Computational Optical Imaging - Optique Numerique

-- Multiple View Geometry and Stereo --

Winter 2013

Ivo Ihrke

with slides by Thorsten Thormaehlen

Feature Detection and Matching

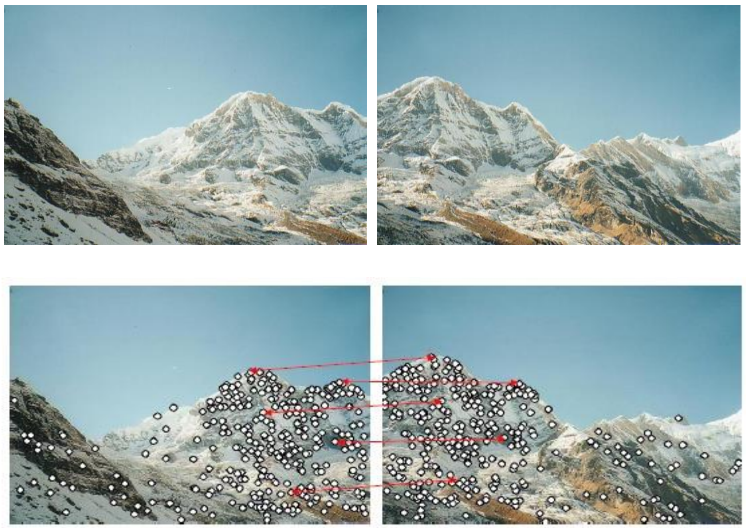
Wide-Baseline-Matching

Wide-baseline Matching with SIFT

- SIFT = Scale Invariant Feature Transform
  - David G. Lowe: “Distinctive image features from scale-invariant keypoints” (IJCV 2004)
  - http://www.cs.ubc.ca/~lowe/keypoints/
- Suited for wide-baseline matching
- Invariance to changes in illumination, scale, and rotation
- No gradient approach, but compares feature description of all candidates
- Many applications

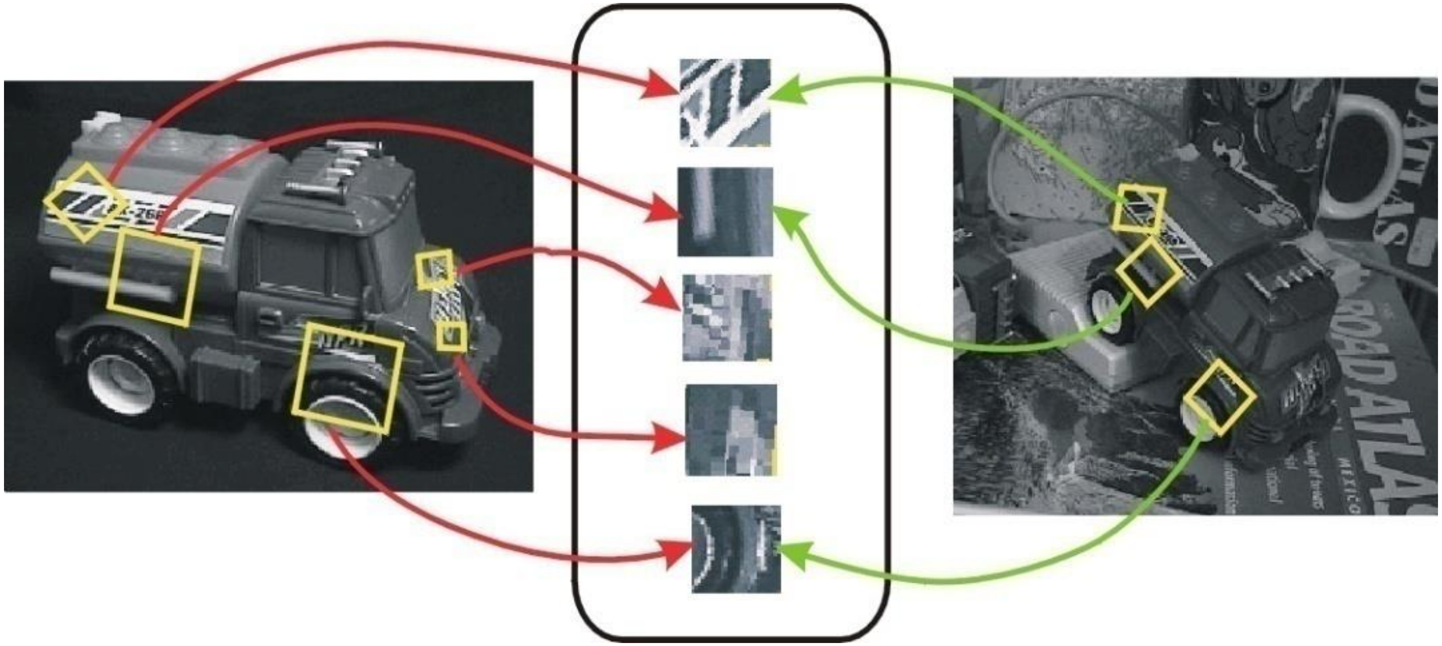
Wide-baseline Matching with SIFT

- Application example: Mosaic generation



Wide-baseline Matching with SIFT

- Application example: Object recognition

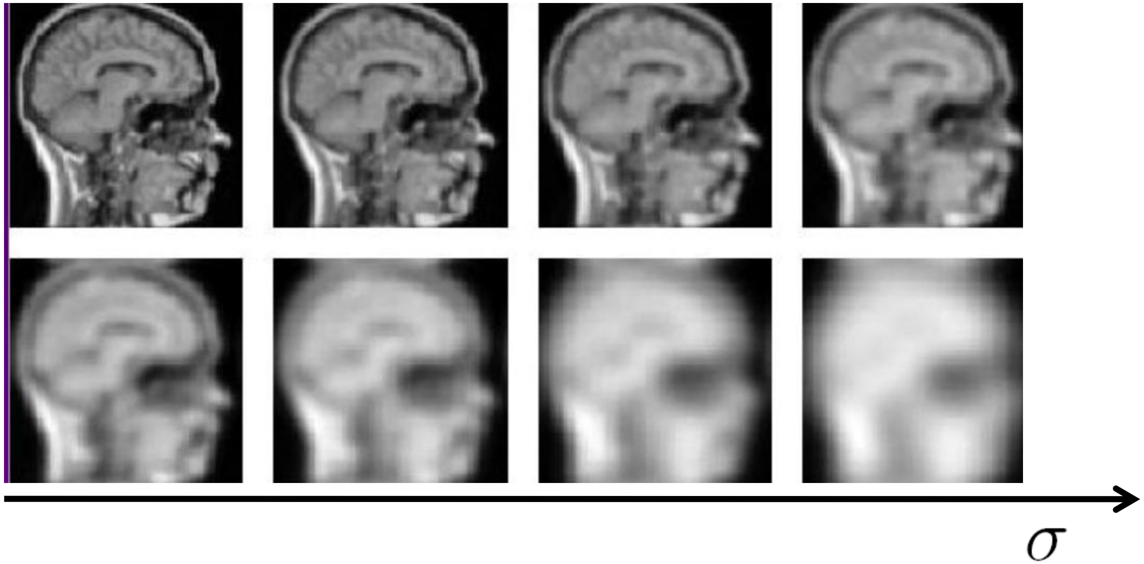


Wide-baseline Matching with SIFT

- Algorithm steps
  - Scale-space extrema detection
  - Keypoint localization
  - Orientation assignment
  - Keypoint descriptor
  - Descriptor matching

SIFT: Scale-space extrema detection

- Need to find “characteristic scale” for features
- Scale space representation

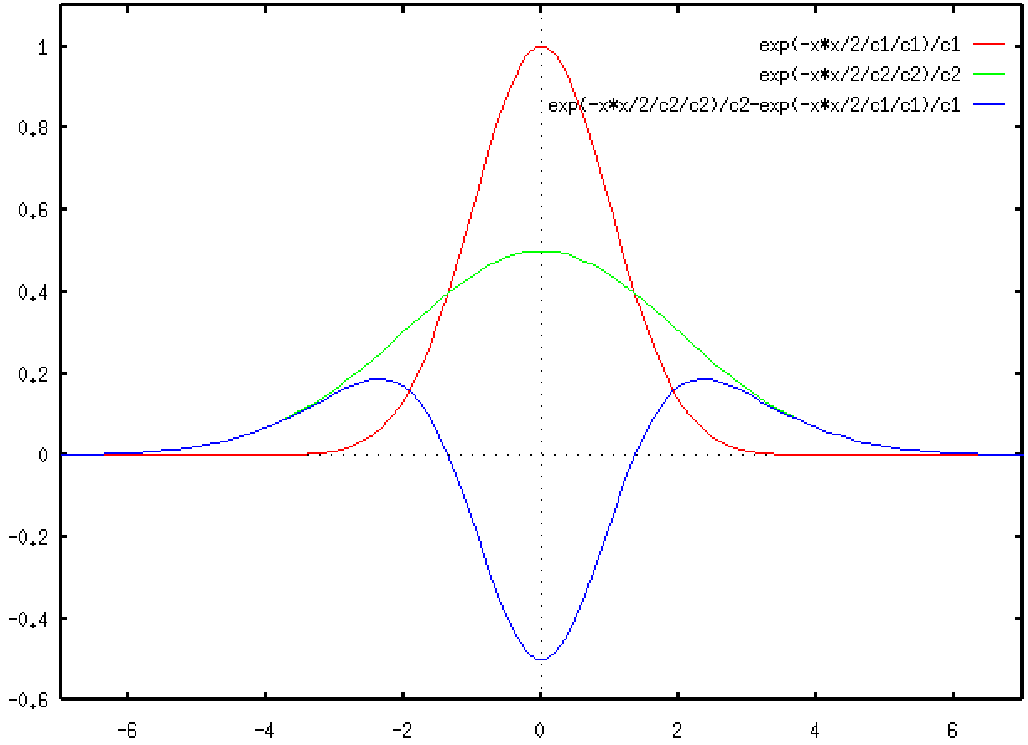


$$L(\underline{\mathbf{n}}, \sigma) = G(\underline{\mathbf{n}}, \sigma) * I(\underline{\mathbf{n}}) \qquad G(\underline{\mathbf{n}}, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{n_x^2 + n_y^2}{2\sigma^2}}$$

SIFT: Scale-space extrema detection

- Difference of Gaussian

$$\mathrm{DoG}(\underline{\mathbf{n}}) = (G(\underline{\mathbf{n}}, \sigma) - G(\underline{\mathbf{n}}, k\,\sigma)) * I(\underline{\mathbf{n}}) = L(\underline{\mathbf{n}}, \sigma) - L(\underline{\mathbf{n}}, k\,\sigma)$$

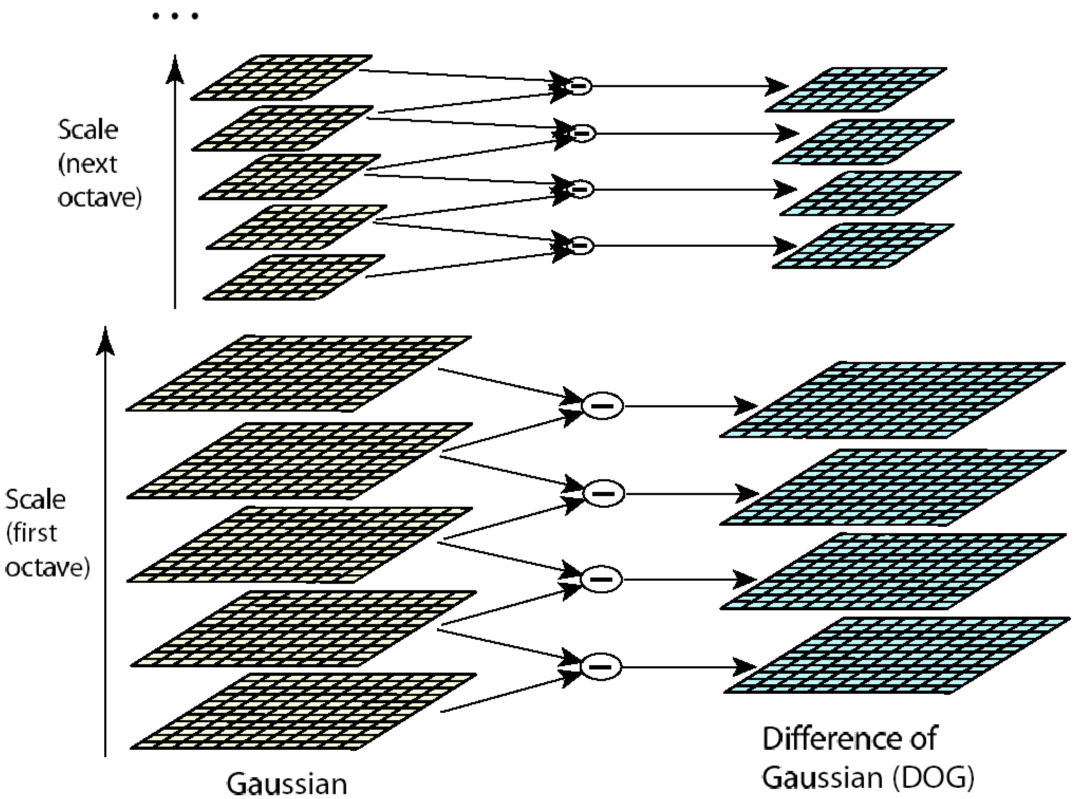


Source: http://fourier.eng.hmc.edu/e161/lectures/gradient/node11.html

SIFT: Scale-space extrema detection

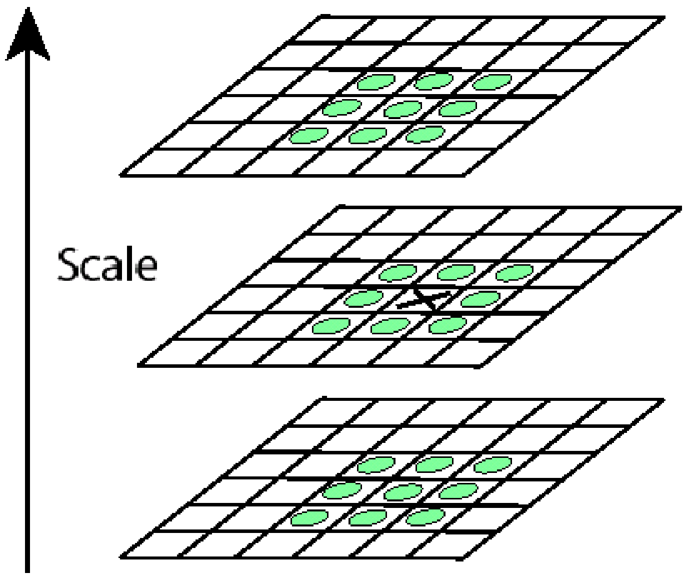
- Difference of Gaussian

$$\mathrm{DoG}(\underline{\mathbf{n}}) = (G(\underline{\mathbf{n}}, \sigma) - G(\underline{\mathbf{n}}, k\,\sigma)) * I(\underline{\mathbf{n}}) = L(\underline{\mathbf{n}}, \sigma) - L(\underline{\mathbf{n}}, k\,\sigma)$$



SIFT: Scale-space extrema detection

- Non-extremum suppression



- X is selected if it is larger or smaller than all 26 neighbors (alternatively other 3D windows)

SIFT: Keypoint localization

- Sub-pixel keypoint localization

$$\mathrm{DoG}(\underline{\mathbf{x}}) = \mathrm{DoG}(\underline{\mathbf{n}}) + \frac{\partial \mathrm{DoG}(\underline{\mathbf{n}})}{\partial \underline{\mathbf{x}}}\underline{\mathbf{x}} + \frac{1}{2}\underline{\mathbf{x}}^\top \frac{\partial^2 \mathrm{DoG}(\underline{\mathbf{n}})}{\partial \underline{\mathbf{x}}^2}\underline{\mathbf{x}}$$

- Finding the extrema by setting the derivative to zero

- Contrast threshold, reject feature if

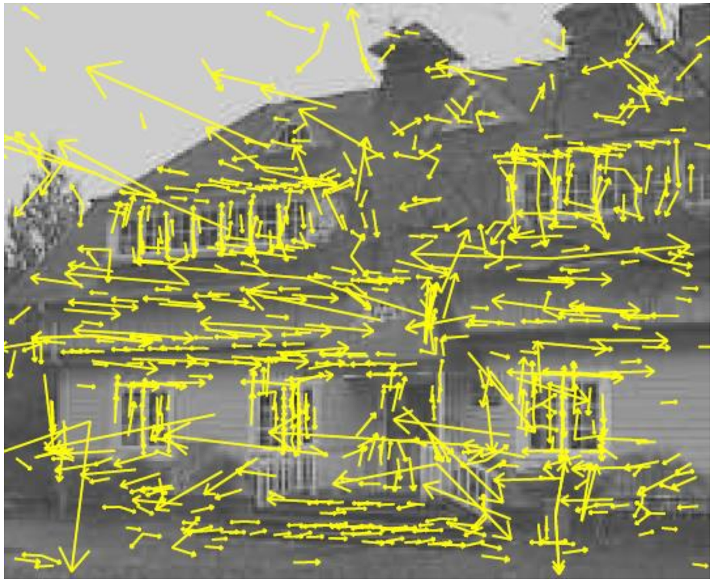
$$|\mathrm{DoG}(\hat{\underline{\mathbf{x}}})| < 0.03$$

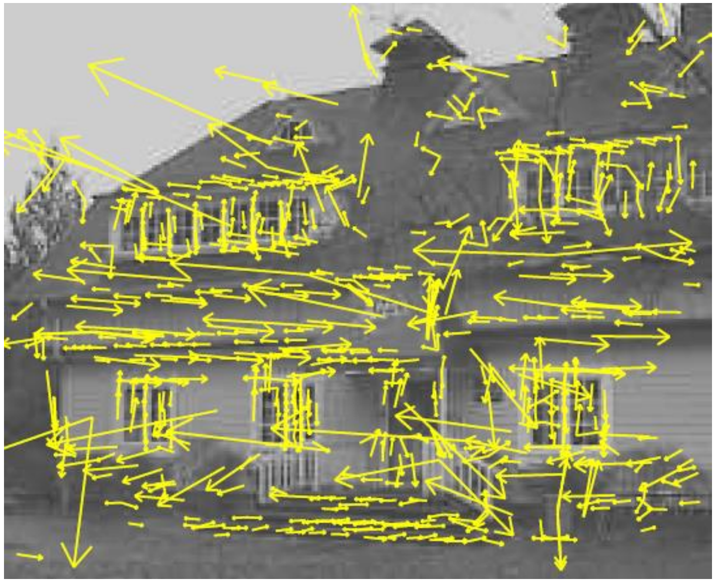
- Cornerness threshold: Reject edges and keep corners, reject feature if

$$\frac{\mathrm{trace}^2(\mathtt{G}(\hat{\underline{\mathbf{x}}}))}{\det(\mathtt{G}(\hat{\underline{\mathbf{x}}}))} > \tau$$

SIFT: Keypoint localization

- Contrast threshold

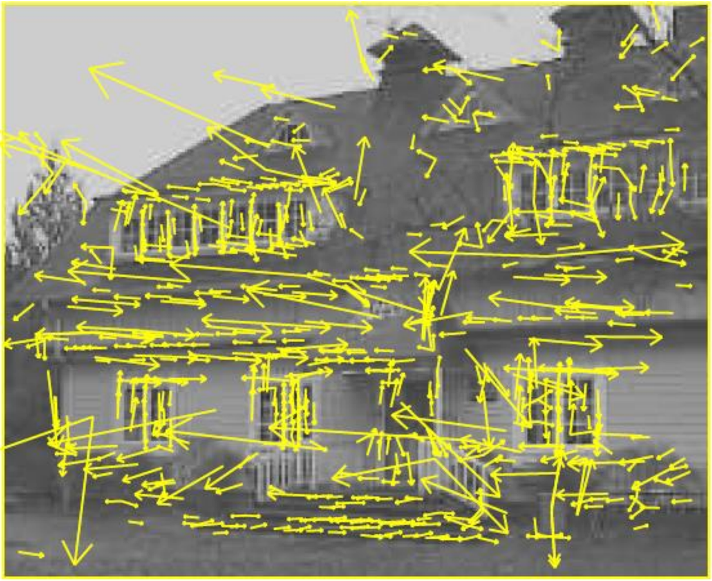


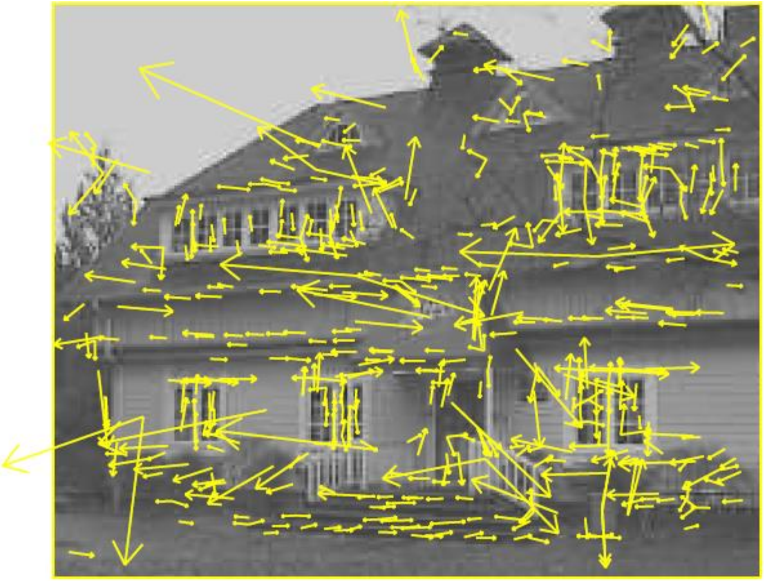


- 729 out of 832 feature are left after contrast thresholding

SIFT: Keypoint localization

- Cornerness threshold





- 536 out of 729 are left after cornerness thresholding

SIFT: Orientation assignment I

- Gradient magnitute for each pixel

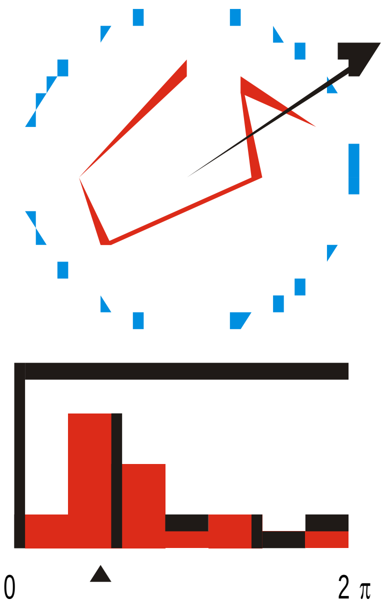
$$m(\underline{\mathbf{n}}) = \sqrt{(L(n_x+1,n_y)-L(n_x-1,n_y))^2+(L(n_x,n_y+1)-L(n_x,n_y-1))^2}$$

- Orientation for each pixel

$$\theta(\underline{\mathbf{n}}) = \tan^{-1}\left(\frac{L(n_x,n_y+1)-L(n_x,n_y-1)}{L(n_x+1,n_y)-L(n_x-1,n_y)}\right)$$

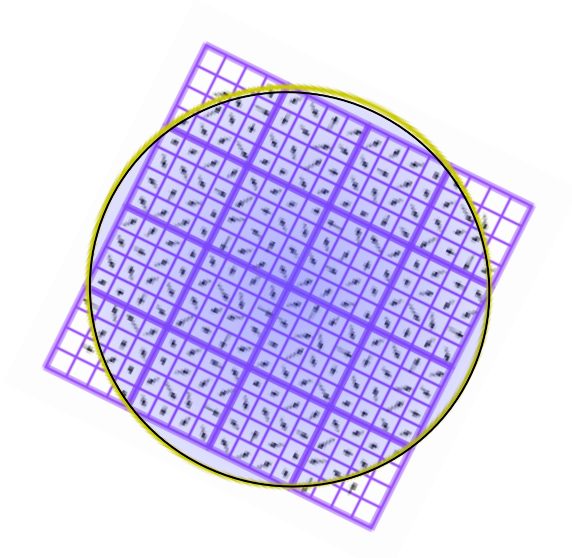
SIFT: Orientation assignment II

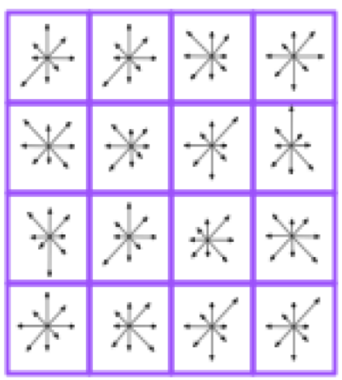
- Image location, scale, and orientation impose a repeatable local 2D coordinate system in which to describe the local image region
- A orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint.
- The orientation histogram has 36 bins covering the 360 degree range of orientations.
- The highest peak in the histogram is detected

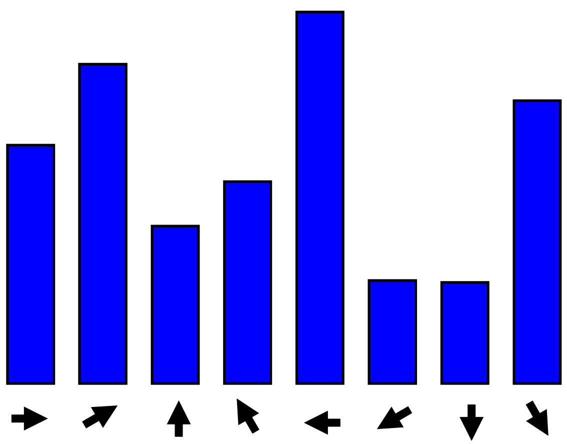


SIFT: Keypoint descriptor

- Create 16 gradient histograms (8 bins)
- Weighted by magnitude and Gaussian window (σ is half the window size)
- Histogram and gradient values are interpolated and smoothed





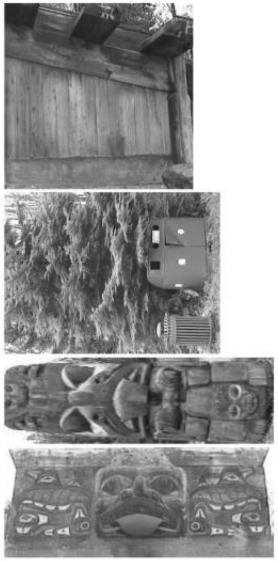


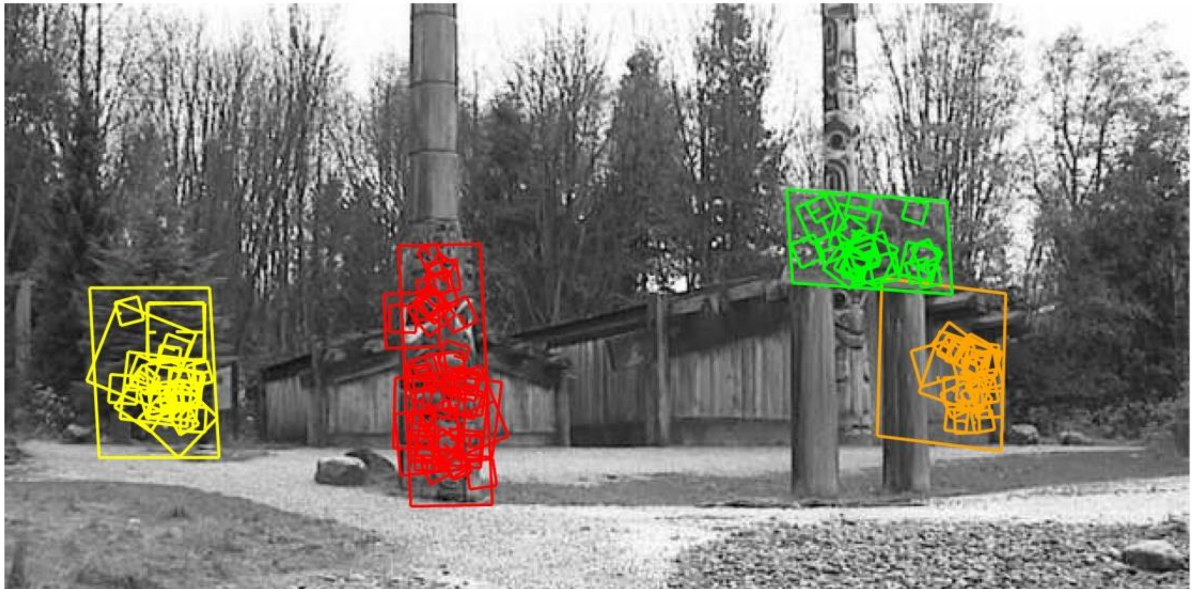
128-feature vector

SIFT: Descriptor matching

- Nearest Neighbor algorithm based on L2-norm
- How to discard bad matches?
  - Threshold on L2-norm => bad performance
  - Solution: threshold on ratio: (best match) / (second best match)

SIFT: Descriptor matching





Software

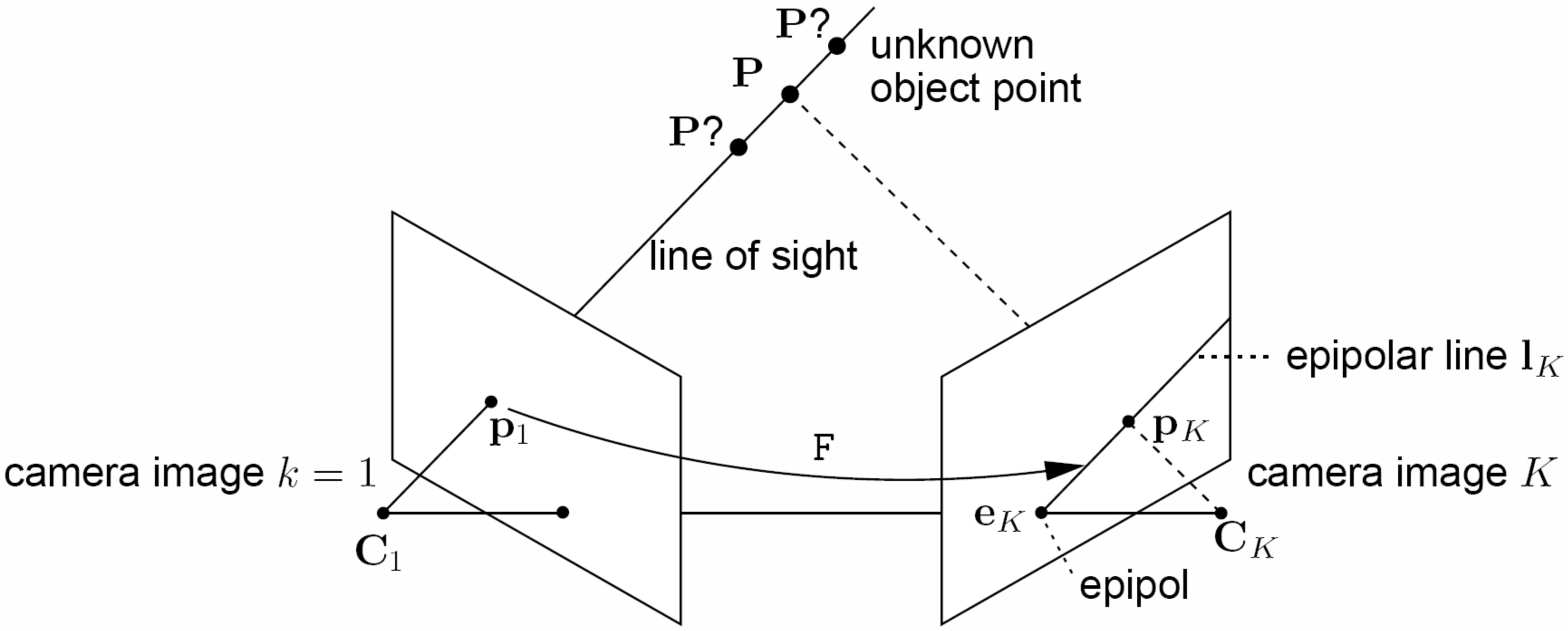
- VLFeat (http://www.vlfeat.org/index.html)
  - Implements SIFT and other modern features
  - C-implementation with MATLAB bindings
    - Win/Linux
  - Good tutorials on parameter selection

Towards Multiple Views and Self-Calibration

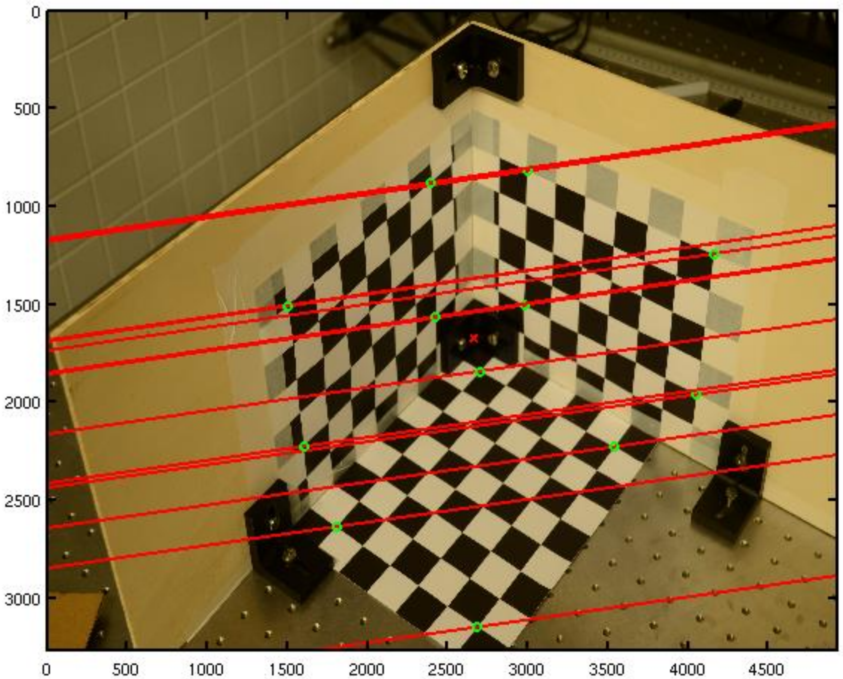
-- Two-View Geometry and Basic Stereo --

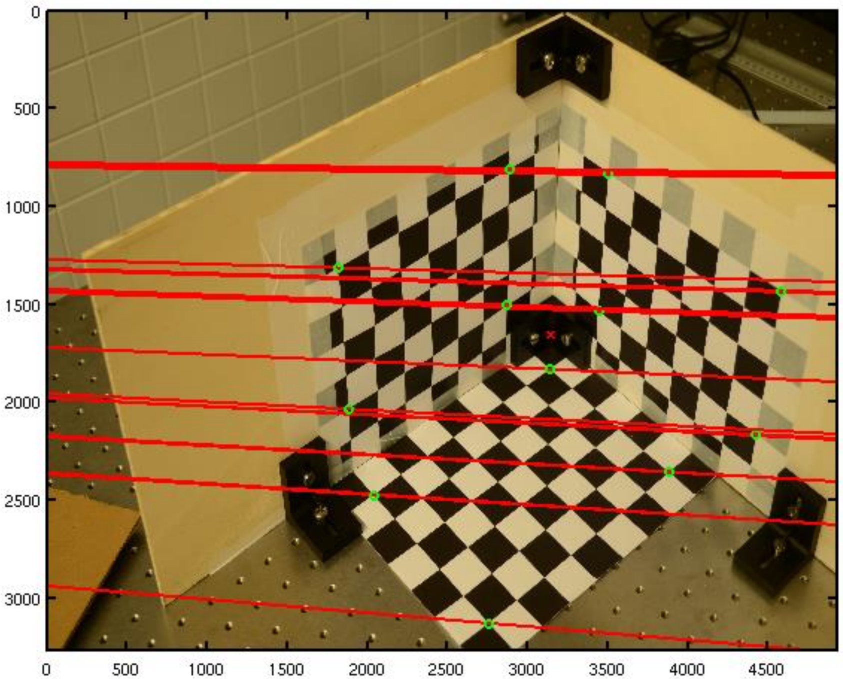
Fundamental Matrix

Fundamental Matrix (F-Matrix): $\mathbf{p}_K^\top \, \mathbf{l}_K = 0 \quad \Leftrightarrow \quad \mathbf{p}_K^\top \, \mathbf{F} \, \mathbf{p}_1 = 0$



Epipolar Geometry in our example





Fundamental Matrix

- F is a rank 2 homogeneous matrix with 7 degrees of freedom

- Epipolar lines

$$\mathbf{l}_2 = \mathtt{F}\,\mathbf{p}_1$$

$$\mathbf{l}_1 = \mathtt{F}^\top \mathbf{p}_2$$

- Epipoles

$$\mathtt{F}\,\mathbf{e}_1 = \mathbf{0}$$

$$\mathtt{F}^\top \mathbf{e}_2 = \mathbf{0}$$

Fundamental Matrix

- F can be computed from camera matrices

- General projective cameras:

$$\mathtt{F} = [\mathbf{e}_2]_\times \mathtt{A}_2 \mathtt{A}_1^+$$

$$\mathbf{e}_2 = \mathtt{A}_2 \, \mathbf{C}_1 \qquad\qquad \mathtt{A}_1 \, \mathbf{C}_1 = \mathbf{0}$$

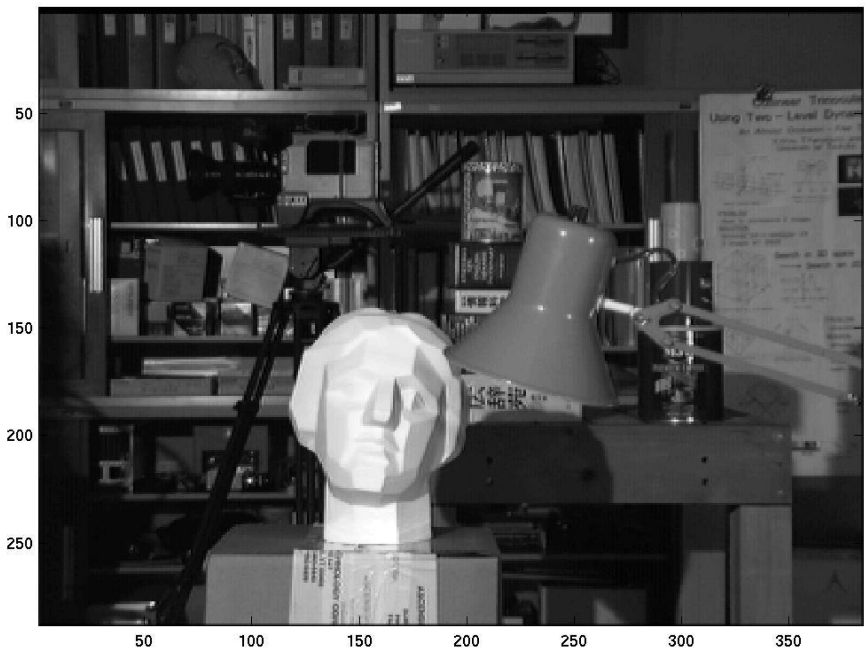
$$[\mathbf{e}_K]_\times = [(e_x, e_y, e_z)^\top]_\times = \begin{bmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{bmatrix}$$

- Canonical cameras not at infinity $\mathtt{A}_1 = \mathtt{K}_1[\mathtt{I}|\mathbf{0}]$ and $\mathtt{A}_1 = \mathtt{K}_2[\mathtt{R}_2|\mathbf{t}_2]$

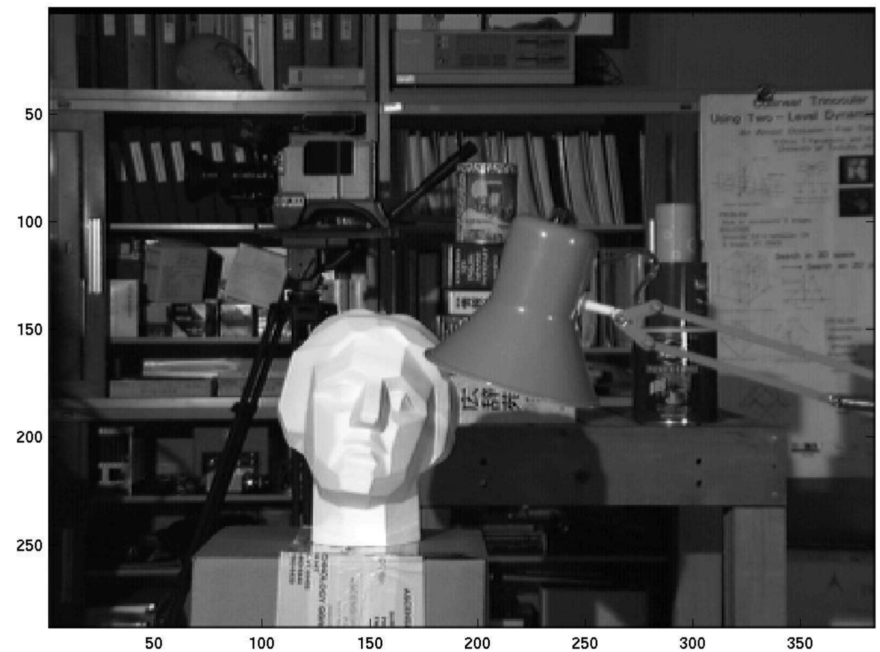
$$\mathtt{F} = \mathtt{K}_2^{-\top}[\mathbf{t}_2]_\times \mathtt{R}_2 \mathtt{K}_1^{-1}$$

Use Case – Stereo Matching

- Typical left and right image with parallel optical axes and only horizontally displaced



Left image

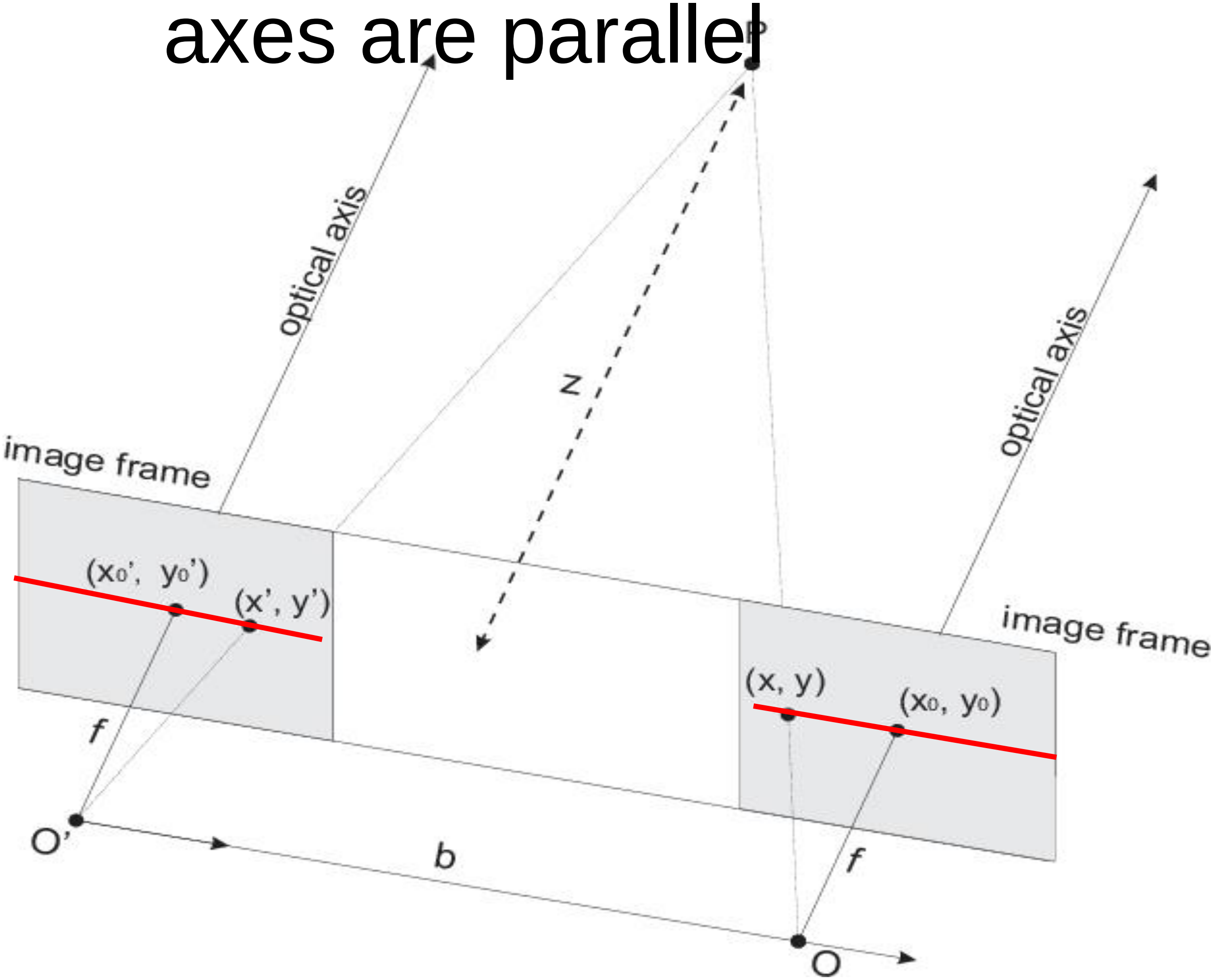


Right image

[implementation by Rohit Singh and Mitul Sara]

Epipolar Geometry of a Stereo Pair

- Epipolar lines are parallel lines if optical axes are parallel



Epipolar Geometry of a Stereo Pair

- Simplest: Search a moving window and perform correlation
- E.g. SSD (sum of squared differences)

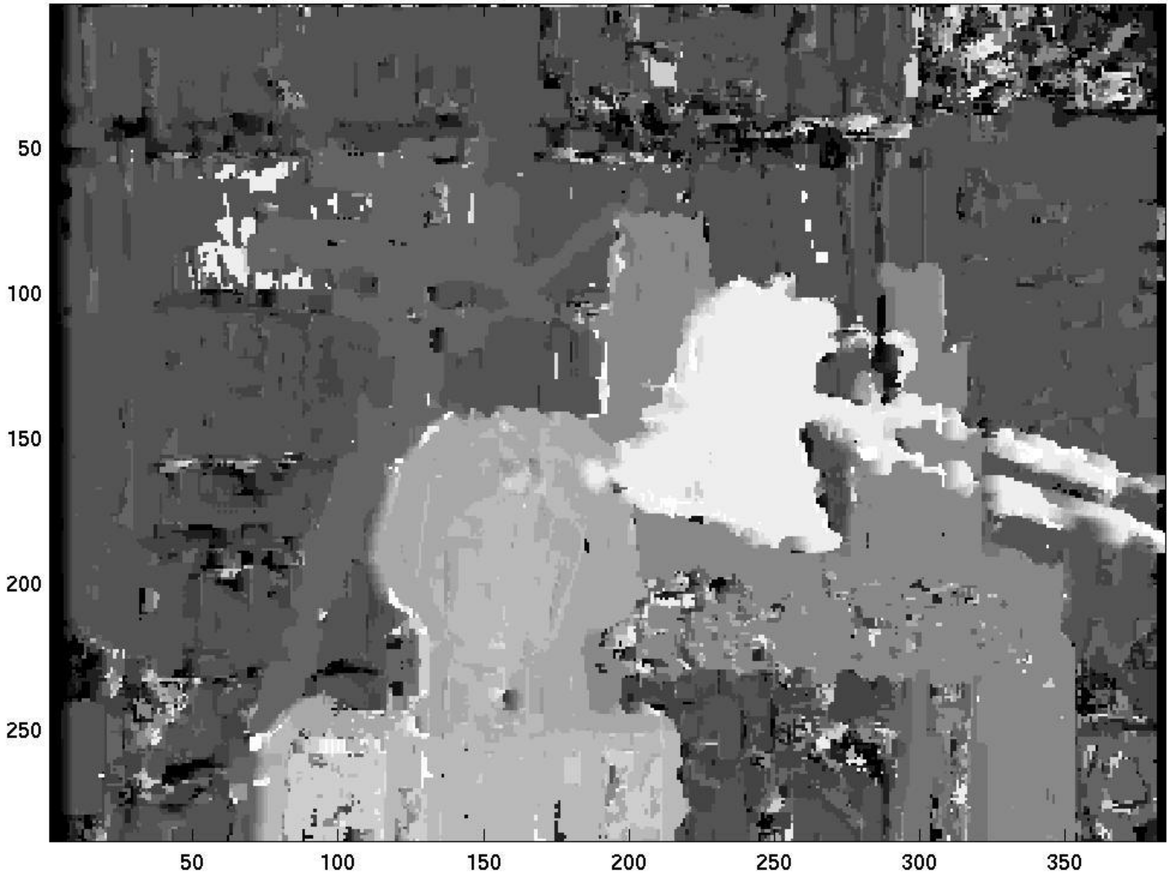
$$\mathrm{SSD}(\underline{\mathbf{d}}) = \sum_{m_x} \sum_{m_y} \left[I_K(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{d}} + \underline{\mathbf{m}}) - I_{K-1}(\underline{\mathbf{n}}_{K-1} + \underline{\mathbf{m}}) \right]^2 \rightarrow \min$$

$$\mathbf{d} = s \cdot \begin{pmatrix} d_x \\ d_y \end{pmatrix} \text{ with } s \in \mathbb{R} \text{ and } \begin{pmatrix} d_x \\ d_y \end{pmatrix} \text{const.}$$

- Alternatively, SAD (sum of absolute differences), CC (cross correlation), etc.
- Usually restricted search range

Correlation-Based Stereo Matching

- Example, SSD, win=5, search range=15



[implementation by Rohit Singh and Mitul Saha]

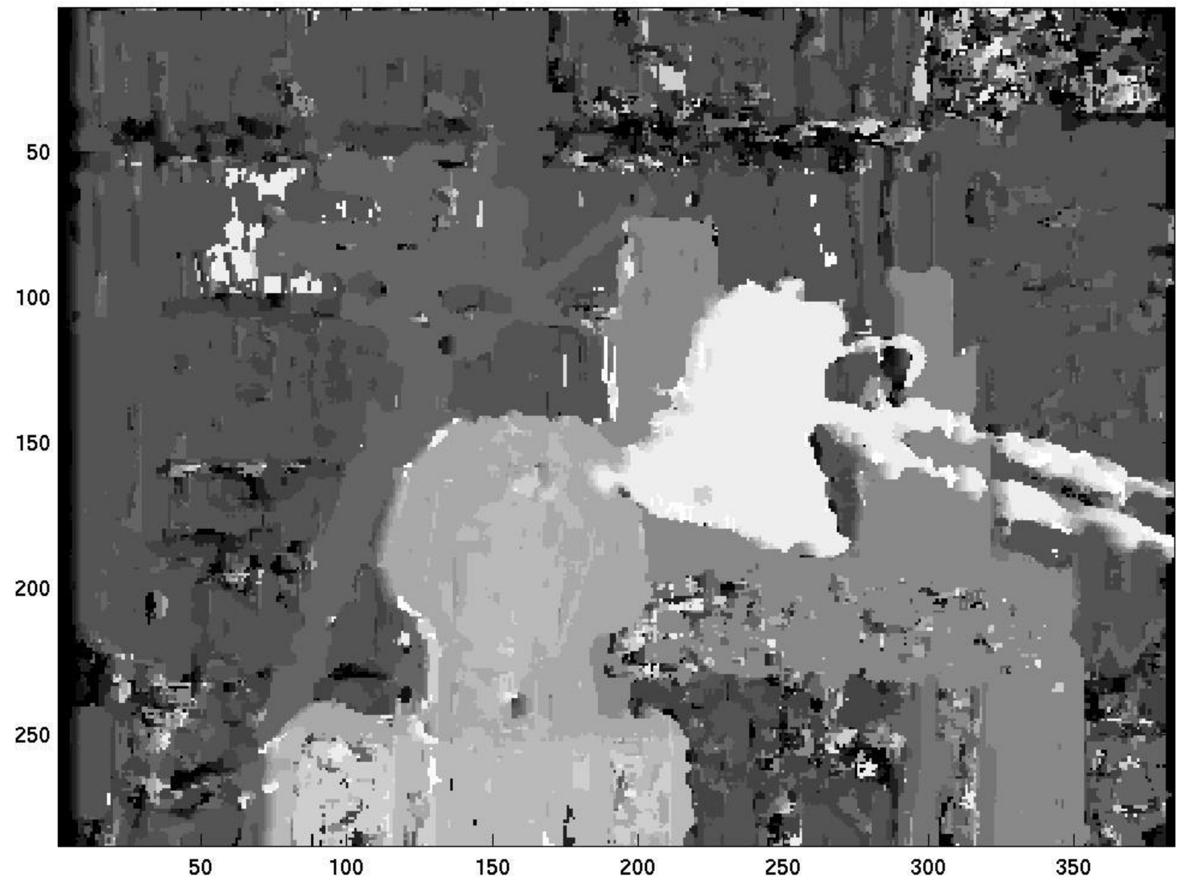
Correlation-Based Stereo Matching

- Example, SSD, win=5, search range=**8**

[implementation by Rohit Singh and Mitul Saha]

Correlation-Based Stereo Matching

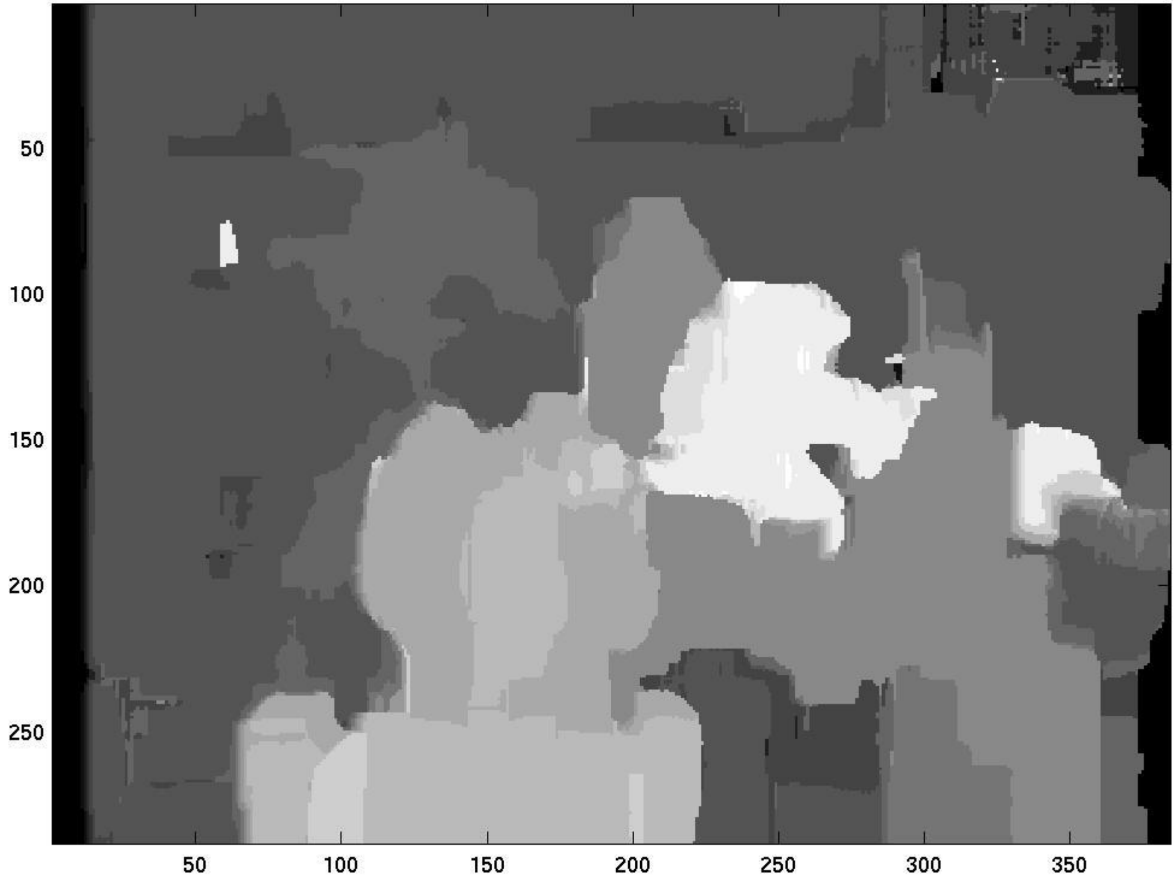
- Example, **SAD**, win=5, search range=8



[implementation by Rohit Singh and Mitul Saha]

Correlation-Based Stereo Matching

- Example, SSD, win=**20**, search range=8



[implementation by Rohit Singh and Mitul Saha]

A Modern Technique

- ADCensus [Mei'11] (#1 Middlebury stereo benchmark, Sept. 2013)

Towards Multiple Views and Self-Calibration

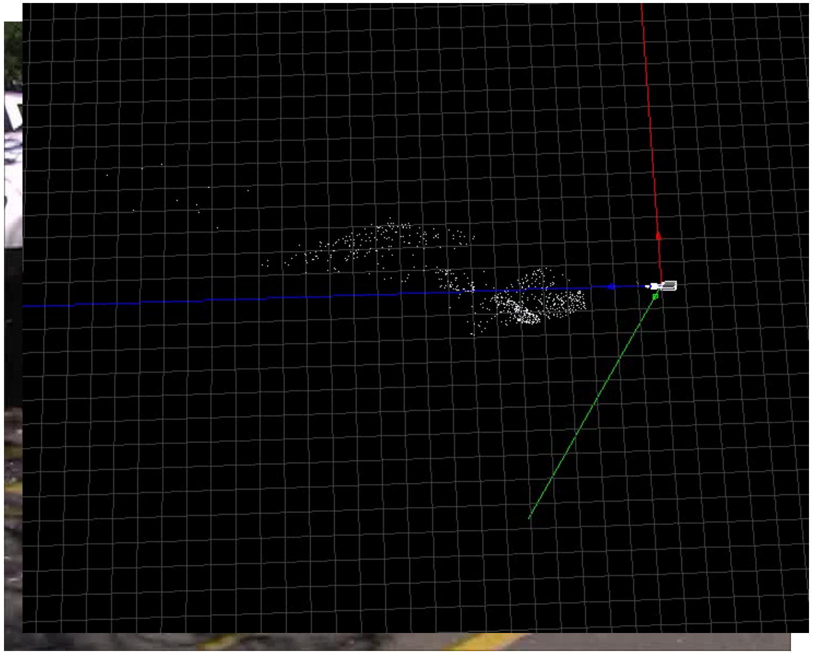
-- The Structure-from-motion Pipeline --

Voodoo Camera Tracker - Steps of camera tracking

image sequence

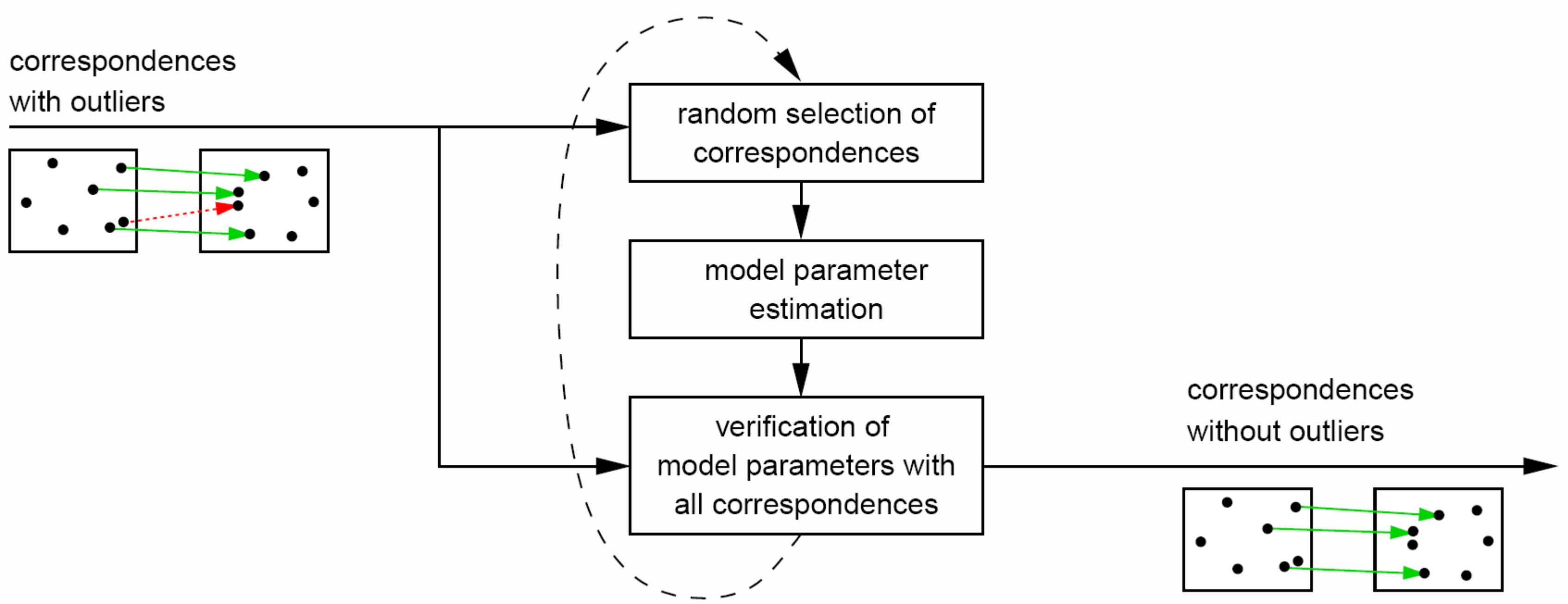
Feature detection and correspondence analysis → Outlier Elimination → Incremental Bundle Adjustment → Self-Calibration

camera parameters

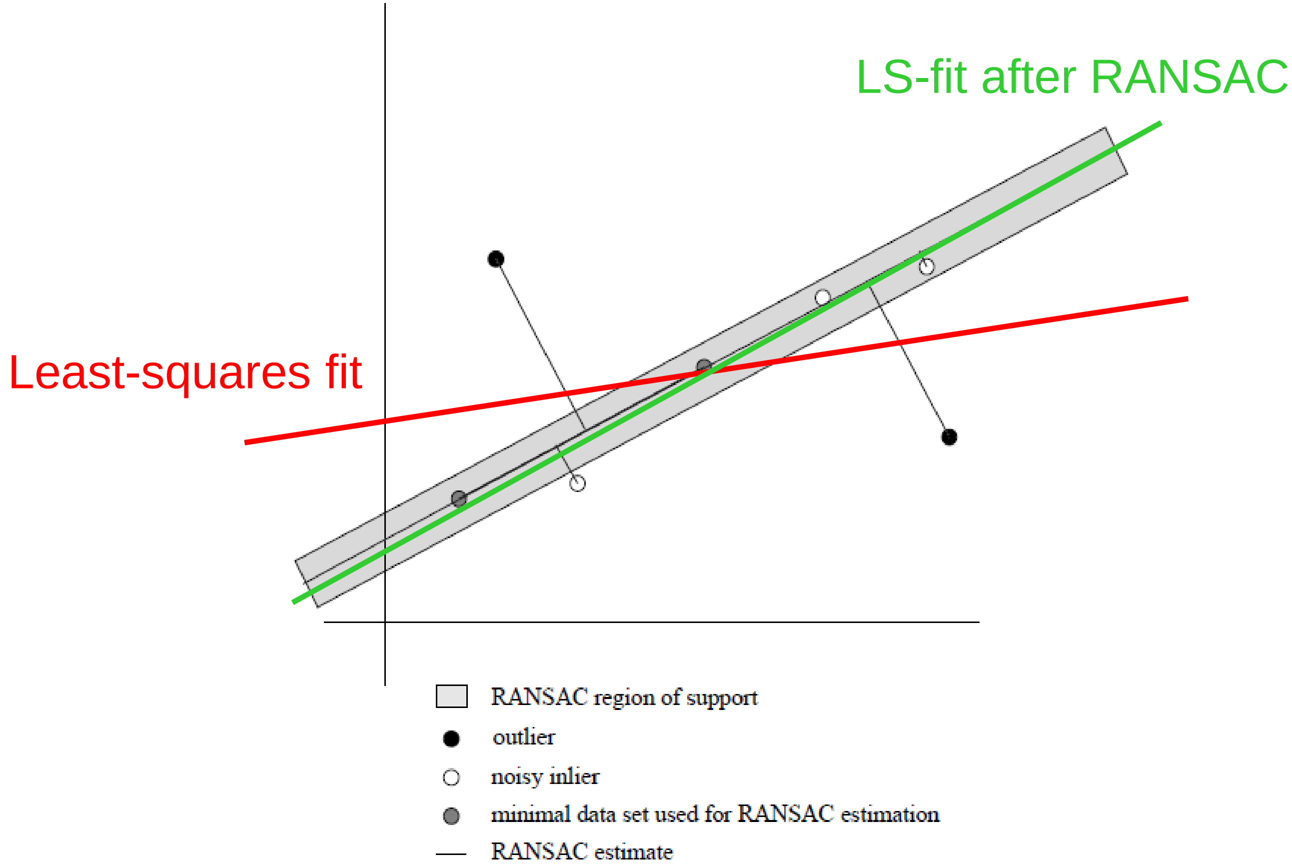


Outlier Elimination

- RANSAC (Random Sample Consensus) method

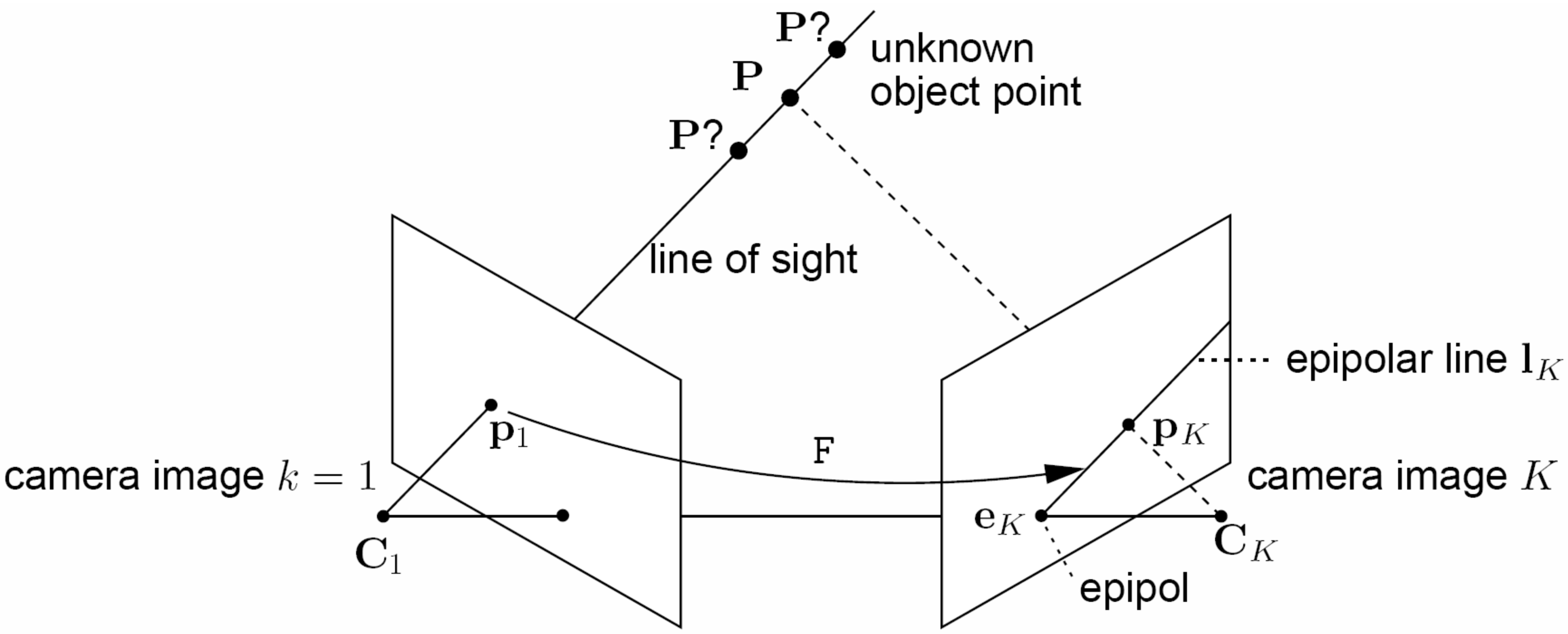


RANSAC example – line fit



Outlier Elimination - Fundamental Matrix

Fundamental Matrix (F-Matrix): $\mathbf{p}_K^\top \mathbf{l}_K = 0 \quad \Leftrightarrow \quad \mathbf{p}_K^\top \mathbf{F} \mathbf{p}_1 = 0$



Fundamental Matrix Estimation

Estimating the fundamental matrix from feature correspondence:

$$\mathbf{p}_2^T \, \mathbf{F} \, \mathbf{p}_1 = 0$$

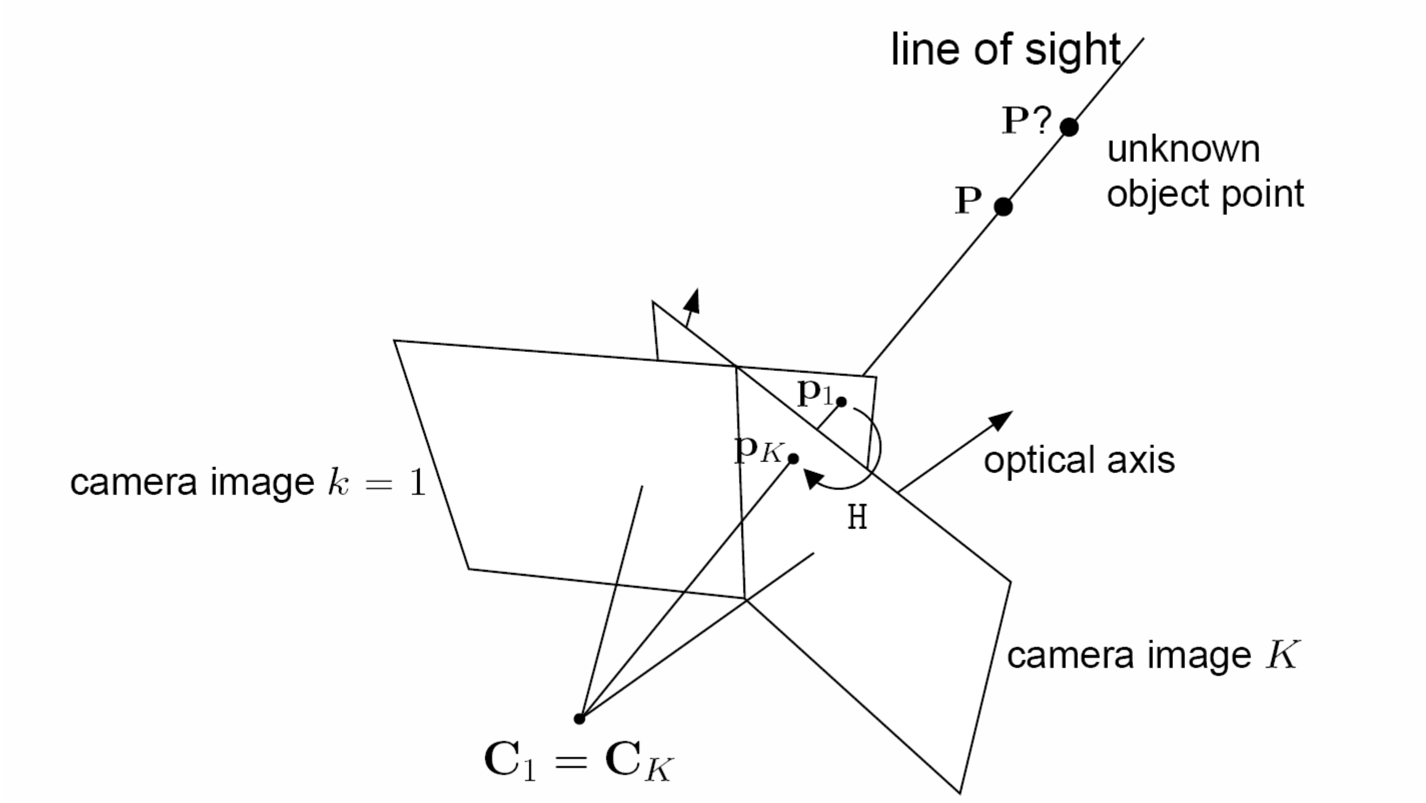
$$(x_2, y_2, 1) \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = 0$$

$$\begin{pmatrix} x_2 x_1 & x_2 y_1 & x_2 & y_2 x_1 & y_2 y_1 & y_2 & x_1 & y_1 & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

multiple of these equations gives a linear equation system $\mathtt{A}\,\mathbf{f} = \mathbf{0}$

Outlier Elimination - 2D Homography

2D Homography (H-Matrix $\mathbf{p}_K = \mathtt{H}\,\mathbf{p}_1$



2D Homography Matrix Estimation

Estimating the 2D homography matrix from feature correspondence:

$$\begin{aligned} \mathtt{H}\,\mathbf{p}_1 &= \mathbf{p}_2 \\ \mathbf{p}_2 \times \mathtt{H}\,\mathbf{p}_1 &= \mathbf{p}_2 \times \mathbf{p}_2 \\ \mathbf{p}_2 \times \mathtt{H}\,\mathbf{p}_1 &= \mathbf{0} \end{aligned}$$

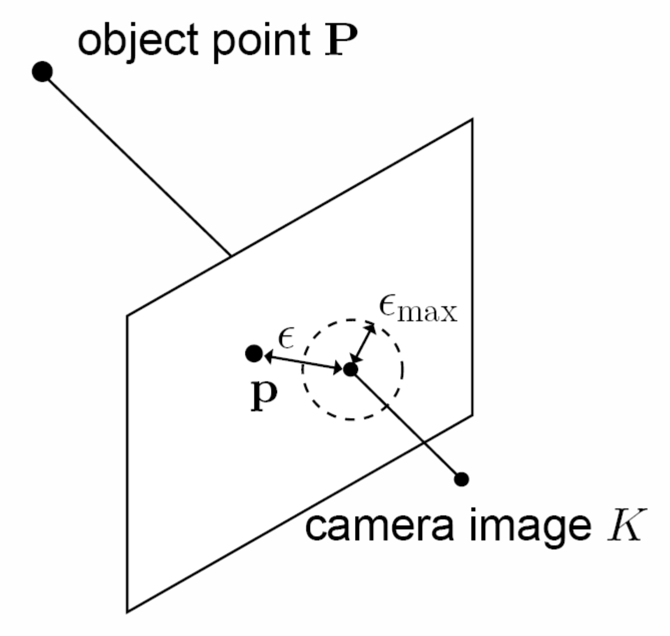
$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} \times \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \mathbf{0}$$

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_2x_1 & -x_2y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_2x_1 & -y_2y_1 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

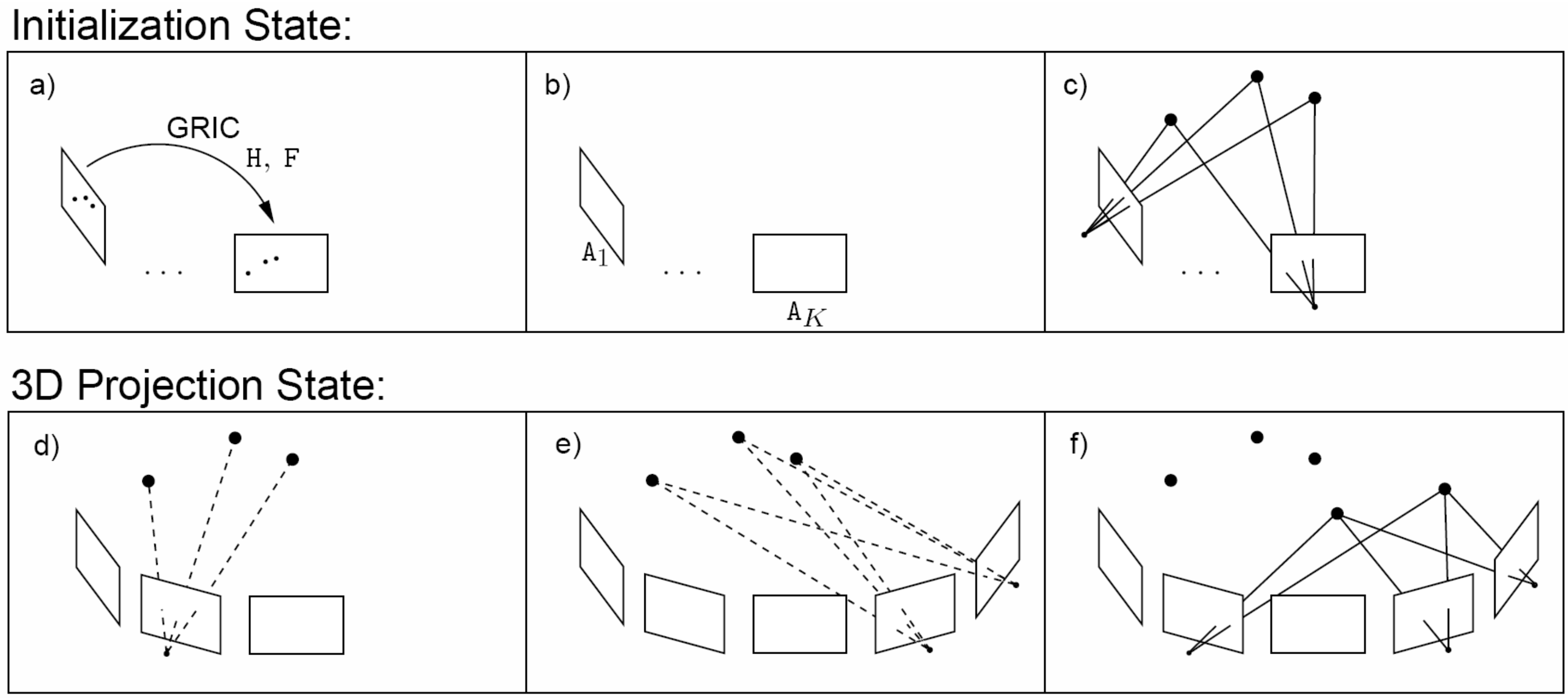
multiple of these equations gives a linear equation system $\mathtt{A}\,\mathbf{h} = \mathbf{b}$

Outlier Elimination - Camera Matrix

Camera matrix (A-Matrix): $\mathbf{p}_{j,\,K} = \mathtt{A}_K\,\mathbf{P}_j$

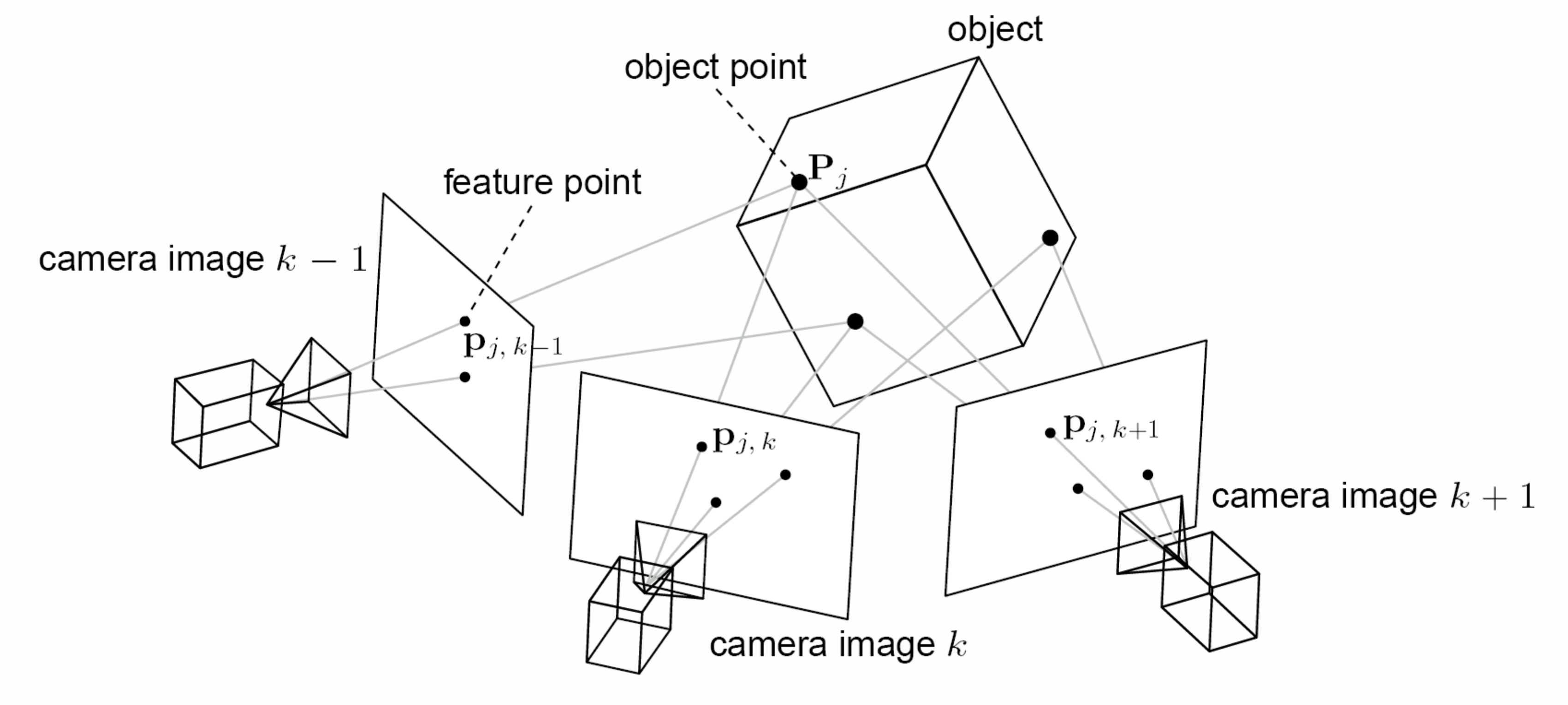


Incremental Bundle Adjustment



Bundle Adjustment

Bundle Adjustment



$$\underset{\hat{\mathtt{A}}_k,\hat{\mathbf{P}}_j}{\arg\min} \sum_k \sum_j d(\tilde{\mathbf{p}}_{j,\,k},\ \hat{\mathtt{A}}_k\,\hat{\mathbf{P}}_j)^2$$

Levenberg Marquardt - Non-linear Least Squares

$$\arg\min_{\hat{\mathbf{X}}} \left(\tilde{\mathbf{Y}} - f(\hat{\mathbf{X}})\right)^2$$

$\tilde{\mathbf{Y}} \in \mathbb{R}^N$ Measurement vector

$\hat{\mathbf{X}} \in \mathbb{R}^M$ Parameter vector

Taylor approximation:

$$f(\hat{\mathbf{X}}) = f(\mathbf{X}_0 + \mathbf{\Delta}_{\hat{\mathbf{X}}}) \approx f(\mathbf{X}_0) + \mathrm{J}\mathbf{\Delta}_{\hat{\mathbf{X}}}$$

with N x M Jacobian Matrix

$$\mathrm{J} = \frac{\partial \tilde{\mathbf{Y}}}{\partial \hat{\mathbf{X}}}$$

Levenberg Marquardt - Non-linear Least Squares

$$
\begin{aligned}
& \arg\min_{\hat{\mathbf{X}}} \left(\tilde{\mathbf{Y}} - f(\hat{\mathbf{X}})\right)^2 \\
= \quad & \arg\min_{\boldsymbol{\Delta}_{\hat{\mathbf{X}}}} \left(\tilde{\mathbf{Y}} - f(\mathbf{X}_0 + \boldsymbol{\Delta}_{\hat{\mathbf{X}}})\right)^2 \\
\approx \quad & \arg\min_{\boldsymbol{\Delta}_{\hat{\mathbf{X}}}} \left(\tilde{\mathbf{Y}} - (f(\mathbf{X}_0) + \mathsf{J}\boldsymbol{\Delta}_{\hat{\mathbf{X}}})\right)^2 \\
= \quad & \arg\min_{\boldsymbol{\Delta}_{\hat{\mathbf{X}}}} \left((\tilde{\mathbf{Y}} - f(\mathbf{X}_0)) - \mathsf{J}\boldsymbol{\Delta}_{\hat{\mathbf{X}}}\right)^2 \\
= \quad & \arg\min_{\boldsymbol{\Delta}_{\hat{\mathbf{X}}}} \left(\epsilon_0 - \mathsf{J}\boldsymbol{\Delta}_{\hat{\mathbf{X}}}\right)^2
\end{aligned}
$$

transformed to linear least squares problem for each iteration

$$
\hat{\mathbf{X}}_1 = \hat{\mathbf{X}}_0 + \boldsymbol{\Delta}_{\hat{\mathbf{X}}} \quad , \qquad \hat{\mathbf{X}}_2 = \hat{\mathbf{X}}_1 + \boldsymbol{\Delta}_{\hat{\mathbf{X}}} \quad , \qquad \ldots
$$

Levenberg Marquardt - Non-linear Least Squares

Linear least squares problem can be solved with normal equations

$$\begin{aligned}
\epsilon_0 - \mathrm{J}\boldsymbol{\Delta}_{\hat{\mathbf{X}}} &= \mathbf{0} \\
\mathrm{J}\boldsymbol{\Delta}_{\hat{\mathbf{X}}} &= \epsilon_0 \\
\mathrm{J}^T\mathrm{J}\boldsymbol{\Delta}_{\hat{\mathbf{X}}} &= \mathrm{J}^T\epsilon_0 \\
\boldsymbol{\Delta}_{\hat{\mathbf{X}}} &= (\mathrm{J}^T\mathrm{J})^{-1}\mathrm{J}^T\epsilon_0
\end{aligned}$$

use to update solution iteratively

$$\hat{\mathbf{X}}_1 = \hat{\mathbf{X}}_0 + \boldsymbol{\Delta}_{\hat{\mathbf{X}}} \quad , \qquad \hat{\mathbf{X}}_2 = \hat{\mathbf{X}}_1 + \boldsymbol{\Delta}_{\hat{\mathbf{X}}} \quad , \qquad \ldots$$

Levenberg Marquardt - Non-linear Least Squares

- Levenberg Marquardt uses slightly different normal equations

  Original normal equations

$$\mathrm{J}^T \mathrm{J} \boldsymbol{\Delta}_{\hat{\mathbf{X}}} \quad = \quad \mathrm{J}^T \epsilon_0$$

  Modified normal equations

$$\left[\mathrm{J}^T \mathrm{J} + \lambda \, \mathrm{diag}(\mathrm{J}^T \mathrm{J})\right] \boldsymbol{\Delta}_{\hat{\mathbf{X}}} \quad = \quad \mathrm{J}^T \epsilon_0$$

- Lambda is changed during optimization

$$\lambda_{i+1} \quad = \quad 0.1 \, \lambda_i \qquad \text{if} \quad \epsilon_i < \epsilon_{i-1}$$ successful iteration

$$\lambda_{i+1} \quad = \quad 10.0 \, \lambda_i \qquad \text{if} \quad \epsilon_i > \epsilon_{i-1}$$ failed iteration

small λ ~ Newton style (quadratic convergence)

large λ ~ Gradient descent style (guaranteed decrease)

Levenberg Marquardt

Requirements for Levenberg Marquardt minimization

- Function to compute f
- Start value X_0
- Optionally, function to compute J

(but numerical derivation works as well)

Sparse Levenberg Marquardt

- For bundle adjustment the problem becomes to large

  (100 cameras + 10000 3D object points = 31200 parameter)

- Can achieve huge speed-up by exploiting sparse structure of Jacobian matrix

- Partition parameters
  - partition A
  - partition B (only dependent on A and itself)

(typically A contains camera parameters, and B contains 3D points)

Sparse Levenberg Marquardt

Jacobian becomes

$$\mathtt{J} = [\mathtt{A}|\mathtt{B}]$$

$$\begin{aligned} \mathtt{J}\boldsymbol{\Delta}_{\hat{\mathbf{X}}} &= \epsilon \\ [\mathtt{A}|\mathtt{B}] \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} &= \epsilon \end{aligned}$$

Normal equations

$$\mathtt{J}^T\mathtt{J}\boldsymbol{\Delta}_{\hat{\mathbf{X}}} = \mathtt{J}^T\epsilon_0$$

become

$$\begin{bmatrix} \mathtt{A}^T\mathtt{A} & \mathtt{A}^T\mathtt{B} \\ \mathtt{B}^T\mathtt{A} & \mathtt{B}^T\mathtt{B} \end{bmatrix} \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} = \begin{pmatrix} \mathtt{A}^T\epsilon \\ \mathtt{B}^T\epsilon \end{pmatrix}$$

Sparse Levenberg Marquardt

Corresponding block structure is

$$\begin{bmatrix} \mathtt{U}^* & \mathtt{W} \\ \mathtt{W}^\top & \mathtt{V}^* \end{bmatrix} \begin{pmatrix} \boldsymbol{\delta_a} \\ \boldsymbol{\delta_b} \end{pmatrix} = \begin{pmatrix} \epsilon_\mathtt{A} \\ \epsilon_\mathtt{B} \end{pmatrix}$$

where $\mathtt{U}^*$ denotes $\mathtt{U}$ augmented by multiplying its diagonal entries by a factor of 1 + λ and $\mathtt{V}^*$ likewise. Left multiplication with

$$\begin{bmatrix} \mathtt{I} & -\mathtt{W}\mathtt{V}^{*-1} \\ \mathtt{0} & \mathtt{I} \end{bmatrix}$$

yields

$$\begin{bmatrix} \mathtt{U}^* - \mathtt{W}\mathtt{V}^{*-1}\mathtt{W}^\top & \mathtt{0} \\ \mathtt{W}^\top & \mathtt{V}^* \end{bmatrix} \begin{pmatrix} \boldsymbol{\delta_a} \\ \boldsymbol{\delta_b} \end{pmatrix} = \begin{pmatrix} \epsilon_\mathtt{A} - \mathtt{W}\mathtt{V}^{*-1}\epsilon_\mathtt{B} \\ \epsilon_\mathtt{B} \end{pmatrix}$$

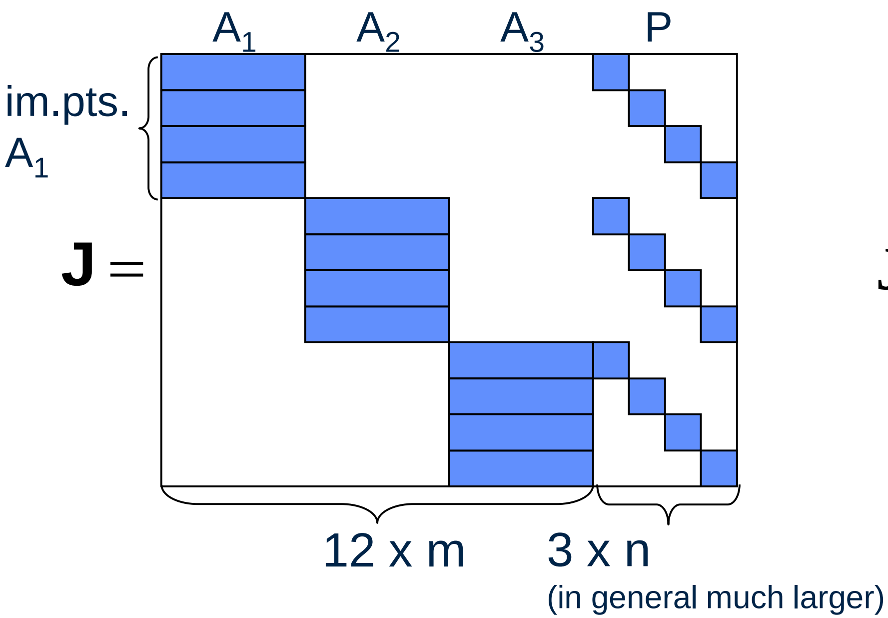
which can be used to find $\boldsymbol{\delta_a}$ with $\left(\mathtt{U}^* - \mathtt{W}\mathtt{V}^{*-1}\mathtt{W}^\top\right)\boldsymbol{\delta_\mathbf{a}} = \epsilon_\mathtt{A} - \mathtt{W}\mathtt{V}^{*-1}\epsilon_\mathtt{B}$

which may be back-substituted to get $\boldsymbol{\delta_b}$ with $\mathtt{V}^*\boldsymbol{\delta_b} = \epsilon_\mathtt{B} - \mathtt{W}^\top\boldsymbol{\delta_a}$

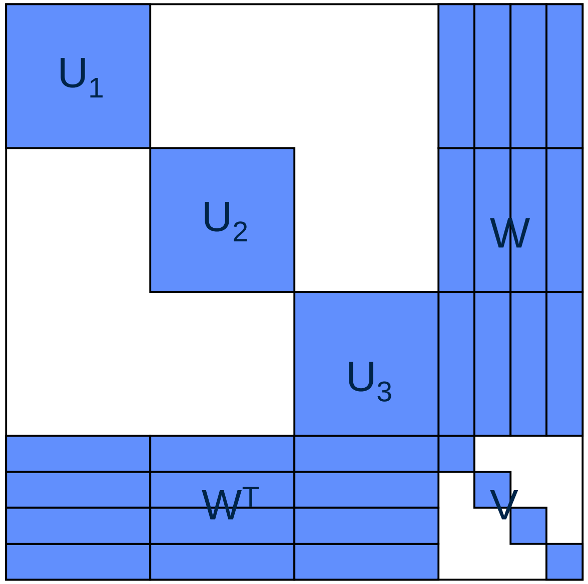
Sparse Levenberg Marquardt

Jacobian for bundle adjustment has sparse block structure

$\mathbf{J} =$



$\mathbf{J}^{\mathrm{T}}\mathbf{J} =$



Needed for non-linear minimization

Voodoo camera tracker – demo session

Self-calibration

- Why self-calibration?
  - Allows flexible acquisition
  - No prior calibration necessary
  - Possibility to vary intrinsic camera parameters
  - Use archive footage

Self-calibration

- We want to find a 4x4 transformation matrix $\mathtt{T}_M$ that transforms all projective cameras into metric cameras $\breve{\mathtt{A}}_k$

$$\breve{\mathtt{A}}_k = \mathtt{A}_k \mathtt{T}_M \qquad \forall \quad k$$

$$\breve{\mathbf{P}}_j = \mathtt{T}_M^{-1} \, \mathbf{P}_j \qquad \forall \quad j$$

- This does not change the back-projections onto the feature points

$$\mathbf{p}_{j,k} = \breve{\mathtt{A}}_k \breve{\mathbf{P}}_j = \mathtt{A}_k \mathtt{T}_M \mathtt{T}_M^{-1} \, \mathbf{P}_j = \mathtt{A}_k \mathbf{P}_j$$

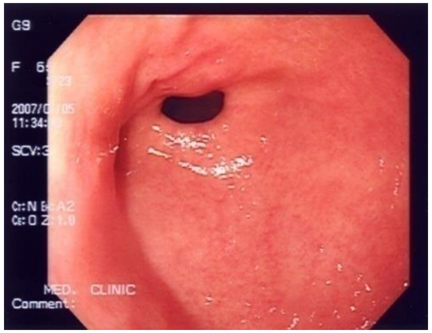
Voodoo Camera Tracker - Some applications

- Virtual advertising

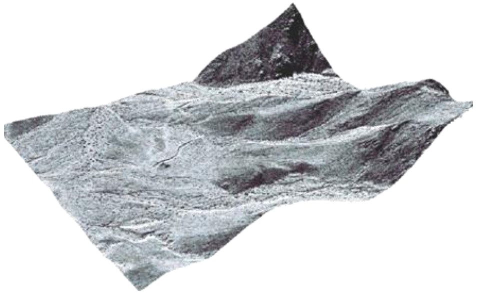
Voodoo Camera Tracker - More applications

- Car navigation


- 3D Endoscopy

- UAV terrain reconstruction

- Architectural visualisation


Matchmoving in Cloverfield

Matchmoving in Cloverfield

References

1. Multiple View Geometry in Computer Vision, Richard Hartley and Andrew Zisserman, 2nd Edition
2. Triggs, B.: “Autocalibration and the absolute quadric”. In “IEEE Conference on Computer Vision and Pattern Recognition”, S. 609–614. 1997
3. Pollefeys, M., Koch, R., Gool, L. V.: “Self-Calibration and Metric Reconstruction in Spite of Varying and Unknown Internal Camera Parameters”. In “EEE International Conference on Computer Vision, S. 90–95. 1998

Homework: Camera motion estimation with Voodoo

- Download and install voodoo camera tracker 1.0.1 from course website or from

    http://www.digilab.uni-hannover.de/download.html

- Download and install Blender 2.49b from course website or from

    http://www.blender.org/download/get-blender/

    (maybe you also need to install Python)

Homework: Camera motion estimation with Voodoo

- Perform camera tracking with voodoo using the „free move“ example sequence
- Export data File → Save → Blender Python Script
- Load the python script into Blender's text editor and execute the script with ALT-P.
- Select the "voodoo_render_cam" camera and press Ctrl-Numpad 0 to activate it.
- To display the image sequence as a backbuffer in Blender, go to [Buttons

Homework: Camera motion estimation with Voodoo

- Add a (animated) 3D model of your choice
- Render the sequence
- Submit frame 190, 200, and 210 as jpeg to thormae@mpii.de
- Submit before next lecture

- (optional) generate an avi-file and put it to a webspace that is accessible for me and send the link

That's it for today

Source: Belfast Telegraph, UK