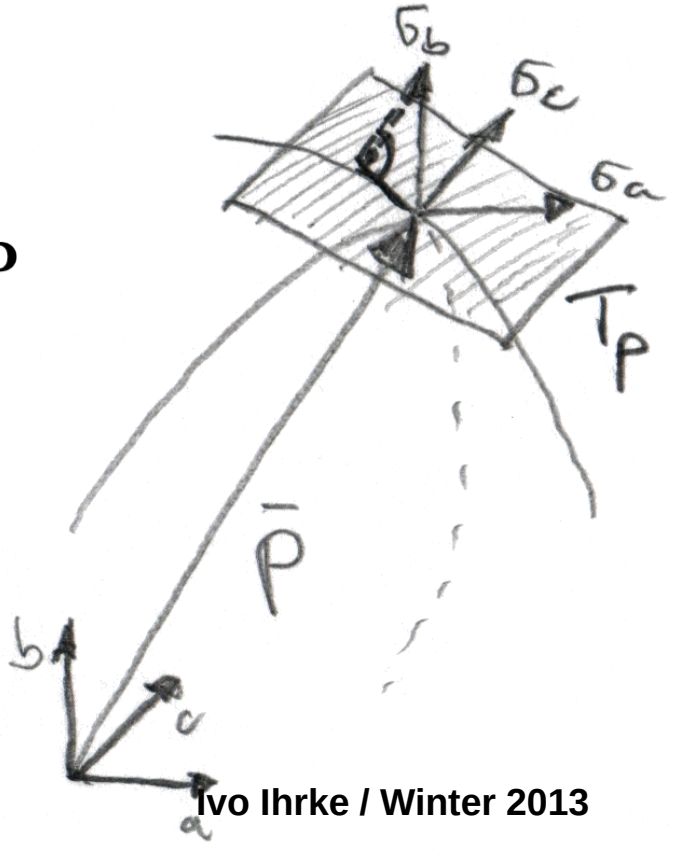


Over-Parameterized Models

- Parameter error may vary off the surface of allowed models

$$|| \, (a + \sigma_a, b + \sigma_b, c + \sigma_c)^T \, || \neq 1$$

- Solution:
  - Project onto tangent space $T_{\mathbf{p}}$
  - Correct to first order
  - Tangent space
    - can be computed by Jacobian of some direct parameterization



Over-Parameterized Models

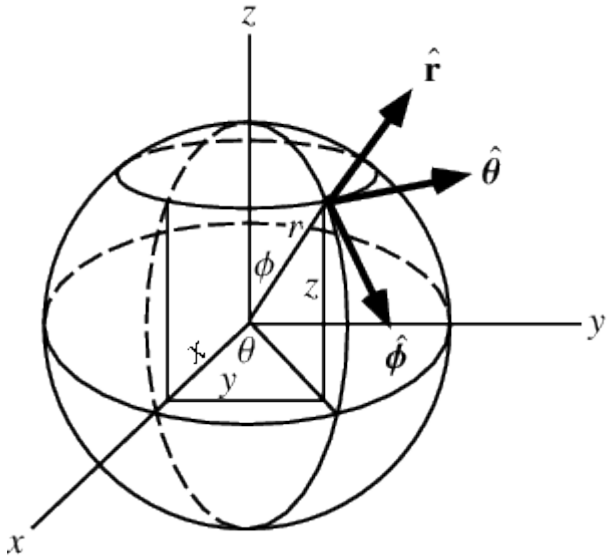
- Example: normalized coordinates
  - Describes a sphere of radius 1
  - Use spherical coordinates

$$
\begin{aligned}
x &= r \cdot \cos(\theta)\sin(\phi) \\
y &= r \cdot \sin(\theta)\sin(\phi) \\
z &= r \cdot \cos(\theta)
\end{aligned}
$$

  - Set $r = 1$

$$
\mathtt{J}_\theta = \begin{pmatrix} \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial \phi} \\ \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial \phi} \\ \frac{\partial z}{\partial \theta} & \frac{\partial z}{\partial \phi} \end{pmatrix}
$$

$$
\hat{\theta} \quad \hat{\phi}
$$



Over-Parameterized Models

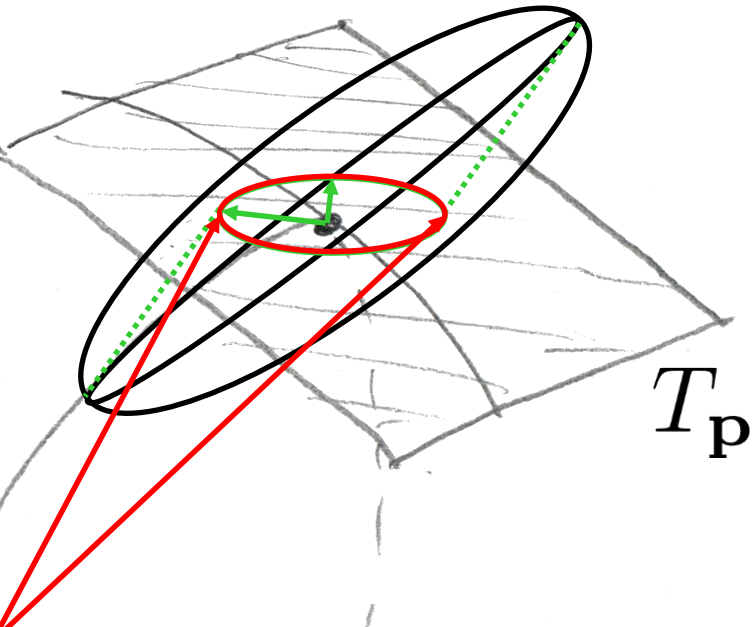
- Then, the covariance transfer can be computed by

$$\Sigma_{\mathbf{p}} = \mathrm{J}_\theta \left(\mathrm{J}_\theta{}^T \, \mathrm{J}_{\mathbf{p}}{}^T \Sigma_{\mathbf{x}}^{-1} \, \mathrm{J}_{\mathbf{p}} \mathrm{J}_\theta \right)^{-1} \mathrm{J}_\theta{}^T$$

1. Error ellipsoid in over-parameterized space

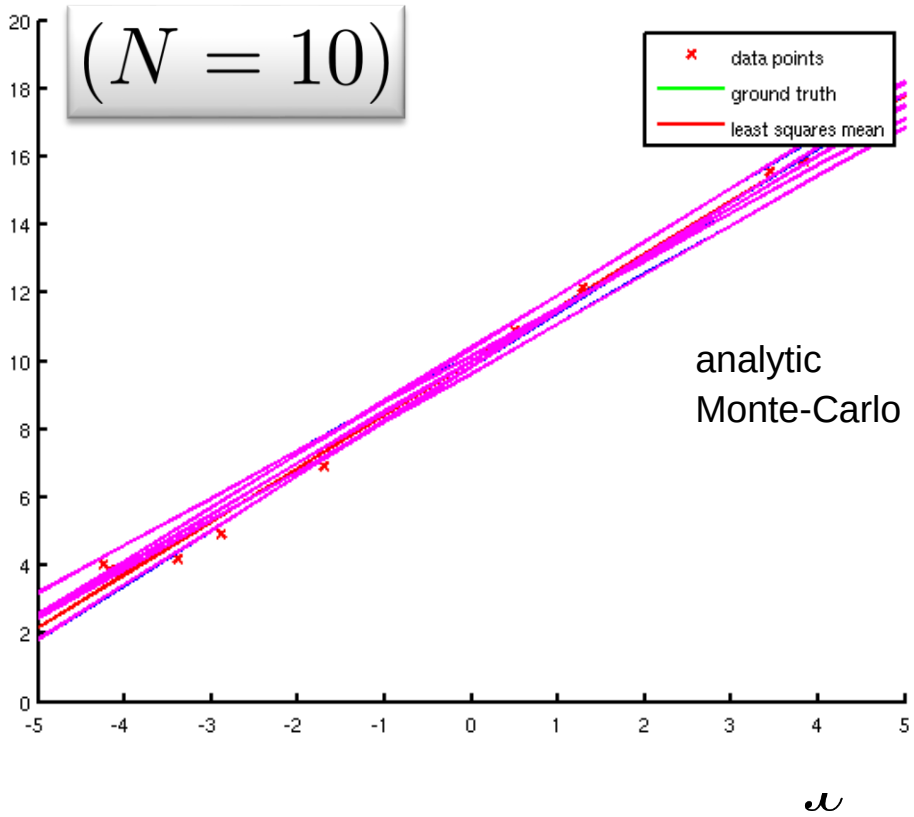
2. Error ellipsoid in tangent space (tangent space coordinates)

3. Error ellipsoid in tangent space (over-parameterized space coordinates)



Over-Parameterized Models

- Example: line fit, implicit representation, same data as before
- For comparison, make sure to use same constraint

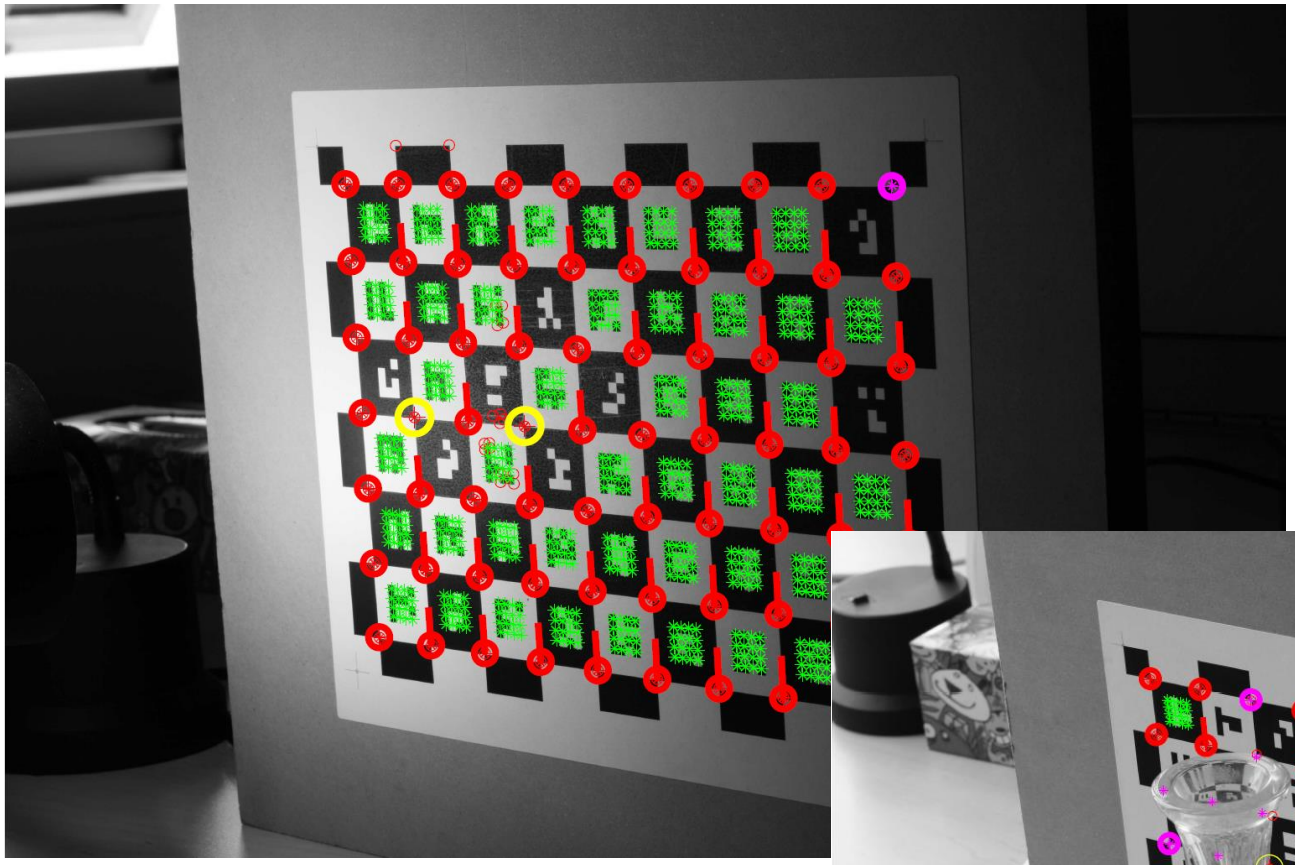


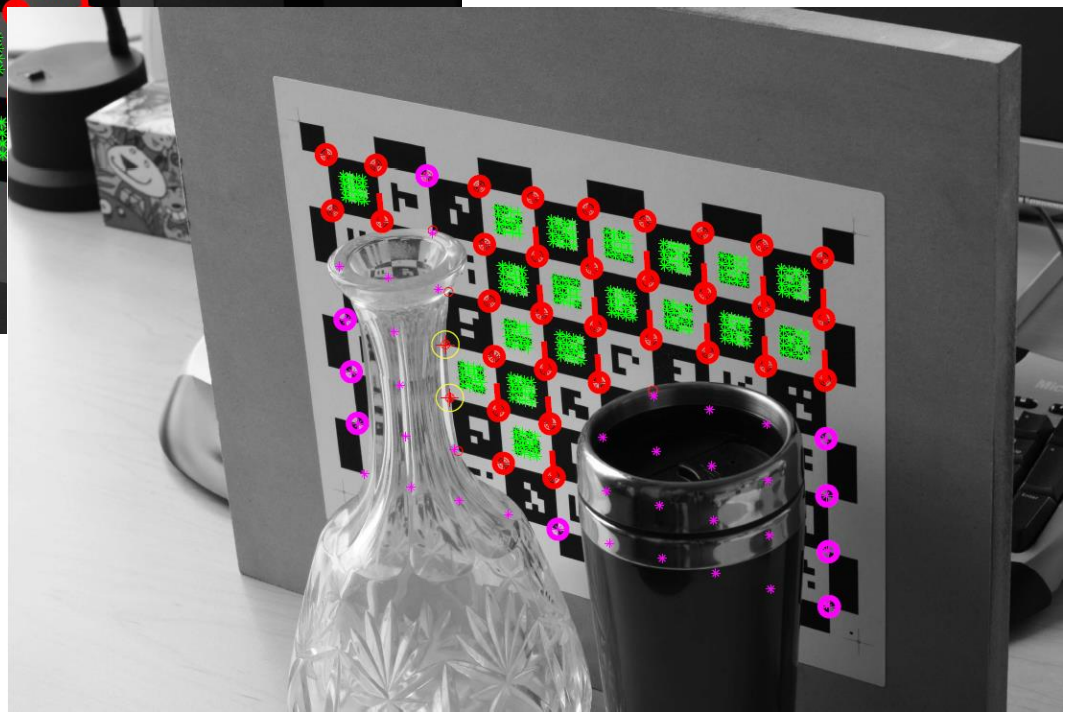
Summary – Uncertainty Estimation

INSTITUT d'OPTIQUE GRADUATE SCHOOL

- Monte-Carlo
  - \+ Very general
  - \+ Easy to implement
  - \+ Variable dependency and over-parameterization simple
  - \- Slow
- Analytical
  - \+ Fast
  - \- Sometimes difficult derivation
  - \- Unflexible (e.g. matrix decompositions are tricky to handle)
  - \- Explicit handling of over-parameterization and variable dependencies

Automatic Checkerboards





[Atcheson et al. “CALTag”]

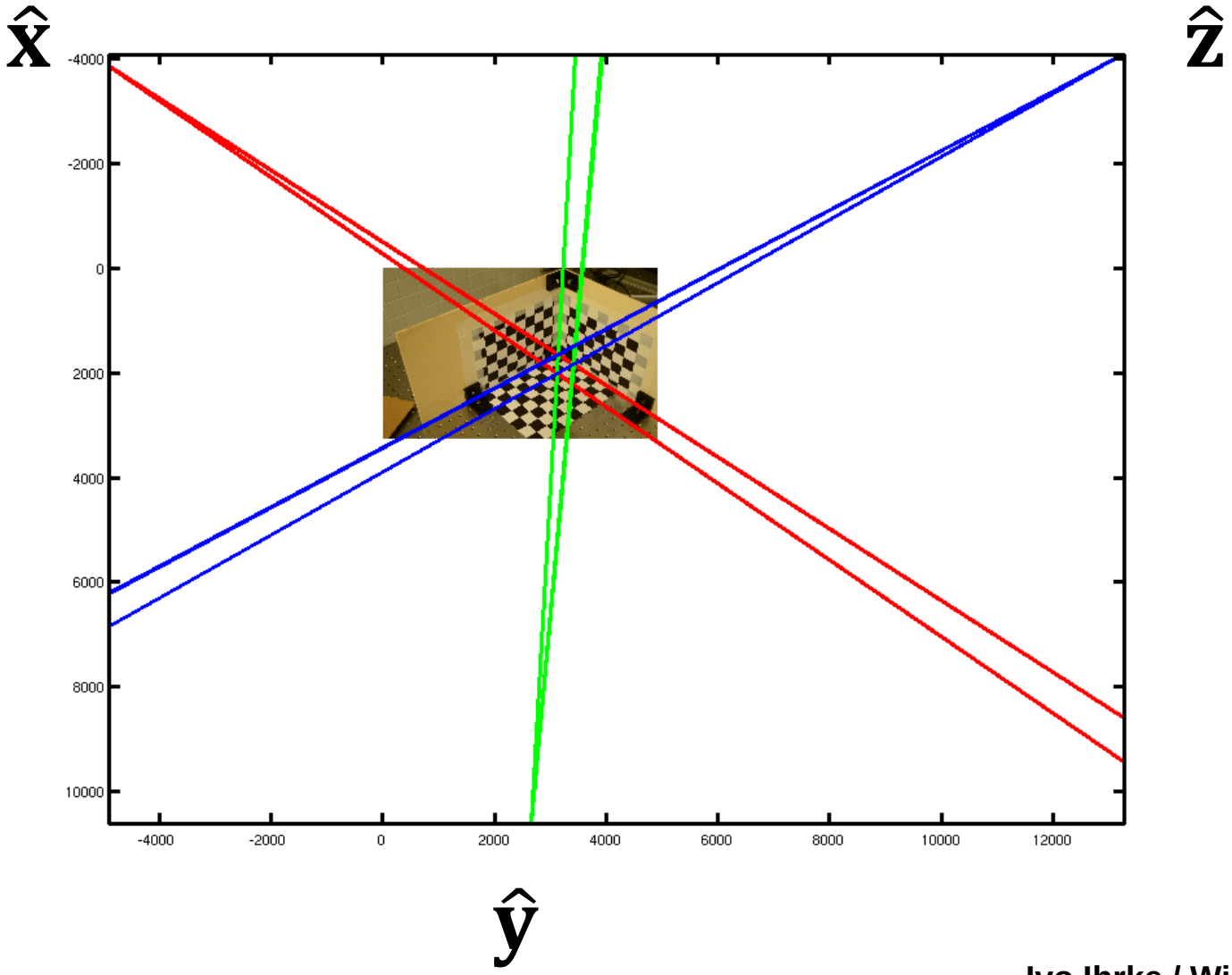
Automatic Checkerboards

- Principal steps:
  - Detected linked edge segments
  - Fit lines
  - Intersect lines to obtain corners
  - Fit homography to each patch
  - Sample Code
  - Classify patch
  - Optional: if planar, fit homography to all patches and retry to find additional patches
- In all steps: outlier filtering

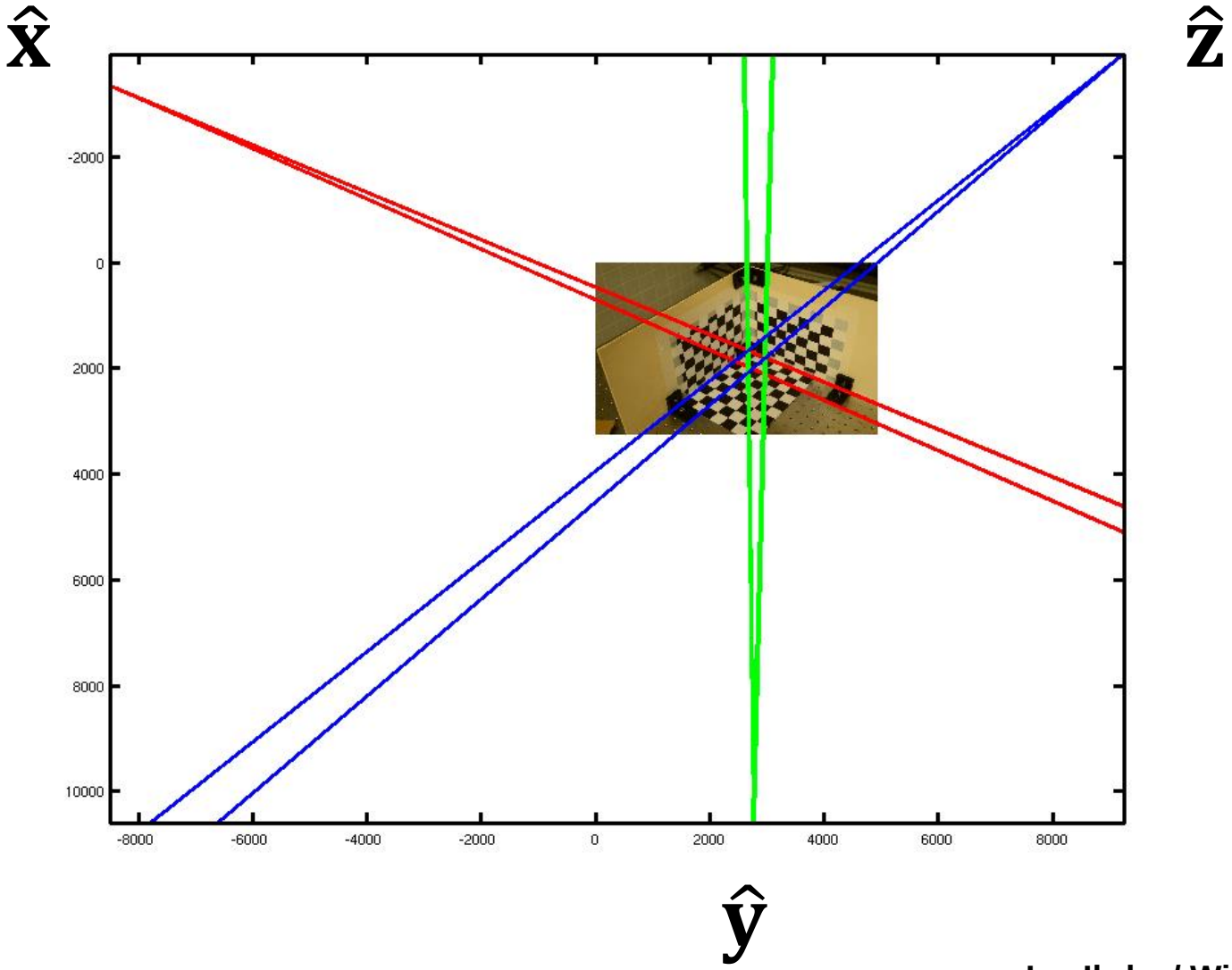
What about single-view “natural” scenes?

- World-parallel lines are distorted
- → vanishing points

Vanishing points



Vanishing points



Vanishing points

- Vanishing points are projections of the directions of the coordinate axes

$$\hat{\mathbf{x}} = \mathtt{A}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} \qquad \hat{\mathbf{y}} = \mathtt{A}\begin{pmatrix}0\\1\\0\\0\end{pmatrix} \qquad \hat{\mathbf{z}} = \mathtt{A}\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

$$\mathtt{A} = \mathtt{KR}\left[\mathbb{1}\middle|-\mathbf{C}\right]$$

$$\Rightarrow \mathtt{V} := [\hat{\mathbf{x}}\hat{\mathbf{y}}\hat{\mathbf{z}}] = \mathtt{A}\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\\0&0&0\end{pmatrix} = \mathtt{KR}$$

Points at infinity = intersection of parallel lines

Vanishing points

$$\Rightarrow \mathtt{V} := [\hat{\mathbf{x}}\hat{\mathbf{y}}\hat{\mathbf{z}}] \quad = \quad \mathtt{A} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \mathtt{KR}$$

- → can decompose V with RQ decomposition
- Fix origin of coordinate system $\hat{\mathbf{o}} \quad = \quad \mathtt{A} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

$$\Rightarrow \hat{\mathbf{o}} \quad = \quad -\mathtt{KR}\mathbf{C}$$

$$-\left(\mathtt{KR}\right)^{-1} \hat{\mathbf{o}} = \mathbf{C}$$

Accuracy considerations

- Extract focal length from intrinsic matrix K

- View 1: f = 34.10mm (sigmas ?)
  - EXIF info: 34.0 mm

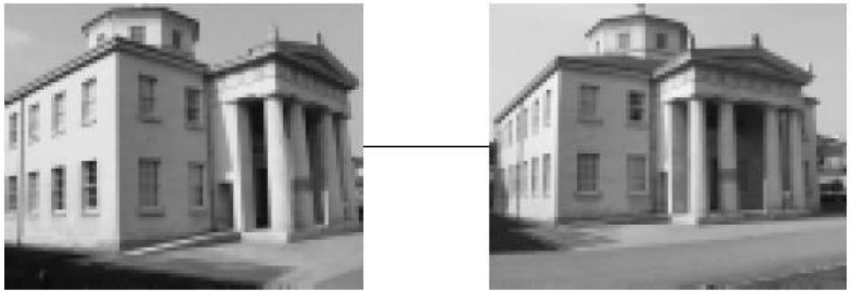
$$\mathrm{K}_1 = \begin{pmatrix} 7400.47 & -829.20 & 3678.15 \\ 0 & 6870.23 & 888.23 \\ 0 & 0 & 1.0 \end{pmatrix}$$

- View 2: f = 33.45mm (sigmas ?)
  - EXIF info: 34.0 mm

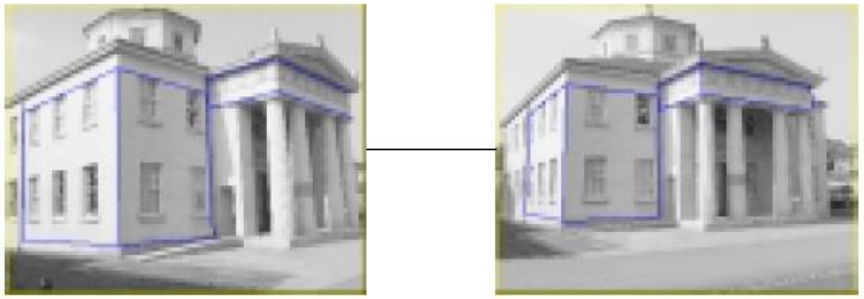
$$\mathrm{K}_2 = \begin{pmatrix} 7279.19 & 862.23 & 1169.35 \\ 0 & 6712.14 & 1104.74 \\ 0 & 0 & 1.0 \end{pmatrix}$$

3D Reconstruction of Architecture

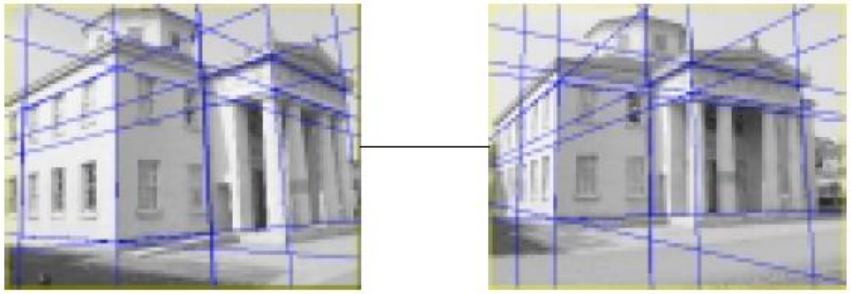
1. Original uncalibrated photographs



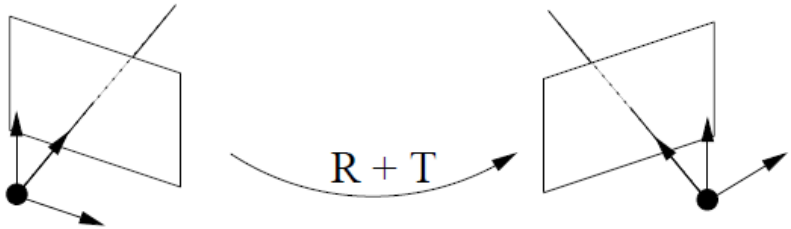
2. Primitive definition and localisation



3. Finding vanishing points and camera calibration



4. Computation of projection matrices



5. Triangulation, 3D reconstruction and texture mapping

[Cipolla'99]

Towards Multiple Views and Self-Calibration

Ivo Ihrke / Winter 2013

Foundations of Camera Motion Estimation

INSTITUT d'OPTIQUE GRADUATE SCHOOL

- Feature detection
- Feature tracking
- 2D homography
- Fundamental matrix

Why bother?

Quelle: Transformers, DreamWorks, Director: Michael Bay, Special effects by Industrial Light & Magic and Digital Domain, USA

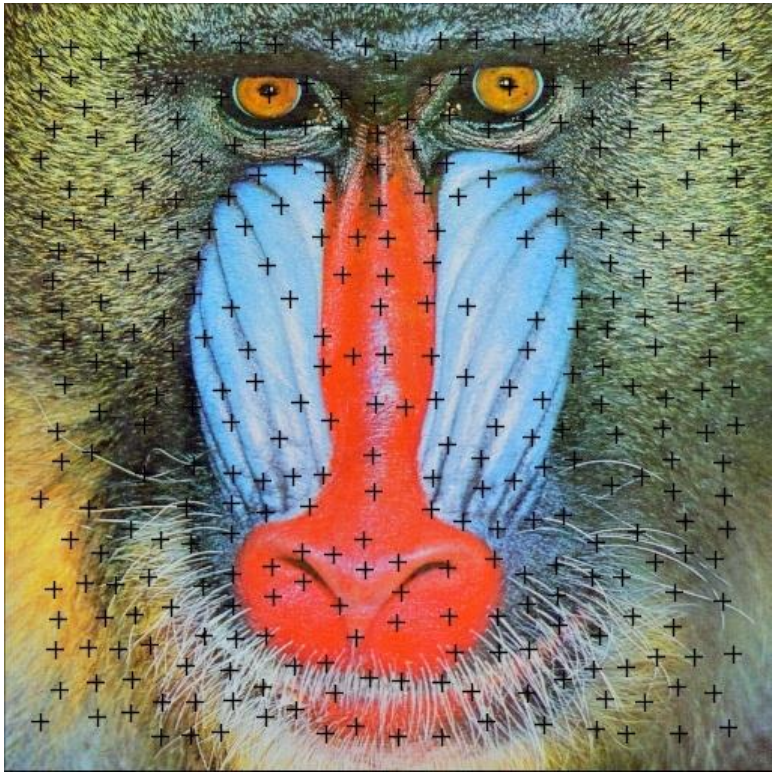
Towards Multiple Views and Self-Calibration

-- Feature Detection, Matching and Tracking --

Ivo Ihrke / Winter 2013

Feature Point Detection

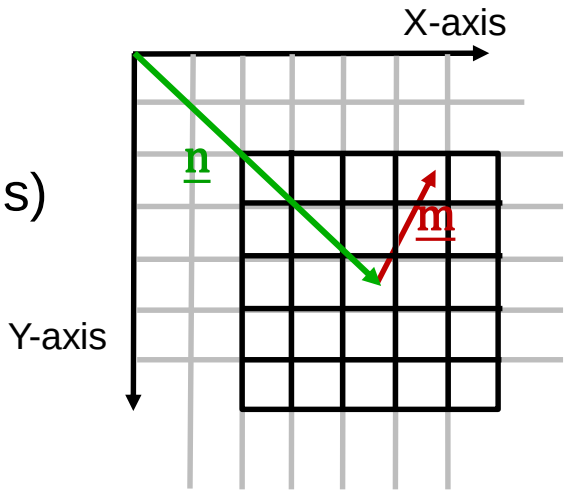
- Harris Corner Detector
- corners at high image signal gradients in two perpendicular directions



Harris Corner Detector

image point $\underline{\mathbf{n}} = (n_x, n_y)^\top$

window point $\underline{\mathbf{m}} = (m_x, m_y)^\top$ (e.g. 5 x 5 pixels)



- calculate

$$\mathtt{G}(\underline{\mathbf{n}}) = \sum_{m_x}\sum_{m_y} \begin{bmatrix} g_x^2(\underline{\mathbf{n}}+\underline{\mathbf{m}}) & g_x(\underline{\mathbf{n}}+\underline{\mathbf{m}})\, g_y(\underline{\mathbf{n}}+\underline{\mathbf{m}}) \\ g_x(\underline{\mathbf{n}}+\underline{\mathbf{m}})\, g_y(\underline{\mathbf{n}}+\underline{\mathbf{m}}) & g_y^2(\underline{\mathbf{n}}+\underline{\mathbf{m}}) \end{bmatrix}$$

$$\frac{\partial I(x,y)}{\partial x} \approx g_x = \frac{I(x+1,y) - I(x-1,y)}{2}$$

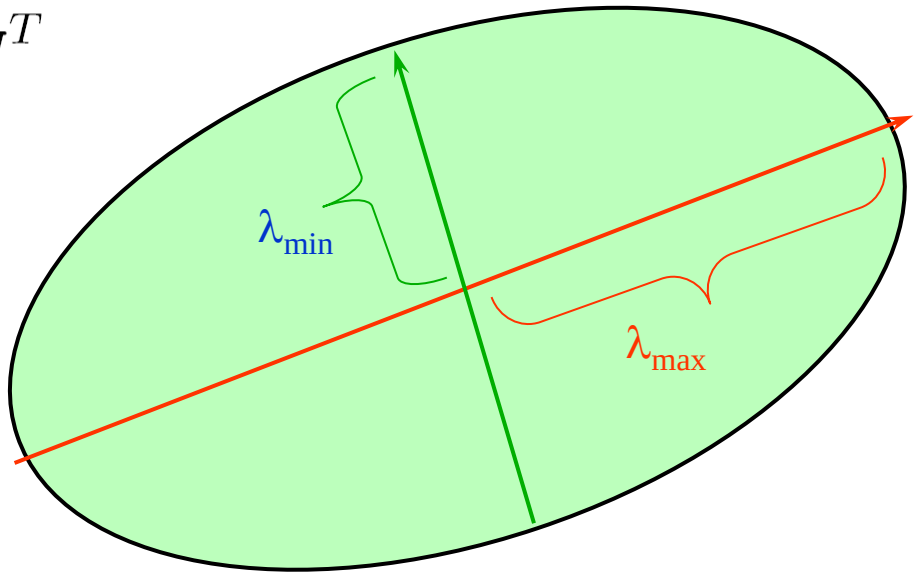
$$\frac{\partial I(x,y)}{\partial y} \approx g_y = \frac{I(x,y+1) - I(x,y-1)}{2}$$

Harris Corner Detector

Eigenvalues of the symmetric matrix $\mathtt{G}(\underline{\mathbf{n}})$:

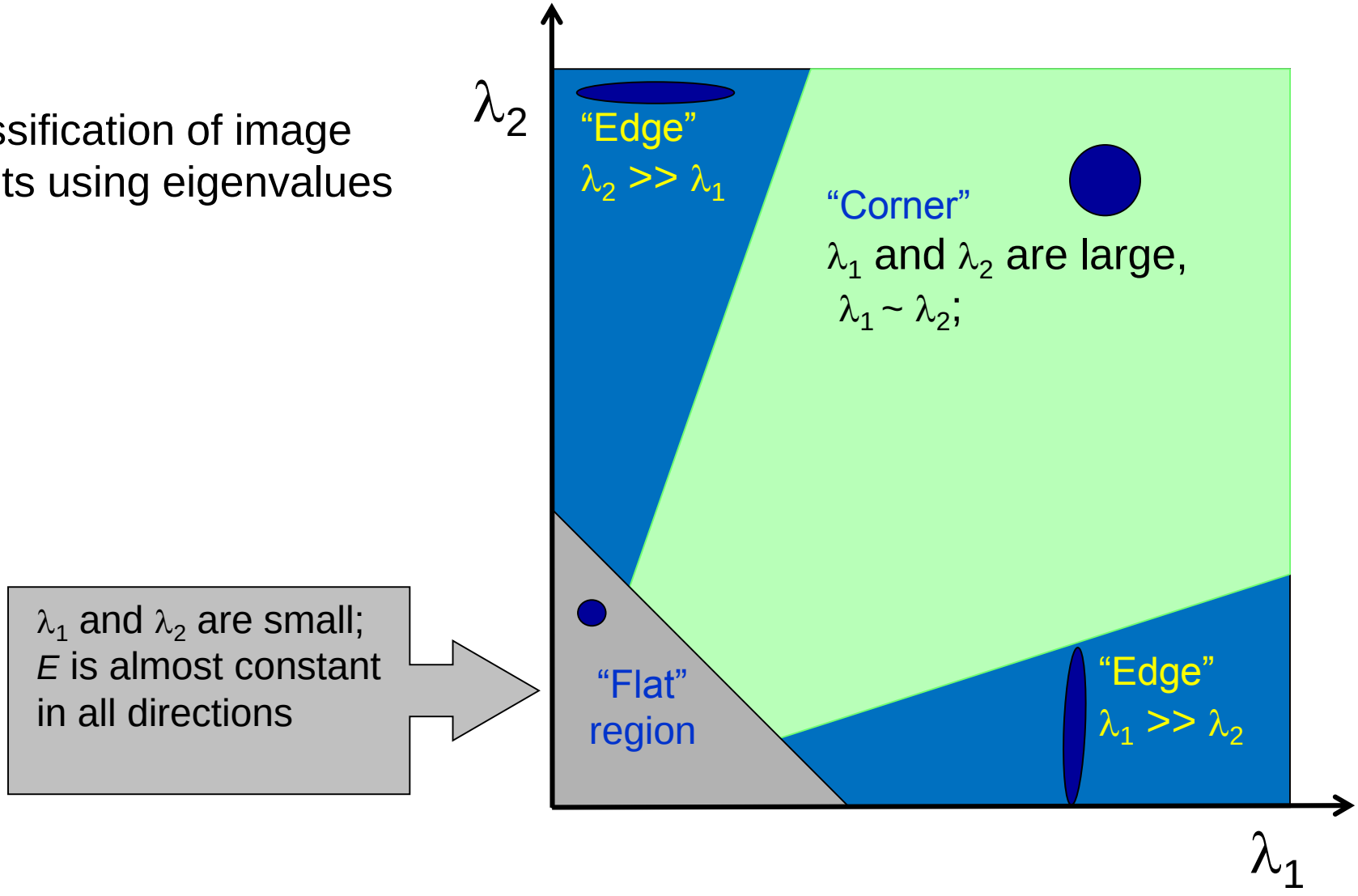
$$\mathtt{G}\,\mathbf{v}_i = \lambda_i\,\mathbf{v}_i$$

$$\mathtt{G}(\underline{\mathbf{n}}) = \mathtt{V}\mathtt{W}\mathtt{V}^T = \mathtt{V}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}\mathtt{V}^T$$



Harris Corner Detector

Classification of image points using eigenvalues



Harris Corner Detector

- Harris Cornerness Response Function

$$\mathrm{CRF}(\underline{\mathbf{n}}) = \det(\mathtt{G}(\underline{\mathbf{n}})) + K_{\mathrm{CRF}}\ \mathrm{trace}^2(\mathtt{G}(\underline{\mathbf{n}}))$$

- Determinant of a matrix

$$\det(\mathtt{G}) = \lambda_1 \cdot \lambda_2$$

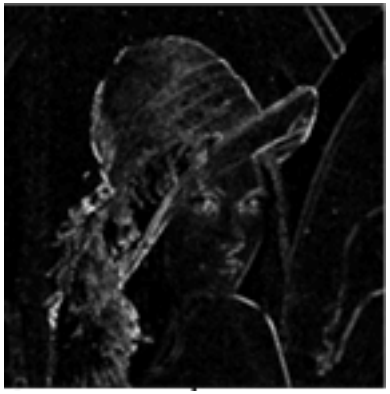
- Trace of a matrix

$$\mathrm{trace}(\mathtt{G}) = \lambda_1 + \lambda_2$$

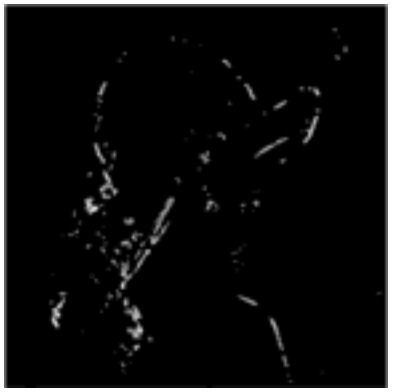
Harris Corner Detector

Input image

Cornerness response



Threshold



Non-maxima suppression

Correspondence Analysis – KLT Tracker

Correspondence Analysis

- For video KLT-Tracker (Kanade, Lucas and Tomasi)
- Minimization of the SSD between two windows in subsequent camera images to find the displacement vector

$$\underline{\mathbf{d}} = (\underline{\mathbf{n}}_K - \underline{\mathbf{n}}_{K-1})$$

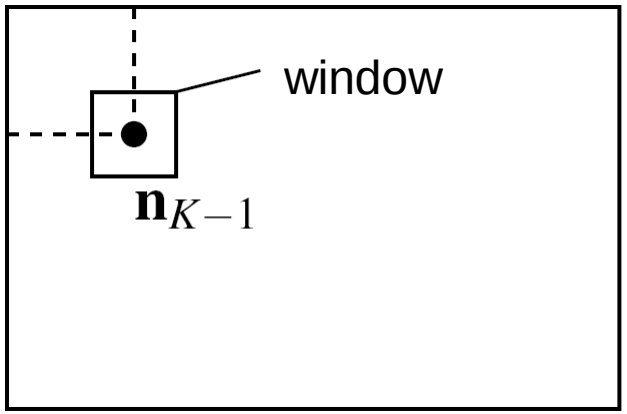


image K-1

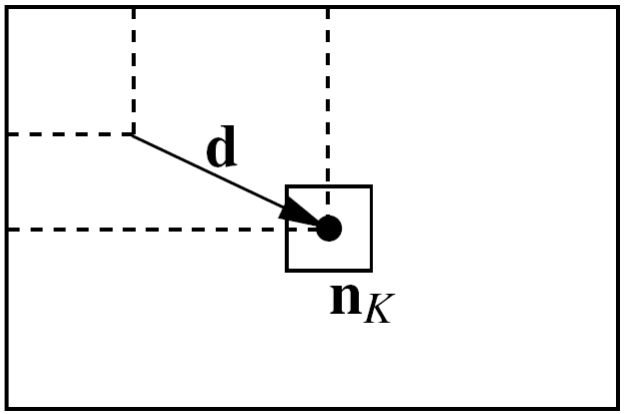


image K

$$\mathrm{SSD}(\underline{\mathbf{d}}) = \sum_{m_x} \sum_{m_y} [I_K(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{d}} + \underline{\mathbf{m}}) - I_{K-1}(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}})]^2 \rightarrow \min$$

KLT Tracker

$$\mathrm{SSD}(\underline{\mathbf{d}}) = \sum_{m_x} \sum_{m_y} \left[I_K(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{d}} + \underline{\mathbf{m}}) - I_{K-1}(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \right]^2 \rightarrow \min$$

- Linearization of the image signal with a Taylor series

$$I_K(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{d}} + \underline{\mathbf{m}}) \approx I_K(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) + \begin{pmatrix} g_x(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \\ g_y(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \end{pmatrix}^\top \underline{\mathbf{d}}$$

- Finding the minimum by setting the derivative to zero

$$\sum_{m_x} \sum_{m_y} 2 \left[I_K(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) + \begin{pmatrix} g_x(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \\ g_y(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \end{pmatrix}^\top \underline{\mathbf{d}} - I_{K-1}(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \right] \begin{pmatrix} g_x(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \\ g_y(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \end{pmatrix} = \mathbf{0}$$

KLT Tracker

$$\sum_{m_x}\sum_{m_y} 2\left[I_K(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}}) + \begin{pmatrix} g_x(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}}) \\ g_y(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}}) \end{pmatrix}^{\top} \underline{\mathbf{d}} - I_{K-1}(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}})\right] \begin{pmatrix} g_x(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}}) \\ g_y(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}}) \end{pmatrix} = \mathbf{0}$$

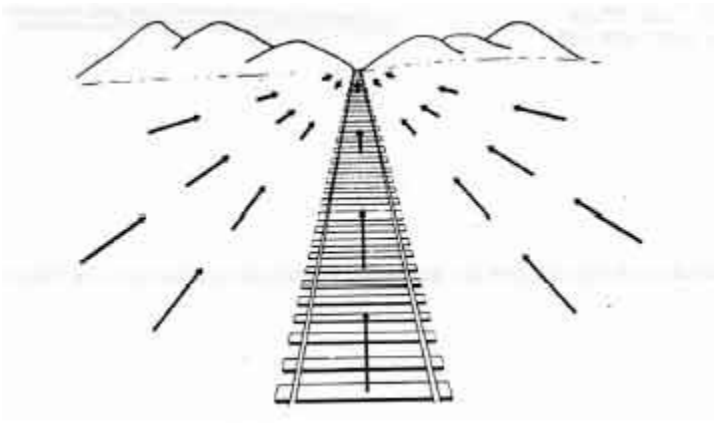
$$\underline{\mathbf{d}} = \mathtt{G}(\underline{\mathbf{n}}_{K-1})^{-1}\,\mathbf{b}(\underline{\mathbf{n}}_{K-1})$$

with

$$\mathbf{b}(\underline{\mathbf{n}}_{K-1}) = \sum_{m_x}\sum_{m_y} [I_{K-1}(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}}) - I_K(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}})] \begin{pmatrix} g_x(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}}) \\ g_y(\underline{\mathbf{n}}_{K-1}+\underline{\mathbf{m}}) \end{pmatrix}$$

Optical Flow – Horn & Schunck'81

- Continuous version of tracking: optical flow
- Apparent motion of brightness patterns in an image sequence (typically two frames)
- For images: $\vec{u}(\vec{x}): R^2 \rightarrow R^2$, is a vector valued fct.
- Often visualized as vector field or color coded

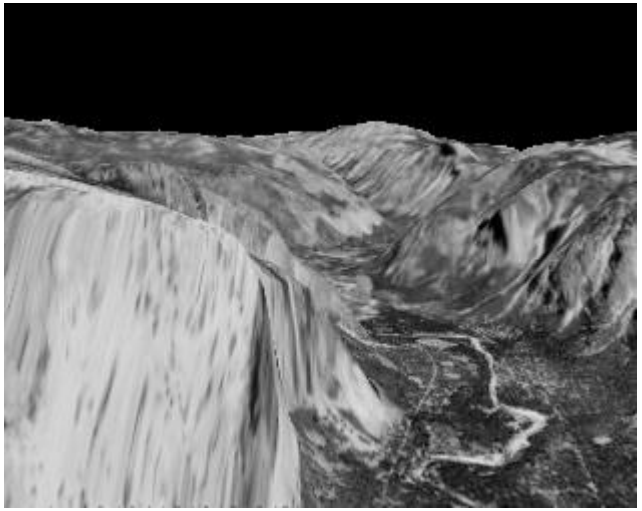


Optical Flow as seen by a person at the back of a train

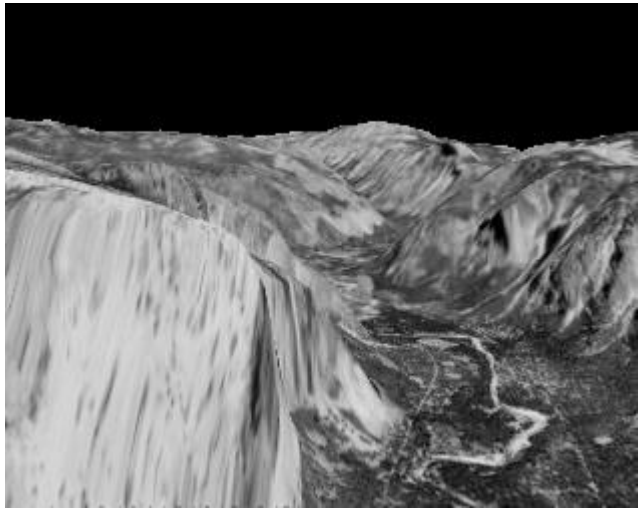
Example Yosemite sequence

INSTITUT d'OPTIQUE GRADUATE SCHOOL

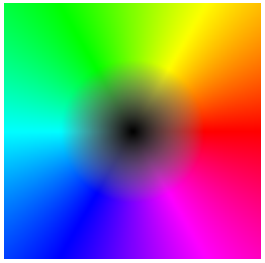
left



right

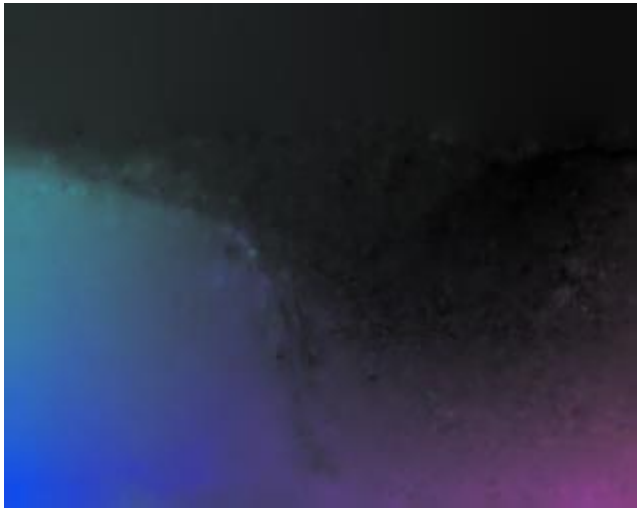


middlebury coding



IPOL coding

Flow field (middlebury coding)



Flow field (IPOL coding)

Optical Flow- Derivation

INSTITUT d'OPTIQUE GRADUATE SCHOOL

- Assume a video $I(\vec{x}, t): R^3 \rightarrow R$
- Brightness constancy implies

$$I(\vec{x} + \vec{u}(\vec{x}, t), t) = I(\vec{x}, t + 1)$$

- Look at one particular time step with flow vectors $\vec{u} = (u_x, u_y)$
  - perform Taylor expansion of $I(\vec{x}, t + 1)$:

$$I(\vec{x}, t + 1) \approx I(\vec{x}, t) + \frac{\partial I}{\partial x} u_x + \frac{\partial I}{\partial y} u_y + \frac{\partial I}{\partial t} + O(\nabla^2)$$

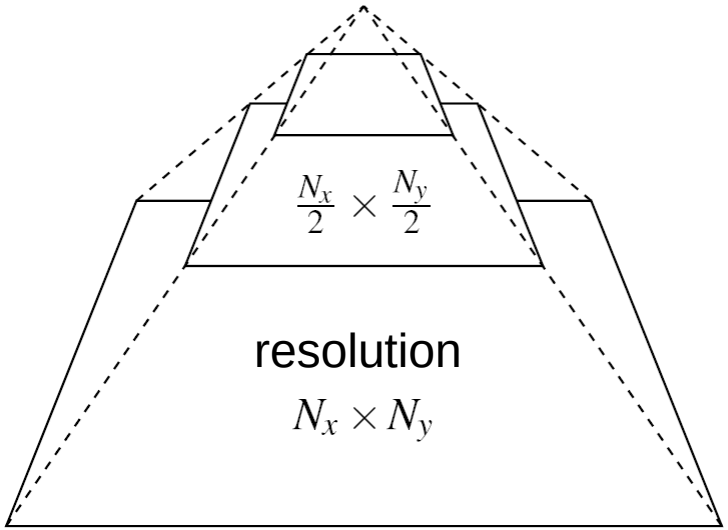
  - implies

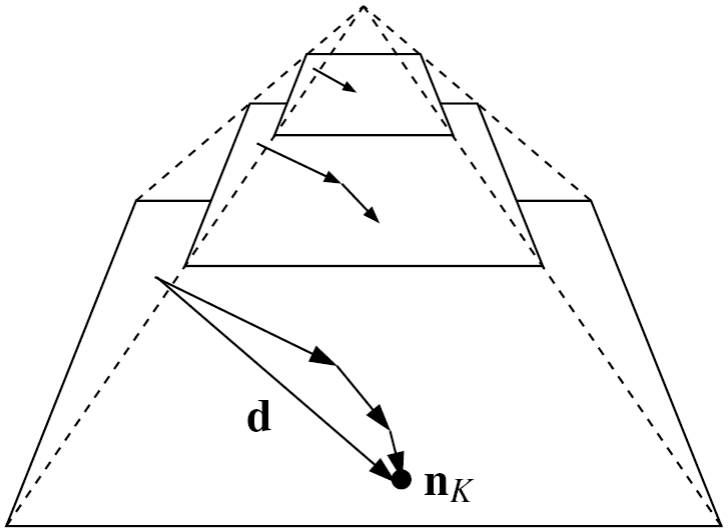
$$\frac{\partial I}{\partial x} u_x + \frac{\partial I}{\partial y} u_y + \frac{\partial I}{\partial t} = 0$$

  - Alternative form: $\nabla I \cdot \vec{u} + \frac{\partial I}{\partial t} = 0$

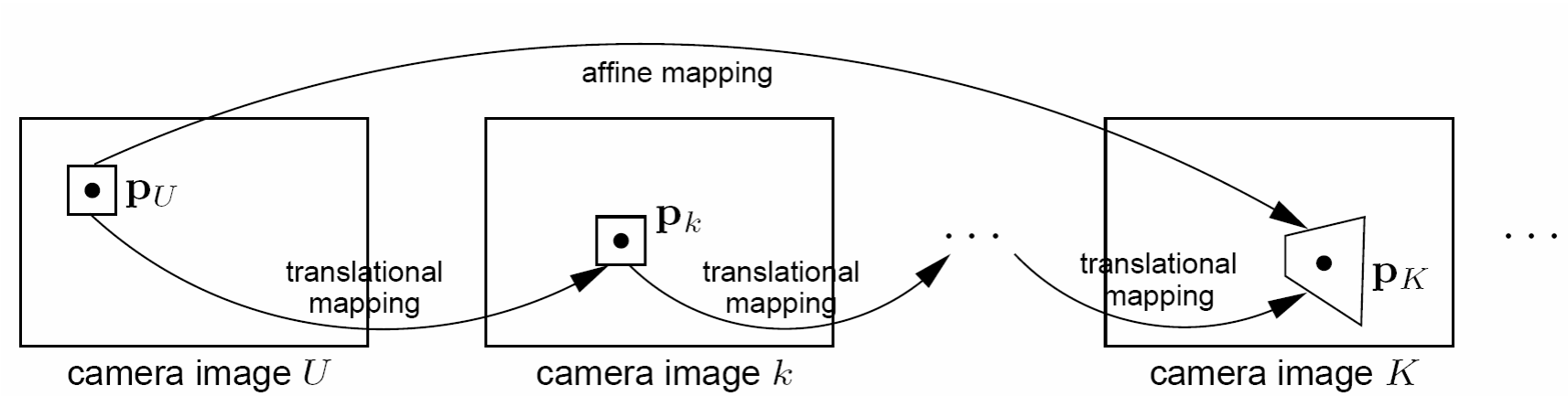
KLT-Tracker: Resolution pyramids

- KLT and optical flow descriptions are differential in nature → only suitable for small displacements (up to 2 pixels)
- Large displacements are small on a coarser scale
  - compute displacements on coarse scale
  - Upsample
  - Iteratively compute residuals on finer scales





KLT-Tracker: Affine mapping for longer sequences



KLT-Tracker summary

- For video (small displacements)
- Minimization of the SSD between two windows in subsequent camera images
- Linearization of the image signal with a Taylor series → linear equation system
- Linearization only valid for small displacements → Resolution Pyramids

References

1. Multiple View Geometry in Computer Vision, Richard Hartley and Andrew Zisserman, 2nd Edition, 2003
2. Cipolla, Drummond, Robertson, “Camera Calibration from Vanishing Points in Images of Architectural Scenes”, BMVC 1999
3. Zhengyou Zhang, “A Flexible New Technique for Camera Calibration”, PAMI, 2000
   implemented in Toolbox:
   Matlab Calibration Toolbox, Jean-Yves Bouguet 2000
   http://www.vision.caltech.edu/bouguetj/calib_doc/
4. Atcheson, Heide, Heidrich, “CALTag: High Precision Fiducial Markers for Camera Calibration.”, VMV2010
   implemented in Toolbox:
   http://www.cs.ubc.ca/nest/imager/tr/2010/Atcheson_VMV2010_CALTag/
5. Thorsten Thormählen “Zuverlässige Schätzung der Kamerabewegung aus einer Bildfolge“. PhD thesis, University of Hannover, 2006
   Implemented in: Voodoo Camera Tracker,
   http://www.digilab.uni-hannover.de.

References

6. Jean-Yves Bouguet “Pyramidal implementation of the affine Lucas-Kanade feature tracker”, 2001
7. Horn, Schunck “Determining optical flow“, Artifical Intelligence, 1981
8. Zhang, “Determining the epipolar geometry and its uncertainty”, 1996