Geometry for Computer Vision

Lecture 5b

Calibrated Multi View Geometry

Per-Erik Forssén

Overview

- The 5-point Algorithm
- Structure from Motion
- Bundle Adjustment

Planar degeneracy

In the uncalibrated case, two view geometry is encoded by the fundamental matrix

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

Planar degeneracy

In the uncalibrated case, two view geometry is encoded by the fundamental matrix

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

If all scene points lie on a plane, or if the camera has undergone a pure rotation (no translation), we also have:

$$\mathbf{x}_1 = \mathbf{H} \mathbf{x}_2$$

Big trouble!

Planar degeneracy

- If $\mathbf{x}_1 = \mathbf{H}\mathbf{x}_2$, then the epipolar constraint becomes $\mathbf{x}_1^T\mathbf{F}\mathbf{x}_2 = \mathbf{x}_1^T\mathbf{F}\mathbf{H}^{-1}\mathbf{x}_1 = 0$
- For $\mathbf{M} = \mathbf{F}\mathbf{H}^{-1}$, this is true whenever **M** is skew-symmetric, i.e.

$$\mathbf{M}^T + \mathbf{M} = 0 \qquad \Leftrightarrow \qquad \mathbf{M} = [\mathbf{m}]_\times$$

Planar degeneracy

If $\mathbf{x}_1 = \mathbf{H}\mathbf{x}_2$, then the epipolar constraint becomes $\mathbf{x}_1^T\mathbf{F}\mathbf{x}_2 = \mathbf{x}_1^T\mathbf{F}\mathbf{H}^{-1}\mathbf{x}_1 = 0$

For $\mathbf{M} = \mathbf{F}\mathbf{H}^{-1}$, this is true whenever **M** is skew-symmetric, i.e.

$$\mathbf{M}^T + \mathbf{M} = 0 \qquad \Leftrightarrow \qquad \mathbf{M} = [\mathbf{m}]_\times$$

Thus $\mathbf{F} = [\mathbf{m}]_\times \mathbf{H}$ where **m** may be chosen freely!

A two-parameter family of solutions.

The 5-point algorithm

Recap from last week's lecture...

In the calibrated case, epipolar geometry is encoded by the *essential matrix*, **E** according to:

$$\hat{\mathbf{x}}_1^T \mathbf{E} \hat{\mathbf{x}}_2 = 0$$

The 5-point algorithm

Recap from last week's lecture...

In the calibrated case, epipolar geometry is encoded by the *essential matrix*, **E** according to:

$$\hat{\mathbf{x}}_1^T \mathbf{E} \hat{\mathbf{x}}_2 = 0$$

In the calibrated setting there are just two possibilities if a plane is seen. See Negahdaripour, *Closed-form relationship between the two interpretations of a moving plane*. JOSA90

The 5-point algorithm

- **E** can be estimated from 5 corresponding points (see today's paper).
- A small sample is useful for RANSAC (le 3).
- The plane degeneracy is essentially avoided.
- There are however up to 10 solutions for **E** to test. Today's paper!

The 5-point algorithm

In lecture 4 we saw that:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} = \mathbf{R}\left[\mathbf{R}^T\mathbf{t}\right]_\times$$

...and how **R** and **t** (up to scale) can be retrieved from **E**, using the visibility constraint on a point correspondence.

Structure from Motion

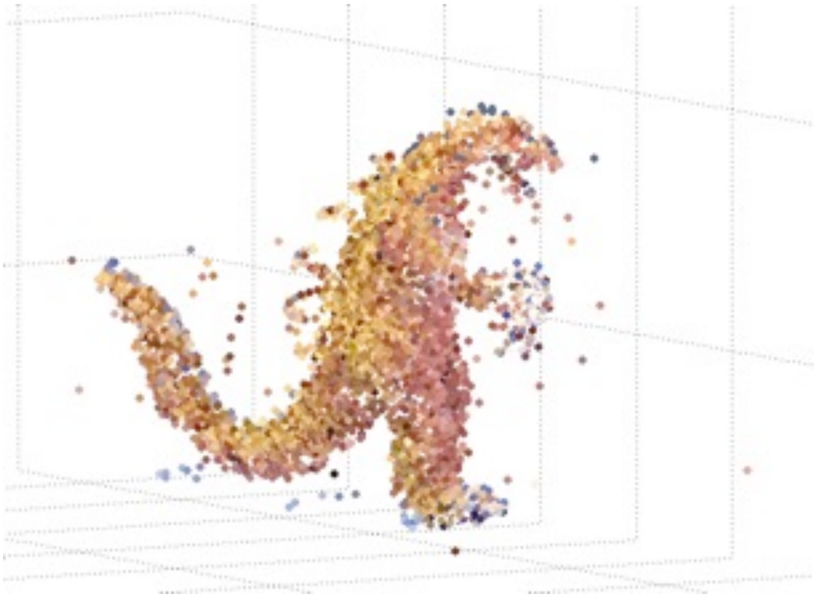
Estimation of the essential matrix is usually the first step in Structure from Motion (SfM)

Structure from Motion

Estimation of the essential matrix is usually the first step in Structure from Motion (SfM)

Input:

Output:



Structure from Motion

Definition:

Given a collection of images
depicting a static scene
compute the 3D scene structure
and the position of each camera (motion)

Structure from Motion

Cost function:

$$\varepsilon = \sum_{k=1}^{K} \sum_{l=1}^{L} v_{k,l} ||\mathbf{x}_{k,l} - \mathrm{proj}(\mathbf{R}_l(\mathbf{X}_k - \mathbf{t}_l))||^2$$

Structure from Motion

Definition of variables:

Given: $\mathbf{x}_{k,l} \quad \text{visible at} \quad v_{k,l}$

Sought: $\{\mathbf{X}_k\}_1^K, \{\mathbf{R}_l, \mathbf{t}_l\}_1^L$

By minimising: $\varepsilon(\{\mathbf{X}_k\}_1^K, \{\mathbf{R}_l, \mathbf{t}_l\}_1^L)$

Structure from Motion

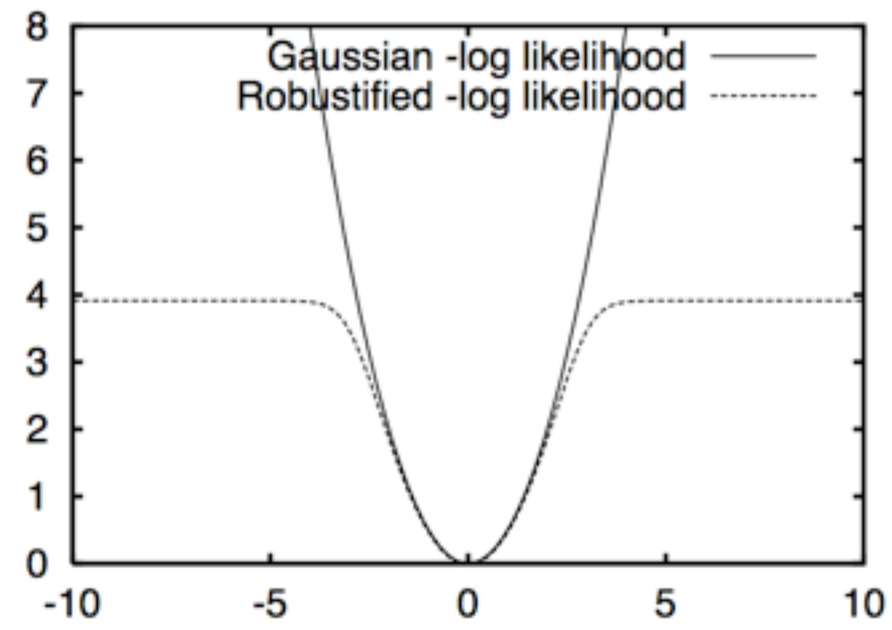
Cost function:

$$\varepsilon = \sum_{k=1}^{K} \sum_{l=1}^{L} v_{k,l} ||\mathbf{x}_{k,l} - \mathrm{proj}(\mathbf{R}_l(\mathbf{X}_k - \mathbf{t}_l))||^2$$

Structure from Motion

Robust cost function:

$$\varepsilon = \sum_{k=1}^{K}\sum_{l=1}^{L} v_{k,l}\,\rho(\mathbf{x}_{k,l} - \mathrm{proj}(\mathbf{R}_l(\mathbf{X}_k - \mathbf{t}_l)))$$



Structure from Motion

Challenges:

1. **Non-linear cost function**
   - least squares solution not possible
2. **Very large problem**
   - efficiency is paramount
3. **Non-convex problem**
   - many local minima

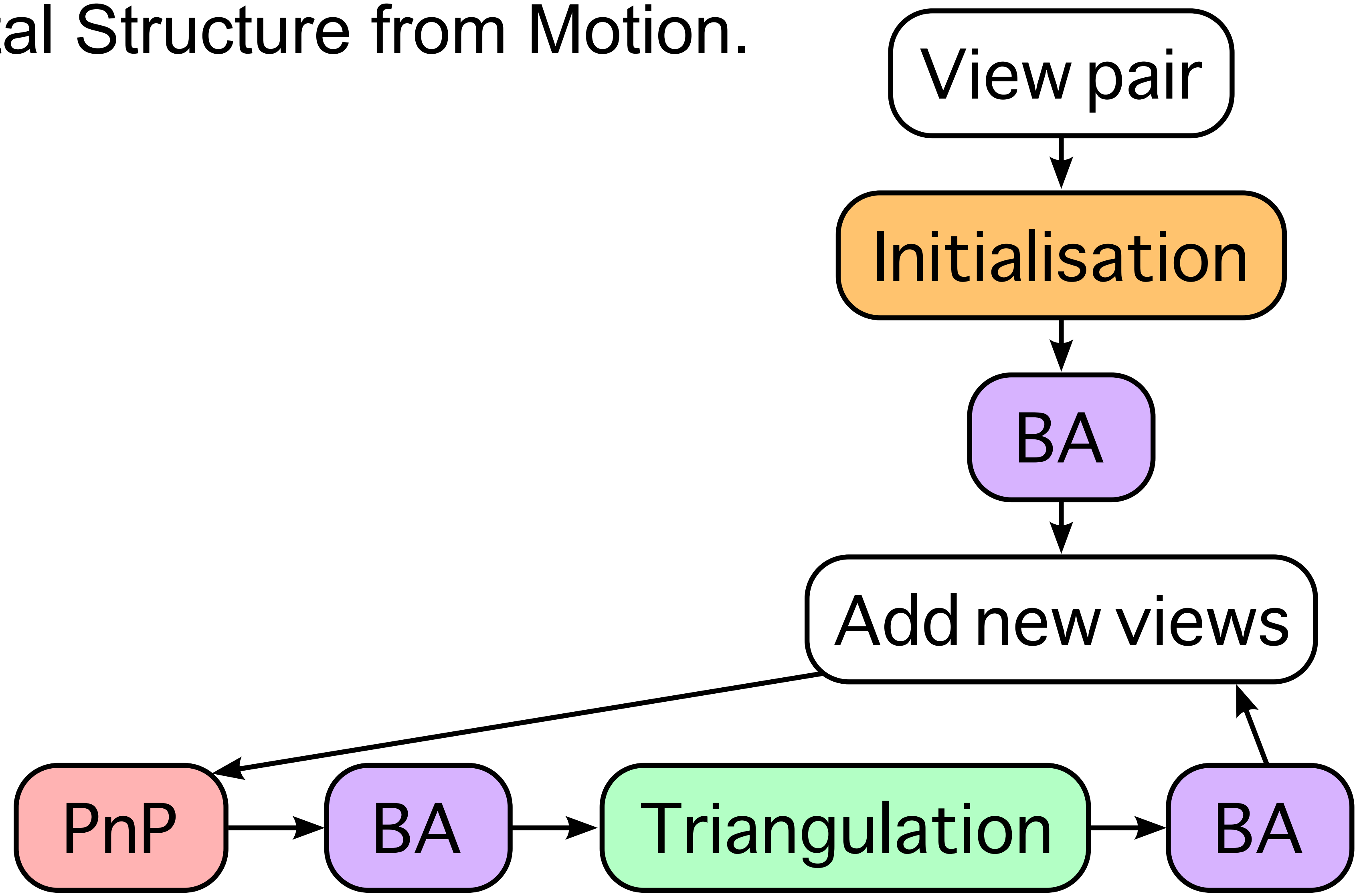
Structure from Motion

Typical solution:

1. Use an approximate method to find a solution close to the global min
2. Use a regularized Newton method (e.g. Levenberg-Marquardt) to refine the solution. This is called **Bundle Adjustment** (BA)

Incremental Structure from Motion

Incremental Structure from Motion.



Incremental Structure from Motion

Natural approach if the input is a video.

Used in many open source packages e.g.:

1. Bundler by Noah Snavely
   http://www.cs.cornell.edu/~snavely/bundler/
2. The Visual SFM package:
   http://ccwu.me/vsfm/

Structure from Motion

Parallel Incremental Bundle Adjustment.

(From an unordered image collection)

1. **Building Rome in a Day**, Agawal, Snavely, Simon, Seitz, Szeliski, ICCV 2009
2. **Building Rome on a Cloudless Day**, Frahm, Georgel, Gallup, Johnsson, Raguram, Wu, Yen, Dun, Clip, Lazebnik, Pollefeys, ECCV 2010

Rotation based SfM

Solve for rotation first [Martinec and Pajdla CVPR07]

1. Find Euclidean reconstructions from **pairs of views**.
2. Solve for all **absolute orientations**
3. Solve for **translations** with a reduced point set

Rotation based SfM

Solve for rotation first [Martinec and Pajdla CVPR07]

1. Find Euclidean reconstructions from **pairs of views**.

Results in $\mathbf{R}_{l,m}, \mathbf{t}_{l,m}\,, \quad l, m \in [1 \ldots L]$

2. Solve for all **absolute orientations**

$$\mathbf{R}_l\,, \quad l \in [1 \ldots L]$$

using:

$$\mathbf{R}_l(:, i) - \mathbf{R}_{l,m}\mathbf{R}_m(:, i) = \mathbf{0}\,, \quad i \in [1, 2, 3]$$

Rotation based SfM

2. Solve for all **absolute orientations**

$$\mathbf{R}_l \,, \quad l \in [1 \dots L]$$

using:

$$\mathbf{R}_l(:,i) - \mathbf{R}_{l,m}\mathbf{R}_m(:,i) = \mathbf{0} \,, \quad i \in [1,2,3]$$

This results in a large sparse linear system.

All $\mathbf{R}_l$ can be found from the three smallest eigenvectors to the system (orthogonality of $\mathbf{R}_l$ is enforced after estimation).

Rotation based SfM

2. Solve for all **absolute orientations**

Martinec and Pajdla used **eigs** in Matlab and this took 0.37 sec, to solve for 259 views, and 2049 relative orientations. (we've also tested this with similar results)

Rotation based SfM

3. Solve for **translations** with a reduced point set

Idea: look at $\mathbf{M} = [\mathbf{m}_1 \ \cdots \ ]$, where $\mathbf{m}_k = [\mathbf{R}_l | \mathbf{t}_l] \, \mathbf{X}_k$

and find **just four** representatives $\mathbf{X}_k$ that span **M**
(Matlab code provided in paper)

Rotation based SfM

3. Solve for **translations** with a reduced point set

Figure 4. Image pair 19-22 in the Raglan scene. Points satisfying EG of this image pair (top row). Non-mismatch candidates identified before the multiview registration (bottom left). The four points used for translation registration (bottom right).

Rotation based SfM

Martinec and Pajdla method timing:

46 frame example. 186131 3D points.

Full BA took 3h 6 min, max residual 98.57 pixels

Reduced BA took 4.68 sec, max residual 98.46 pixels

>2000x speedup (compared to using all points)

Rotation based SfM

Bonus feature: Better detection of incorrect EG.

Figure 1. A non-existent epipolar geometry (EG) raised by matching similar structures on different buildings in the Zwinger scene. The shown image pair 37-70 has 163 inliers which are 45% of all tentative matches. It would be extremely difficult to find out that this EG does not exist based on the two images only.

Rotation based SfM

Also extended by Enqvist, Kahl, and Olsson, **Non-Sequential Structure from Motion**, ICCV11 workshop

- Better detection of incorrect epipolar-geometries
- Translations are found using Using Second Order Cone Programming(SOCP). Auxiliary variables are used to be robust to outliers.

Why bundle adjustment?

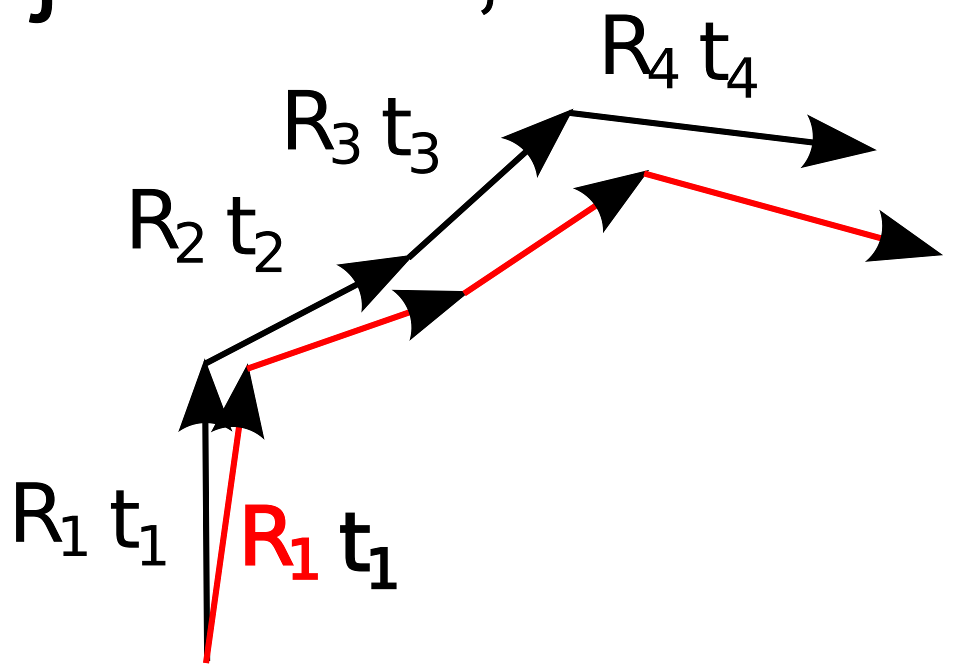
A decent 3D model can often be found by incrementally adding new cameras using PnP (or even using today's paper)

But...

Why bundle adjustment?

A decent 3D model can often be found by incrementally adding new cameras using PnP (or even using today's paper)

But for long trajectories, errors will start to accumulate.



Bundle Adjustment

BA is essentially **ML** over all image correspondences given all cameras, and all 3D points.

$$\{\mathbf{R}^*, \mathbf{t}^*, \mathbf{X}^*\} = \arg \min_{\{\mathbf{R},\mathbf{t},\mathbf{X}\}} \sum_{k,l} d(\mathbf{x}_{kl}, \mathbf{K}[\mathbf{R}_k|\mathbf{t}_k]\mathbf{X}_l)^2$$

Bundle Adjustment

BA is essentially **ML** over all image correspondences given all cameras, and all 3D points. (Optionally also intrinsics.)

$$\{\mathbf{R}^*, \mathbf{t}^*, \mathbf{X}^*\} = \arg \min_{\{\mathbf{R},\mathbf{t},\mathbf{X}\}} \sum_{k,l} d(\mathbf{x}_{kl}, \mathbf{K}[\mathbf{R}_k|\mathbf{t}_k]\mathbf{X}_l)^2$$

Needs initial guess. (Obtained by RANSAC on 5-point method and P3P)

Bundle Adjustment

The choice of **parametrisation** of 3D points, and camera rotations is important.

If both near and far points are seen, it might be better to use $\mathbf{X} = [X_1, X_2, X_3, X_4]^T$ than $\mathbf{X} = [X_1, X_2, X_3, 1]^T$

Good choices for rotations are unit quarternions, and axis-angle vectors (lecture 7)

Bundle Adjustment

Bundle adjustment cost function:

$$\varepsilon = \sum_{k=1}^{K} \sum_{l=1}^{L} v_{k,l} ||\mathbf{x}_{k,l} - \mathrm{proj}(\mathbf{R}_l(\mathbf{X}_k - \mathbf{t}_l))||^2$$

Bundle Adjustment

Bundle adjustment cost function:

$$\varepsilon = \sum_{k=1}^{K} \sum_{l=1}^{L} v_{k,l} ||\mathbf{x}_{k,l} - \mathrm{proj}(\mathbf{R}_l(\mathbf{X}_k - \mathbf{t}_l))||^2$$

New notation:

$$\varepsilon(\{\mathbf{R}_l, \mathbf{t}_l\}_1^L, \{\mathbf{X}_k\}_1^K) = \varepsilon(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

Bundle Adjustment

New notation for cost function...

$$\varepsilon(\{\mathbf{R}_l, \mathbf{t}_l\}_1^L, \{\mathbf{X}_k\}_1^K) = \varepsilon(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

Bundle Adjustment

New notation for cost function...

$$\varepsilon(\{\mathbf{R}_l, \mathbf{t}_l\}_1^L, \{\mathbf{X}_k\}_1^K) = \varepsilon(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

Taylor expansion...

$$\mathbf{r}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{r}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x}$$

Stationary point (set derivative of cost = 0)

$$\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x})\Delta\mathbf{x} = -\mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

Now solve for $\Delta\mathbf{x}$ ....

Bundle Adjustment

Levenberg-Marqardt:

begin

  $k := 0; \quad \nu := 2; \quad \mathbf{x} := \mathbf{x}_0$

  $\mathbf{A} := \mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}); \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$

  $found := (\|\mathbf{g}\|_\infty \le \varepsilon_1); \quad \mu := \tau * \max\{a_{ii}\}$

  while (**not** $found$) **and** $(k < k_{\max})$

    $k := k+1; \quad \text{Solve } (\mathbf{A} + \mu \mathbf{I})\mathbf{h}_{\text{lm}} = -\mathbf{g}$

    if $\|\mathbf{h}_{\text{lm}}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$

      $found :=$ **true**

    else

      $\mathbf{x}_{\text{new}} := \mathbf{x} + \mathbf{h}_{\text{lm}}$

      $\varrho := (F(\mathbf{x}) - F(\mathbf{x}_{\text{new}}))/(L(\mathbf{0}) - L(\mathbf{h}_{\text{lm}}))$

      if $\varrho > 0$ {step acceptable}

        $\mathbf{x} := \mathbf{x}_{\text{new}}$

        $\mathbf{A} := \mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}); \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$

        $found := (\|\mathbf{g}\|_\infty \le \varepsilon_1)$

        $\mu := \mu * \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}; \quad \nu := 2$

      else

        $\mu := \mu * \nu; \quad \nu := 2 * \nu$

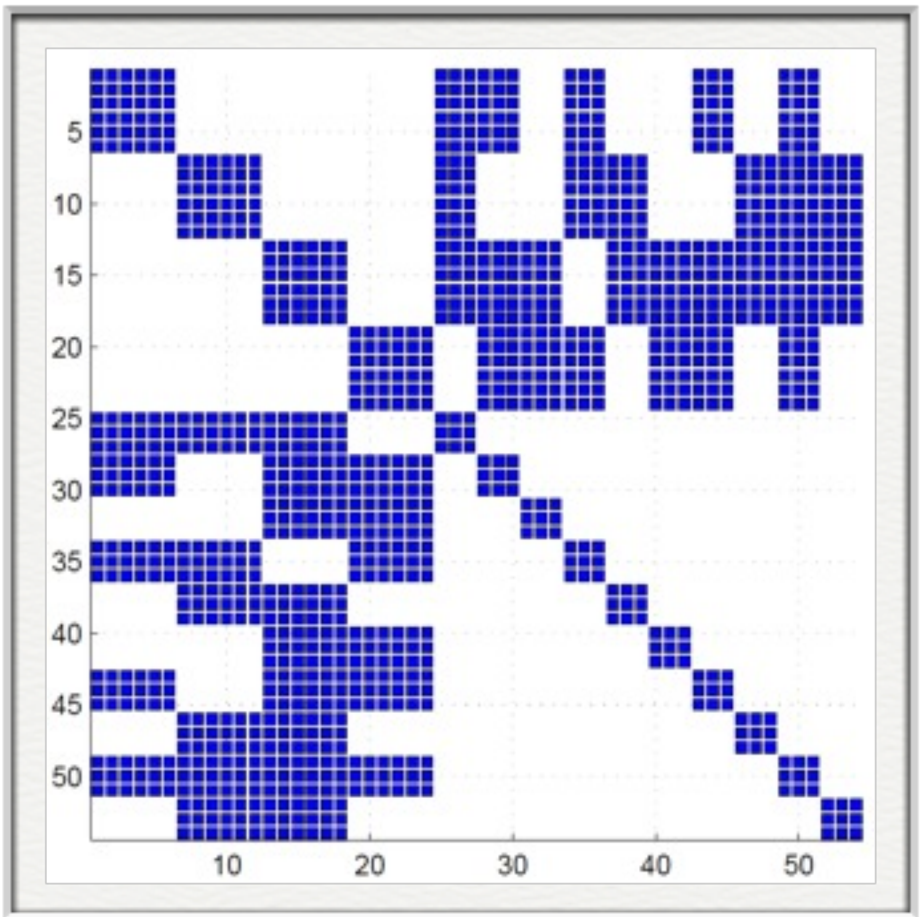
end

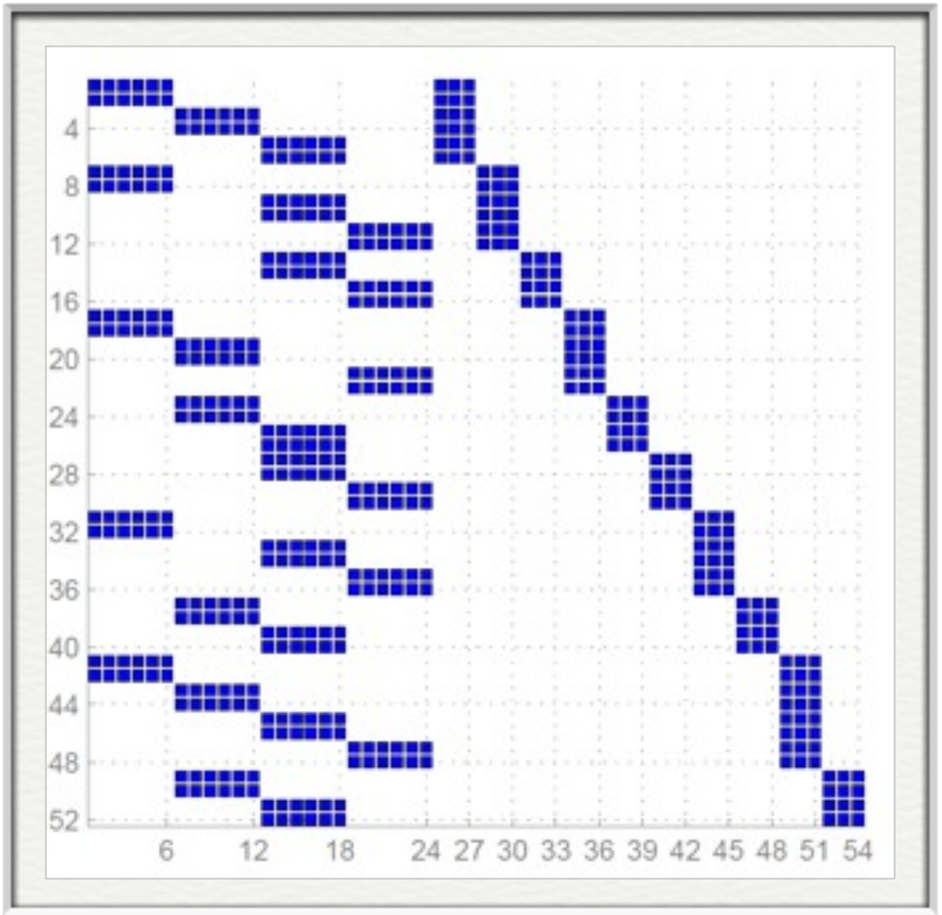
Solving the normal equations

Bundle Adjustment

Jacobian and approximate Hessian matrices:

$$\mathbf{J}^T\mathbf{J}\Delta\mathbf{x} = -\mathbf{J}^T\mathbf{r}(\mathbf{x})$$





Bundle Adjustment

Shur complement from text book:

$$(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^{\mathrm{T}}\mathbf{J}))\Delta\mathbf{x} = -\mathbf{J}^{\mathrm{T}}\mathbf{r}(\mathbf{x}_k)\,, \tag{7}$$

$$\begin{bmatrix} \mathbf{J}_c^{\mathrm{T}}\mathbf{J}_c & \mathbf{J}_c^{\mathrm{T}}\mathbf{J}_m \\ \mathbf{J}_m^{\mathrm{T}}\mathbf{J}_c & \mathbf{J}_m^{\mathrm{T}}\mathbf{J}_m \end{bmatrix} + \lambda\,\mathrm{diag}\begin{bmatrix} \mathbf{J}_c^{\mathrm{T}}\mathbf{J}_c & 0 \\ 0 & \mathbf{J}_m^{\mathrm{T}}\mathbf{J}_m \end{bmatrix} = \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{bmatrix}. \tag{8}$$

The normal equations (7) now read

$$\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{c} \\ \Delta\mathbf{m} \end{bmatrix} = -\begin{bmatrix} \mathbf{J}_c^{\mathrm{T}} \\ \mathbf{J}_m^{\mathrm{T}} \end{bmatrix} \mathbf{r}\,. \tag{9}$$

The camera parameter update can now be computed separately by elimination

$$(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}})\,\Delta\mathbf{c} = (\mathbf{W}\mathbf{V}^{-1}\mathbf{J}_m^{\mathrm{T}} - \mathbf{J}_c^{\mathrm{T}})\,\mathbf{r}\,. \tag{10}$$

Once we have the camera update, the update for the 3D points is obtained as:

$$\Delta\mathbf{m} = -\mathbf{V}^{-1}(\mathbf{J}_m^{\mathrm{T}}\mathbf{r} + \mathbf{W}^{\mathrm{T}}\Delta\mathbf{c})\,. \tag{11}$$

Bundle Adjustment

Comments:

1. To solve for the cameras, **Cholesky factorisation** is used instead of an explicit inverse.

2. For very large systems, **sparse Cholesky** solvers are preferable.

3. It quickly becomes impossible to store matrices explicitly, due to memory requirements (e.g. 200 cameras, 20K 3D points ➟ 30 TB for J^TJ).

Bundle Adjustment

Too many details to mention!

See the paper: Triggs et al., *Bundle Adjustment - A Modern Synthesis*, LNCS Book chapter, 2000

Discussion

Discussion of the papers:

1. David Nistér, *An Efficient Solution to the Five-Point Relative Pose Problem*, CVPR'03
2. Long Quan, *Invariants of six points and projective reconstruction from three uncalibrated images*, TPAMI'95