http://users.ics.forth.gr/~lourakis/sba/PRCV_colloq.pdf
lecture by Lourakis "Bundle adjustment gone public"

- Bundle Adjustment (BA) is a key ingredient of SaM, almost always used as its last step
  - • It is an optimization problem over the 3D structure and viewing parameters (camera pose, intrinsic calibration, & radial distortion parameters), which are simultaneously refined for minimizing reprojection error
  - very large nonlinear least squares problem, typically solved with the Levenberg-Marquardt (LM) algorithm
  - Std LM involves the repetitive solution of linear systems, each with O(N^3) time and O(N^2) storage complexity, resp.
    - • Example: for 54 cameras and 5207 3D points, N = 15945. ==> N^3 = 1e12
  - Sparse LM is a better solution.
  - Example:
    - M images
    - N features
    - **x_i_j** = measured feature "i" on image "j"
    - **a_j** = vector of parameters for camera "j"
    - **b_i** = vectors of parameters for point "i"
    - Q(aj , bi) = the predicted projection of point i on image j,
    - d(., .) the Euclidean distance between image points
    - vij = 1 iff point i is visible in image j
    - minimize reprojection error over a_j, b_i: min_aj, bi (summation_i=1_to_N(summation_j=1_to_M((v_i_j * d(Q(aj,bi), x_i_j))^2)))
      - ==> total number of parameters is M*(camera parameters) + N*(point parameters)
    - let **P** = parameter vector of camera then point parameters = [P_C P_P]
    - let **X_hat** = [(x_hat_1_1)^T   x_hat_1_2)^T … x_hat_1_M)^T   x_hat_2_1)^T … x_hat_N_M)^T]
    -    where ***x_hat_i_j = Q(a_j, b_i)*** is the projection onto camera plane
    - let error **eps** = [(eps_1_1)^T   eps_1_2)^T … eps_1_M)^T   eps_2_1)^T … eps_N_M)^T]
    -    where ***eps = x_i_j - x_hat_i_j***

http://users.ics.forth.gr/~lourakis/sba/PRCV_colloq.pdf

Bundle Adjustment (BA) becomes

min (summation_i=1_to_N(summation_j=1_to_M((eps_i_j)^2))) over P

Jacobian J = d(X_hat) / d(P) which has a block structure because of P being [camera parameters   point parameters]

J = [A | B] where A = d(X_hat) / d(a) and B = A = d(X_hat) / d(b)

The LM updating vector delta = [(delta(a))^T (delta(b))^T

The normal equations:

```
|  A^T * A     A^T * B |  *  | delta(a)) |  =  | A^T * eps |
| B^T * A      B^T * B |     | delta(b)) |     | B^T * eps |
```

The lhs matrix above is sparse due to A and B being sparse:

$\partial \hat{x}_{ij} \, \partial a_k = 0, \forall\, j \,!\!= k$ and

$\partial \hat{x}_{ij} \, \partial b_k = 0, \forall\, i \,!\!= k$

http://users.ics.forth.gr/~lourakis/sba/PRCV_colloq.pdf

<u>(example cont.) M images = 3, N features = 4</u>

$\mathbf{J} = \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{P}}$ has a block structure $[\,\mathbf{A}|\mathbf{B}\,]$,

Let $\mathbf{A}_{ij} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$ and $\mathbf{B}_{ij} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$

- The Jacobian J in block form:

$$
\frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{P}} =
\begin{array}{c}
 \\ \mathbf{x}_{11} \\ \mathbf{x}_{12} \\ \mathbf{x}_{13} \\ \mathbf{x}_{21} \\ \mathbf{x}_{22} \\ \mathbf{x}_{23} \\ \mathbf{x}_{31} \\ \mathbf{x}_{32} \\ \mathbf{x}_{33} \\ \mathbf{x}_{41} \\ \mathbf{x}_{42} \\ \mathbf{x}_{43}
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} \mathbf{a}_1{}^T & \mathbf{a}_2{}^T & \mathbf{a}_3{}^T & \mathbf{b}_1{}^T & \mathbf{b}_2{}^T & \mathbf{b}_3{}^T & \mathbf{b}_4{}^T \end{array} \\
\begin{pmatrix}
\mathbf{A}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{12} & \mathbf{0} & \mathbf{B}_{12} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{13} & \mathbf{B}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{A}_{21} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{21} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{22} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{23} & \mathbf{0} & \mathbf{B}_{23} & \mathbf{0} & \mathbf{0} \\
\mathbf{A}_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{31} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{32} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{32} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{33} & \mathbf{0} \\
\mathbf{A}_{41} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{41} \\
\mathbf{0} & \mathbf{A}_{42} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{42} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{43} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{43}
\end{pmatrix}
\end{array}
$$

(1)

This is the so-called *primary structure* of BA

- Approximate Hessian in block form:

$$
\mathbf{J}^T\mathbf{J} =
\begin{array}{c}
 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} \mathbf{a}_1{}^T & \mathbf{a}_2{}^T & \mathbf{a}_3{}^T & \mathbf{b}_1{}^T & \mathbf{b}_2{}^T & \mathbf{b}_3{}^T & \mathbf{b}_4{}^T \end{array} \\
\begin{pmatrix}
\mathbf{U}_1 & 0 & 0 & \mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} \\
0 & \mathbf{U}_2 & 0 & \mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} \\
0 & 0 & \mathbf{U}_3 & \mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} \\
\mathbf{W}_{11}^T & \mathbf{W}_{12}^T & \mathbf{W}_{13}^T & \mathbf{V}_1 & 0 & 0 & 0 \\
\mathbf{W}_{21}^T & \mathbf{W}_{22}^T & \mathbf{W}_{23}^T & 0 & \mathbf{V}_2 & 0 & 0 \\
\mathbf{W}_{31}^T & \mathbf{W}_{32}^T & \mathbf{W}_{33}^T & 0 & 0 & \mathbf{V}_3 & 0 \\
\mathbf{W}_{41}^T & \mathbf{W}_{42}^T & \mathbf{W}_{43}^T & 0 & 0 & 0 & \mathbf{V}_4
\end{pmatrix}
\end{array}
\equiv
$$

(2)

$$
\equiv \begin{pmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{pmatrix},
$$

$\mathbf{U}_j \equiv \sum_{i=1}^{4} \mathbf{A}_{ij}^T \mathbf{A}_{ij}$, <u>**for 1 image, summing over all features**</u>

$\mathbf{V}_i \equiv \sum_{j=1}^{3} \mathbf{B}_{ij}^T \mathbf{B}_{ij}$, <u>**for 1 feature, summing over all images**</u>

$\mathbf{W}_{ij} = \mathbf{A}_{ij}^T \mathbf{B}_{ij}$

(example cont.) M images = 3, N features = 4

Bundle Adjustment Revisited

Yu Chen1, Yisong Chen1, Guoping Wang1
1 Peking University, Department of Computer Science and Technology,
Graphics and Interactive Lab, Beijing, China

For convenience, we use (18) to show how to solve the bundle adjustment problem. set $\hat{u}_{ij} = \pi(C_j, X_i)$, and we order the parameter x into camera block c and structure block p:

$$x = [c, \; p] \tag{20}$$

it's easily to realize that:

$$J_{ij} = \frac{\partial r_{ij}}{\partial x_k} = \frac{\partial \hat{u}_{ij}}{\partial x_k}, \frac{\partial \hat{u}_{ij}}{\partial c_k} \underline{= 0, \forall j \neq k}, \frac{\partial \hat{u}_{ij}}{\partial p_k} \underline{= 0, \forall i \neq k} \tag{21}$$

Consider now, that we have $m = 3$ cameras and $n = 4$ 3D points. Set $A_{ij} = \frac{\partial \hat{u_{ij}}}{\partial c_j}, B_{ij} = \frac{\partial \hat{u_{ij}}}{\partial p_i}$, we can obtain the Jacobi:

k dimension is row number w.r.t. the block of i where i is the feature number

$$J = \frac{\partial \hat{u}}{\partial x} = \begin{array}{c} \\ k = 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ k = 4 \\ \\ \end{array} \begin{bmatrix} A_{11} & 0 & 0 & B_{11} & 0 & 0 & 0 \\ 0 & A_{12} & 0 & B_{12} & 0 & 0 & 0 \\ 0 & 0 & A_{13} & B_{13} & 0 & 0 & 0 \\ A_{21} & 0 & 0 & 0 & B_{21} & 0 & 0 \\ 0 & A_{22} & 0 & 0 & B_{22} & 0 & 0 \\ 0 & 0 & A_{23} & 0 & B_{23} & 0 & 0 \\ A_{31} & 0 & 0 & 0 & 0 & B_{31} & 0 \\ 0 & A_{32} & 0 & 0 & 0 & B_{32} & 0 \\ 0 & 0 & A_{33} & 0 & 0 & B_{33} & 0 \\ A_{41} & 0 & 0 & 0 & 0 & 0 & B_{41} \\ 0 & A_{42} & 0 & 0 & 0 & 0 & B_{42} \\ 0 & 0 & A_{43} & 0 & 0 & 0 & B_{43} \end{bmatrix} \tag{22}$$

http://users.ics.forth.gr/~lourakis/sba/PRCV_colloq.pdf

<u>(example cont.) M images = 3, N features = 4</u>

```
NOTE: eps_a = A^T * eps, eps_b =  B^T * eps
```

- The augmented normal equations $(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\delta_\mathbf{p} = \mathbf{J}^T\epsilon$ take the form

where μ > 0

(3) $$\begin{pmatrix} \mathbf{U}^* & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}^* \end{pmatrix} \begin{pmatrix} \delta_\mathbf{a} \\ \delta_\mathbf{b} \end{pmatrix} = \begin{pmatrix} \epsilon_\mathbf{a} \\ \epsilon_\mathbf{b} \end{pmatrix}$$

- Performing block Gaussian elimination in the lhs matrix, $\delta_\mathbf{a}$ is determined with Cholesky from $\mathbf{V}^*$'s Schur complement:

(4) $$(\mathbf{U}^* - \mathbf{W}\,\mathbf{V}^{*-1}\,\mathbf{W}^T)\,\delta_\mathbf{a} = \epsilon_\mathbf{a} - \mathbf{W}\,\mathbf{V}^{*-1}\,\epsilon_\mathbf{b}$$

Schur complement: multiply 1st matrix on res by

```
  | I                          0 |
  | (((-V*)^-1)W^T     I |
resulting in:
| (U*) - W(((V*)^-1)W^T   W   | *  | delta_a | = | eps_a |
| 0                                (V*) |     | delta_b |    | eps_b |
```

$$\mathbf{V}^{*-1} = \begin{pmatrix} \mathbf{V}_1^{*-1} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{V}_2^{*-1} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

note (V*) is invertible and only the block diagonals are populated, so each V_i is inverted.

separate delta_b: 0*delta_a + (V*) * delta_b = eps_b ==> delta_b = eps_b * ((V*)^-1)
solving for delta_a (typically M images << N features) after substitute delta_b:
((U*) - W(((V*)^-1)W^T) * delta_a + W * delta_b = eps_a
((U*) - W(((V*)^-1)W^T) * delta_a = eps_a - W((V*)^-1)eps_b

NOTE: (U*) - W(((V*)^-1)W^T is called the reduced camera matrix (because delta_a is camera parameters)

http://users.ics.forth.gr/~lourakis/sba/PRCV_colloq.pdf

RCM (reduced camera matrix) is sparse because not all features appear in all cameras.
this is known as ***secondary structure*.**

For very large datasets, RCM tends to be in one of two classes:
(1) ○ visual mapping: extended areas are traversed, limited image overlap (sparse RCM)
(2) ○ centered-object: a large number of overlapping images taken in a small area (dense RCM)

Solving for delta_a in the equation containing RCM. several ways:
(1) Store as dense, decompose with **ordinary linear algebra** ○[M. Lourakis, A. Argyros: SBA: A Software Package For Generic Sparse Bundle Adjustment. ACM Trans. Math. Softw. 36(1): (2009) ○ C. Engels, H. Stewenius, D. Nister: Bundle Adjustment Rules. Photogrammetric Computer Vision (PCV), 2006.
(2) • Store as sparse, factorize with **sparse direct solvers** ○ K. Konolige: Sparse Sparse Bundle Adjustment. BMVC 2010: 1-11
(3) Store as sparse, use **conjugate gradient methods** memory efficient, iterative, precoditioners necessary! ○ S. Agarwal, N. Snavely, S.M. Seitz, R. Szeliski: Bundle Adjustment in the Large. ECCV (2) 2010: 29-42 ○ M. Byrod, K. Astrom: Conjugate Gradient Bundle Adjustment. ECCV (2) 2010: 114-127
(4) • Avoid storing altogether ○ C. Wu, S. Agarwal, B. Curless, S.M. Seitz: Multicore Bundle Adjustment. CVPR 2011: 30 57-3064 ○ M. Lourakis: Sparse Non-linear Least Squares Optimization for Geometric Vision. ECCV (2) 2010: 43-56

Engels, Stewenius, Nister 2006, "Bundle Adjustment Rules"

m images (= video frames from same calibrated camera)
? features
each feature **x** has **M** dimensions
n iterations of bundle adjustment over the last m video frames

Engels "RCM" is formed from a jacobian which places point parameters before camera parameters,
so is different than that in the Lourakis notes.

$$J_f = \begin{bmatrix} J_P & J_C \end{bmatrix}, \tag{15}$$

$$H = \begin{bmatrix} J_P^\top J_P & J_P^\top J_C \\ J_C^\top J_P & J_C^\top J_C \end{bmatrix}, \tag{16}$$

$$\begin{bmatrix} H_{PP} & H_{PC} \\ H_{PC}^\top & H_{CC} \end{bmatrix} \begin{bmatrix} dP \\ dC \end{bmatrix} = \begin{bmatrix} b_P \\ b_C \end{bmatrix}, \tag{17}$$

where we have defined $H_{PP} = J_P^\top J_P$, $H_{PC} = J_P^\top J_C$, $H_{CC} = J_C^\top J_C$, $b_P = -J_P^\top f$, $b_C = -J_C^\top f$ to simpify the notation, and dP and dC represent the update of the point parameters and the camera parameters, respectively. Note that the matrices H_{PP} and H_{CC} are block-diagonal, where the blocks correspond to

of as multiplying by

$$\begin{bmatrix} I & 0 \\ -H_{PC}^\top & I \end{bmatrix} \tag{20}$$

from the left on both sides, resulting in the smaller equation system (from the lower part)

$$\underbrace{(H_{CC} - H_{PC}^\top H_{PP}^{-1} H_{PC})}_{A}\, dC = \underbrace{b_C - H_{PC}^\top H_{PP}^{-1} b_P}_{B} \tag{21}$$

for the camera parameter update dC. For very large systems,

We use straightforward Cholesky factorization.

Engels, Stewenius, Nister 2006, "Bundle Adjustment Rules"

M images = 3, j
N features = 4, i

their "RCM" is formed from a jacobian which places point parameters before camera parameters, so is different than that in the Lourakis notes.

J_P is [n*m*[1X3] X n] is [n*m X 3*n]

J_C is [n*m*[1X9] X m] is [n*m X 9*m]

J is [(n*mX1) X (3Xn + 9Xm)] is [n*m X (3*n + 9*m)]

Lourakis

Let $\underset{\boxed{1X9}}{\mathbf{A}_{ij}} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$ **C** and $\underset{\boxed{1X3}}{\mathbf{B}_{ij}} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$ **P**

$$\mathbf{J} = \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{P}} = \left[\frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{a}} \; \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{b}} \right] = \left[\; J_C \quad J_P \; \right]$$

- The Jacobian J in block form:

$$\frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{P}} = \begin{array}{c} \\ \mathbf{x}_{11} \\ \mathbf{x}_{12} \\ \mathbf{x}_{13} \\ \mathbf{x}_{21} \\ \mathbf{x}_{22} \\ \mathbf{x}_{23} \\ \mathbf{x}_{31} \\ \mathbf{x}_{32} \\ \mathbf{x}_{33} \\ \mathbf{x}_{41} \\ \mathbf{x}_{42} \\ \mathbf{x}_{43} \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathbf{a}_1{}^T & \mathbf{a}_2{}^T & \mathbf{a}_3{}^T & \mathbf{b}_1{}^T & \mathbf{b}_2{}^T & \mathbf{b}_3{}^T & \mathbf{b}_4{}^T \end{array} \\ \begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{12} & \mathbf{0} & \mathbf{B}_{12} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{13} & \mathbf{B}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{21} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{21} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{22} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{23} & \mathbf{0} & \mathbf{B}_{23} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{31} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{32} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{32} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{33} & \mathbf{0} \\ \mathbf{A}_{41} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{41} \\ \mathbf{0} & \mathbf{A}_{42} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{42} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{43} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{43} \end{pmatrix} \end{array} \quad (1)$$

Engels

Let $\underset{\boxed{1X9}}{\mathbf{A}_{ij}} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$ **C** and $\underset{\boxed{1X3}}{\mathbf{B}_{ij}} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$ **P**

$$J_f = \left[\; J_P \quad J_C \; \right],$$

Qu: [2X3] etc

- The Jacobian J in block form:

[mn X (9m + 3m)]

Qu: [2*mn X (9m + 3m)]

$$\frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{P}} = \begin{array}{c} \\ \mathbf{x}_{11} \\ \mathbf{x}_{12} \\ \mathbf{x}_{13} \\ \mathbf{x}_{21} \\ \mathbf{x}_{22} \\ \mathbf{x}_{23} \\ \mathbf{x}_{31} \\ \mathbf{x}_{32} \\ \mathbf{x}_{33} \\ \mathbf{x}_{41} \\ \mathbf{x}_{42} \\ \mathbf{x}_{43} \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathbf{b}_1{}^T & \mathbf{b}_2{}^T & \mathbf{b}_3{}^T & \mathbf{b}_4{}^T & \mathbf{a}_1{}^T & \mathbf{a}_2{}^T & \mathbf{a}_3{}^T \end{array} \\ \begin{pmatrix} \mathbf{B}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{B}_{12} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{12} & \mathbf{0} \\ \mathbf{B}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{B}_{21} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{21} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{23} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{23} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_{31} & \mathbf{0} & \mathbf{A}_{31} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_{32} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{32} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{33} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{41} & \mathbf{A}_{41} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{42} & \mathbf{0} & \mathbf{A}_{42} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{43} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{43} \end{pmatrix} \end{array} \quad (1)$$

J is [n*m X (3*n + 9*m)]

J^T * J is [(3*n + 9*m)] X (3*n + 9*m)]

Engels, Stewenius, Nister 2006, "Bundle Adjustment Rules"

M images = 3, j

N features = 4, i

their "RCM" is formed from a jacobian which places point parameters before camera parameters, so is different than that in the Lourakis notes.

Lourakis

Let $\mathbf{A}_{ij} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$ and $\mathbf{B}_{ij} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$

C **P**

$$\mathbf{J} = \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{P}} = \left[\frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{a}} \; \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{b}} \right] = [\; J_C \quad J_P \;]$$

J^T * J is [(3*n + 9*m)] X (3*n + 9*m)]

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

$$\begin{array}{c} \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} \mathbf{a}_1{}^T & \mathbf{a}_2{}^T & \mathbf{a}_3{}^T & \mathbf{b}_1{}^T & \mathbf{b}_2{}^T & \mathbf{b}_3{}^T & \mathbf{b}_4{}^T \end{array} \\
\begin{pmatrix}
\mathbf{U}_1 & 0 & 0 & \mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} \\
0 & \mathbf{U}_2 & 0 & \mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} \\
0 & 0 & \mathbf{U}_3 & \mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} \\
\mathbf{W}_{11}^T & \mathbf{W}_{12}^T & \mathbf{W}_{13}^T & \mathbf{V}_1 & 0 & 0 & 0 \\
\mathbf{W}_{21}^T & \mathbf{W}_{22}^T & \mathbf{W}_{23}^T & 0 & \mathbf{V}_2 & 0 & 0 \\
\mathbf{W}_{31}^T & \mathbf{W}_{32}^T & \mathbf{W}_{33}^T & 0 & 0 & \mathbf{V}_3 & 0 \\
\mathbf{W}_{41}^T & \mathbf{W}_{42}^T & \mathbf{W}_{43}^T & 0 & 0 & 0 & \mathbf{V}_4
\end{pmatrix}
\end{array}$$

Engels

Let $\mathbf{A}_{ij} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$ and $\mathbf{B}_{ij} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$

1X9 **C** 1X3 **P**

$$\mathbf{J} = \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{P}} = \left[\frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{b}} \; \frac{\partial \hat{\mathbf{X}}}{\partial \mathbf{a}} \right] = [\; J_P \quad J_C \;],$$

J^T * J is [(3*n + 9*m)] X (3*n + 9*m)]

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

$$\begin{array}{c} \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} \mathbf{b}_1{}^T & \mathbf{b}_2{}^T & \mathbf{b}_3{}^T & \mathbf{b}_4{}^T & \mathbf{a}_1{}^T & \mathbf{a}_2{}^T & \mathbf{a}_3{}^T \end{array} \\
\begin{pmatrix}
\mathbf{V}_1 & 0 & 0 & 0 & \mathbf{W}_{11}^T & \mathbf{W}_{12}^T & \mathbf{W}_{13}^T \\
0 & \mathbf{V}_2 & 0 & 0 & \mathbf{W}_{21}^T & \mathbf{W}_{22}^T & \mathbf{W}_{23}^T \\
0 & 0 & \mathbf{V}_3 & 0 & \mathbf{W}_{31}^T & \mathbf{W}_{32}^T & \mathbf{W}_{33}^T \\
0 & 0 & 0 & \mathbf{V}_4 & \mathbf{W}_{41}^T & \mathbf{W}_{42}^T & \mathbf{W}_{43}^T \\
\mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} & \mathbf{U}_1 & 0 & 0 \\
\mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} & 0 & \mathbf{U}_2 & 0 \\
\mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} & 0 & 0 & \mathbf{U}_3
\end{pmatrix}
\end{array}$$

where

$\mathbf{U}_j \equiv \sum_{i=1}^{4} \mathbf{A}_{ij}^T \mathbf{A}_{ij}$, **<u>for 1 image, sum over features</u>** 9X9

$\mathbf{V}_i \equiv \sum_{j=1}^{3} \mathbf{B}_{ij}^T \mathbf{B}_{ij}$, **<u>for 1 feature, sum over images</u>** 3X3 $\equiv \sum_{j=1}^{3} \left(\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}\right)^T \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$

$\mathbf{W}_{ij} = \mathbf{A}_{ij}^T \mathbf{B}_{ij}$ 9X3

Engels, Stewenius, Nister 2006, “Bundle Adjustment Rules”

Note that $\mathbf{V}^{*-1} = \begin{pmatrix} \mathbf{V}_1^{*-1} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{V}_2^{*-1} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$

$\delta_\mathbf{b}$ is dP is [3*n_features X 1]
$\epsilon_\mathbf{b}$ is b_P is [3*n_features X 1]
$\delta_\mathbf{a}$ is dC is [9*m_images X 1]
$\epsilon_\mathbf{a}$ is b_C is [9*m_images X 1]

V* is [3*n X 3*n]

W is [9*m X 3*n]

U* is [9*m X 9*m]

Lourakis

The augmented normal equations $(\mathbf{J}^T\mathbf{J}+\mu\mathbf{I})\delta_\mathbf{p} = \mathbf{J}^T\epsilon$ take the form

(3) $$\begin{pmatrix} \mathbf{U}^* & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}^* \end{pmatrix} \begin{pmatrix} \delta_\mathbf{a} \\ \delta_\mathbf{b} \end{pmatrix} = \begin{pmatrix} \epsilon_\mathbf{a} \\ \epsilon_\mathbf{b} \end{pmatrix}$$

$$\begin{bmatrix} U - WV^{-1}W^T & 0 \end{bmatrix} \begin{bmatrix} \delta_\mathbf{a} \\ \delta_\mathbf{b} \end{bmatrix} = \begin{bmatrix} I & -WV^{-1} \end{bmatrix} \begin{bmatrix} \epsilon_\mathbf{a} \\ \epsilon_\mathbf{b} \end{bmatrix}$$

(solve delta a first because typically m_images << n_features)

determine $\delta_\mathbf{a}$ with Cholesky **(or other method)**

$$(\mathbf{U}^* - \mathbf{W}\ \mathbf{V}^{*-1}\ \mathbf{W}^T)\ \delta_\mathbf{a} = \epsilon_\mathbf{a} - \mathbf{W}\ \mathbf{V}^{*-1}\ \epsilon_\mathbf{b}$$

$\delta_\mathbf{b}$ can be computed by back substitution into

$$\mathbf{V}^*\ \delta_\mathbf{b} = \epsilon_\mathbf{b} - \mathbf{W}^T\ \delta_\mathbf{a}$$

$$\delta_\mathbf{b} = \mathbf{V}^{*-1}\ \epsilon_\mathbf{b} - \mathbf{V}^{*-1}\ \mathbf{W}^T\ \delta_\mathbf{a}$$

Engels

$$\begin{pmatrix} \mathbf{V}^* & \mathbf{W}^T \\ \mathbf{W} & \mathbf{U}^* \end{pmatrix} \begin{pmatrix} \delta_\mathbf{b} \\ \delta_\mathbf{a} \end{pmatrix} = \begin{pmatrix} \epsilon_\mathbf{b} \\ \epsilon_\mathbf{a} \end{pmatrix}$$

$$\begin{bmatrix} H_{PP} & H_{PC} \\ H_{PC}^\top & H_{CC} \end{bmatrix} \begin{bmatrix} dP \\ dC \end{bmatrix} = \begin{bmatrix} b_P \\ b_C \end{bmatrix},$$

$$\begin{aligned} H_{PP} &= J_P^\top J_P \\ H_{PC} &= J_P^\top J_C, \\ H_{CC} &= J_C^\top J_C, \\ b_P &= -J_P^\top f, \\ b_C &= -J_C^\top f \end{aligned}$$

$$(\mathbf{U}^* - \mathbf{W}\ \mathbf{V}^{*-1}\ \mathbf{W}^T)\ \delta_\mathbf{a} = \epsilon_\mathbf{a} - \mathbf{W}\ \mathbf{V}^{*-1}\ \epsilon_\mathbf{b}$$

$$\underbrace{(H_{CC} - H_{PC}^\top H_{PP}^{-1} H_{PC})}_{A}\, dC = \underbrace{b_C - H_{PC}^\top H_{PP}^{-1} b_P}_{B}$$

$\delta_\mathbf{b}$ can be computed by back substitution into

$$\mathbf{V}^*\ \delta_\mathbf{b} = \epsilon_\mathbf{b} - \mathbf{W}^T\ \delta_\mathbf{a}$$

$$H_{PP}\ dP = b_P - H_{PC}\ dC$$

$$dP = H_{PP}^{-1} b_P - H_{PP}^{-1} H_{PC} dC.$$

Qu

$$\begin{pmatrix} \boldsymbol{B} & \boldsymbol{E} \\ \boldsymbol{E}^T & \boldsymbol{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{p_c} \\ \boldsymbol{p_p} \end{pmatrix} = - \begin{pmatrix} \boldsymbol{g_c} \\ \boldsymbol{g_p} \end{pmatrix} \tag{3.61}$$

see eqn (3.70) too

$$-\boldsymbol{g}^k = -\boldsymbol{J}^{k^T}\boldsymbol{F}^k$$

where k is a feature block summed over all images?

$$(\boldsymbol{B} - \boldsymbol{E}\boldsymbol{C}^{-1}\boldsymbol{E}^T)\boldsymbol{p_c} = -\boldsymbol{g_c} + \boldsymbol{E}\boldsymbol{C}^{-1}\boldsymbol{g_p}$$

$$\boldsymbol{p_p} = \boldsymbol{C}^{-1}(-\boldsymbol{g_p} - \boldsymbol{E}^T\boldsymbol{p_c})$$

Engels, et al 2006

M images = 3, j

N features = 4, i

$$
\begin{aligned}
H_{PP} &= J_P^\top J_P && \equiv \mathbf{V}^* \\
H_{PC} &= J_P^\top J_C, && \equiv \mathbf{W}^T \\
H_{CC} &= J_C^\top J_C, && \equiv \mathbf{U}^* \\
b_P &= -J_P^\top f, && \equiv \epsilon_\mathbf{b} \\
b_C &= -J_C^\top f && \equiv \epsilon_\mathbf{a}
\end{aligned}
$$

Let $\underset{\boxed{1X9}}{\mathbf{A}_{ij}} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$ **C** and $\underset{\boxed{1X3}}{\mathbf{B}_{ij}} = \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$ **P**

Note that $\mathbf{V}^{*-1} = \begin{pmatrix} \mathbf{V}_1^{*-1} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{V}_2^{*-1} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$

$$(\mathbf{U}^* - \mathbf{W}\,\mathbf{V}^{*-1}\,\mathbf{W}^T)\,\delta_\mathbf{a} = \epsilon_\mathbf{a} - \mathbf{W}\,\mathbf{V}^{*-1}\,\epsilon_\mathbf{b}$$

$$\underbrace{(H_{CC} - H_{PC}^\top H_{PP}^{-1} H_{PC})}_{A}\,dC = \underbrace{b_C - H_{PC}^\top H_{PP}^{-1} b_P}_{B}$$

$$\mathbf{J}^T\mathbf{J} =
\begin{array}{c|c}
 & \mathbf{b}_1^T \quad \mathbf{b}_2^T \quad \mathbf{b}_3^T \quad \mathbf{b}_4^T \quad \mathbf{a}_1^T \quad \mathbf{a}_2^T \quad \mathbf{a}_3^T \\
\begin{matrix}\mathbf{b}_1\\ \mathbf{b}_2\\ \mathbf{b}_3\\ \mathbf{b}_4\\ \mathbf{a}_1\\ \mathbf{a}_2\\ \mathbf{a}_3\end{matrix} &
\begin{pmatrix}
\mathbf{V}_1 & 0 & 0 & 0 & \mathbf{W}_{11}^T & \mathbf{W}_{12}^T & \mathbf{W}_{13}^T \\
0 & \mathbf{V}_2 & 0 & 0 & \mathbf{W}_{21}^T & \mathbf{W}_{22}^T & \mathbf{W}_{23}^T \\
0 & 0 & \mathbf{V}_3 & 0 & \mathbf{W}_{31}^T & \mathbf{W}_{32}^T & \mathbf{W}_{33}^T \\
0 & 0 & 0 & \mathbf{V}_4 & \mathbf{W}_{41}^T & \mathbf{W}_{42}^T & \mathbf{W}_{43}^T \\
\mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} & \mathbf{U}_1 & 0 & 0 \\
\mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} & 0 & \mathbf{U}_2 & 0 \\
\mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} & 0 & 0 & \mathbf{U}_3
\end{pmatrix}
\end{array}$$

$\mathbf{U}_j \equiv \sum_{i=1}^{4} \mathbf{A}_{ij}^T \mathbf{A}_{ij},$ **[9X9], for 1 image, sum over features**

$\mathbf{V}_i \equiv \sum_{j=1}^{3} \mathbf{B}_{ij}^T \mathbf{B}_{ij},$ **[3X3], for 1 feature, sum over images**

$\mathbf{W}_{ij} = \mathbf{A}_{ij}^T \mathbf{B}_{ij}$ **[9X3]**

V* is [3*n X 3*n]

W is [9*m X 3*n]

U* is [9*m X 9*m]

$$\begin{pmatrix} \mathbf{V}^* & \mathbf{W}^T \\ \mathbf{W} & \mathbf{U}^* \end{pmatrix} \begin{pmatrix} \delta_\mathbf{b} \\ \delta_\mathbf{a} \end{pmatrix} = \begin{pmatrix} \epsilon_\mathbf{b} \\ \epsilon_\mathbf{a} \end{pmatrix}$$

1 Initialize λ.

2 **Compute cost function** at initial camera and point configuration.

3 Clear the left hand side matrix A and right hand side vector B.

4 For each track p **(p is feature i of N)**

{

Clear a variable H_{pp} to represent block p of H_{PP} (in our case a symmetric 3×3 matrix) and a variable b_p to represent part p of b_P (in our case a 3-vector). (H_{pp}: $\mathbf{V}^*$, H_{PP}: $\mathbf{V}^*$, b_p: $\epsilon_\mathbf{b}$, b_P: $\epsilon_\mathbf{b}$)

(Compute derivatives) For each camera c on track p **(c is image j of M)**

{ Compute error vector f of reprojection in camera c of point p and its Jacobians J_p ($\mathbf{B}$) and J_c ($\mathbf{A}$) with respect to the point parameters (in our case a 2×3 matrix) and the camera parameters (in our case a 2×6 matrix), respectively.

Add $J_p^\top J_p$ ($\mathbf{B}^T\mathbf{B}$) to the upper triangular part of H_{pp} ($\mathbf{V}^*$).

Subtract $J_p^\top f$ ($\mathbf{B}^T$) from b_p ($\epsilon_\mathbf{b}$).

If camera c is free

{ Add $J_c^\top J_c$ ($\mathbf{A}^T\mathbf{A}$) (optionally with an augmented diagonal) upper triangular part of block (c, c) of left hand si matrix A (in our case a 6×6 matrix).

Compute block (p, c) of H_{PC} as $H_{pc} = J_p^\top J_c$ (in our case a 3×6 matrix) and store it until track is done.

Subtract $J_c^\top f$ from part c of right hand side vector B (related to b_C).

}

}// **end c loop**

Augment diagonal of H_{pp}, which is now accumulated and ready. Invert H_{pp}, taking advantage of the fact that it is a symmetric matrix.

Compute $H_{pp}^{-1} b_p$ and store it in a variable t_p.

(Outer product of track) For each free camera c on track p

{

Subtract $H_{pc}^\top t_p = H_{pc}^\top H_{pp}^{-1} b_p$ from part c of right hand side vector B.

Compute the matrix $H_{pc}^\top H_{pp}^{-1}$ and store it in a variable T_{pc}

For each free camera $c2 \geq c$ on track p

{

Subtract $T_{pc} H_{pc2} = H_{pc}^\top H_{pp}^{-1} H_{pc2}$ from block $(c, c2)$ of left hand side matrix A.

} }

}// **end p loop**

$$\overset{\textbf{n blocks}}{\overbrace{}}\;
\mathbf{V}^* = \text{n blocks}\left\{\begin{matrix}
\underset{[3X3]}{\mathbf{V}_1} & 0 & 0 & 0 \\
0 & \underset{[3X3]}{\mathbf{V}_2} & 0 & 0 \\
0 & 0 & \underset{[3X3]}{\mathbf{V}_3} & 0 \\
0 & 0 & 0 & \underset{[3X3]}{\mathbf{V}_4}
\end{matrix}\right. \qquad
\delta_\mathbf{b} = \begin{matrix} [x0,y0,z0]^T \\ [x1,y1,z1]^T \\ [x2,y2,z2]^T \\ [x3,y3,z3]^T \end{matrix} \quad \textbf{[3*nX1]}$$

$$\mathbf{U}^* = \text{m blocks}\left\{\begin{matrix}
\underset{[9X9]}{\mathbf{U}_1} & 0 & 0 \\
0 & \underset{[9X9]}{\mathbf{U}_2} & 0 \\
0 & 0 & \underset{[9X9]}{\mathbf{U}_3}
\end{matrix}\right. \qquad
\delta_\mathbf{a} = \begin{matrix} [9X1]^T \\ [9X1]^T \\ [9X1]^T \end{matrix} \quad \textbf{[9*mX1]}$$

n blocks

m blocks

Engels, et al 2006

Note that $\mathbf{V}^{*-1} = \begin{pmatrix} \mathbf{V}_1^{*-1} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{V}_2^{*-1} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$

$$\begin{aligned} H_{PP} &= J_P^\top J_P & \equiv \mathbf{V}^* \\ H_{PC} &= J_P^\top J_C, & \equiv \mathbf{W}^T \\ H_{CC} &= J_C^\top J_C, & \equiv \mathbf{U}^* \\ b_P &= -J_P^\top f, & \equiv \epsilon_\mathbf{b} \\ b_C &= -J_C^\top f & \equiv \epsilon_\mathbf{a} \end{aligned}$$

$\mathbf{V}_i \equiv \sum_{j=1}^3 \mathbf{B}_{ij}^T \mathbf{B}_{ij}$, for 1 feature, sum over images

$\mathbf{U}_j \equiv \sum_{i=1}^4 \mathbf{A}_{ij}^T \mathbf{A}_{ij}$, for 1 image, sum over features

$$(\mathbf{U}^* - \mathbf{W}\ \mathbf{V}^{*-1}\ \mathbf{W}^T)\ \delta_\mathbf{a} = \epsilon_\mathbf{a} - \mathbf{W}\ \mathbf{V}^{*-1}\ \epsilon_\mathbf{b}$$

$$\underbrace{(H_{CC} - H_{PC}^\top H_{PP}^{-1} H_{PC})}_{A}\, dC = \underbrace{b_C - H_{PC}^\top H_{PP}^{-1} b_P}_{B}$$

Engels

5 (Optional) Fix gauge by freezing appropriate coordinates and thereby reducing the linear system with a few dimensions.

6 **(Linear Solving)** Cholesky factor the left hand side matrix B and solve for dC. Add frozen coordinates back in.

7 **(Back-substitution)** For each track p

{

Start with point update for this track $dp = t_p$.

For each camera c on track p

{

Subtract $T_{pc}^\top dc$ from dp (where dc is the update for camera c).

}

Compute updated point.

}

8 **Compute the cost function** for the updated camera and point configuration.

9 If cost function has improved, accept the update step, decrease λ and go to Step 3 (unless converged, in which case quit).

10 Otherwise, increase λ and go to Step 3 (unless exceeded the maximum number of iterations, in which case quit).

every real-valued symmetric positive-definite matrix has a unique Cholesky decomposition.

A in the RCM is the Schur complement of HPP (HPP is called V* by Lourakis).

V* is a symmetric positive definite matrix (spdm). There exists proof the that Schur complement of a spdm is a symmetric positive definite matrix.

So A can be solved by Cholesky decomposition.

Bill **Triggs**, Philip Mclauchlan, Richard Hartley, Andrew Fitzgibbon.
Bundle Adjustment – A Modern Synthesis.
International Workshop on Vision Algorithms,
Sep **2000**, Corfu, Greece. pp.298–372,
10.1007/3-540-44480-7_21 . inria-00548290 **see Appendix B, and page 23…**

$$(\mathbf{U}^* - \mathbf{W}\boxed{\mathbf{V}^{*-1}}\mathbf{W}^T)\,\delta_{\mathbf{a}} = \epsilon_{\mathbf{a}} - \mathbf{W}\,\mathbf{V}^{*-1}\,\epsilon_{\mathbf{b}}$$

$$\underbrace{\underbrace{(H_{CC} - H_{PC}^\top \boxed{H_{PP}^{-1}} H_{PC})}_{A}}_{\overline{\mathsf{D}}}\, dC = \underbrace{b_C - H_{PC}^\top H_{PP}^{-1} b_P}_{B}$$

6.1 The Schur Complement and the Reduced Bundle System

Schur complement: Consider the following block triangular matrix factorization:

$$\mathsf{M} = \begin{pmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \mathsf{C}\,\mathsf{A}^{-1} & 1 \end{pmatrix} \begin{pmatrix} \mathsf{A} & 0 \\ 0 & \overline{\mathsf{D}} \end{pmatrix} \begin{pmatrix} 1 & \mathsf{A}^{-1}\mathsf{B} \\ 0 & 1 \end{pmatrix}, \qquad \overline{\mathsf{D}} \equiv \mathsf{D} - \mathsf{C}\,\mathsf{A}^{-1}\mathsf{B} \tag{16}$$

$$\begin{pmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{pmatrix}^{-1} = \begin{pmatrix} 1 & -\mathsf{A}^{-1}\mathsf{B} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathsf{A}^{-1} & 0 \\ 0 & \overline{\mathsf{D}}^{-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\mathsf{C}\,\mathsf{A}^{-1} & 1 \end{pmatrix} = \begin{pmatrix} \mathsf{A}^{-1} + \mathsf{A}^{-1}\mathsf{B}\,\overline{\mathsf{D}}^{-1}\mathsf{C}\,\mathsf{A}^{-1} & -\mathsf{A}^{-1}\mathsf{B}\,\overline{\mathsf{D}}^{-1} \\ -\overline{\mathsf{D}}^{-1}\mathsf{C}\,\mathsf{A}^{-1} & \overline{\mathsf{D}}^{-1} \end{pmatrix} \tag{17}$$

Here A must be square and invertible, and for (17), the whole matrix must also be square and invertible. $\overline{\mathsf{D}}$ is called the **Schur complement** of A in M. If both A and D are invertible, complementing on D rather than A gives **$\overline{\mathsf{D}}$, the Schur complement, is HPP is V*.**

$$\begin{pmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \overline{\mathsf{A}}^{-1} & -\overline{\mathsf{A}}^{-1}\mathsf{B}\,\mathsf{D}^{-1} \\ -\mathsf{D}\,\mathsf{C}\,\overline{\mathsf{A}}^{-1} & \mathsf{D}^{-1} + \mathsf{D}^{-1}\mathsf{C}\,\overline{\mathsf{A}}^{-1}\mathsf{B}\,\mathsf{D}^{-1} \end{pmatrix}, \qquad \overline{\mathsf{A}} = \mathsf{A} - \mathsf{B}\boxed{\mathsf{D}^{-1}}\mathsf{C}$$

Equating upper left blocks gives the **Woodbury formula**:

$$(\mathsf{A} \pm \mathsf{B}\,\mathsf{D}^{-1}\mathsf{C})^{-1} = \mathsf{A}^{-1} \mp \mathsf{A}^{-1}\mathsf{B}\,(\mathsf{D} \pm \mathsf{C}\,\mathsf{A}^{-1}\mathsf{B})^{-1}\,\mathsf{C}\,\mathsf{A}^{-1} \tag{18}$$

This is the usual method of updating the inverse of a nonsingular matrix A after an update (especially a low rank one) $\mathsf{A} \to \mathsf{A} \pm \mathsf{B}\,\mathsf{D}^{-1}\mathsf{C}$. (See §8.1).

Bill **Triggs**, Philip Mclauchlan, Richard Hartley, Andrew Fitzgibbon.
Bundle Adjustment – A Modern Synthesis.
International Workshop on Vision Algorithms,
Sep **2000**, Corfu, Greece. pp.298–372,
10.1007/3-540-44480-7_21 . inria-00548290

see Appendix B, and page 23…

$\mathsf{L} = \textbf{profile\_cholesky\_decomp}(\mathsf{A})$
  for $i = 1$ **to** n **do**
    for $j = \text{first}(i)$ **to** i **do**

$$a = \mathsf{A}_{ij} - \sum_{k=\max(\text{first}(i),\text{first}(j))}^{j-1} \mathsf{L}_{ik}\,\mathsf{L}_{jk}$$

$$\mathsf{L}_{ij} = (j < i)\ ?\ a\,/\,\mathsf{L}_{jj}\ :\ \sqrt{a}$$

$\mathsf{x} = \textbf{profile\_cholesky\_forward\_subs}(\mathsf{A}, \mathsf{b})$
  for $i = \text{first}(\mathsf{b})$ **to** n **do**

$$\mathsf{x}_i = \left(\mathsf{b}_i - \sum_{k=\max(\text{first}(i),\text{first}(\mathsf{b}))}^{i-1} \mathsf{L}_{ik}\,\mathsf{x}_k\right) / \mathsf{L}ii$$

$\mathsf{y} = \textbf{profile\_cholesky\_back\_subs}(\mathsf{A}, \mathsf{x})$
  $\mathsf{y} = \mathsf{x}$
  for $i = \text{last}(\mathsf{b})$ **to** 1 **step** -1 **do**
    for $k = \max(\text{first}(i), \text{first}(\mathsf{y}))$ **to** i **do**

$$\mathsf{y}_k = \mathsf{y}_k - \mathsf{y}_i\,\mathsf{L}_{ik}$$

$$\mathsf{y}_i = \mathsf{y}_i\,/\,\mathsf{L}_{ii}$$

Figure 10: A complete implementation of profile Cholesky decomposition.

cholesky:

_A_ = L ∗ D ∗ L^T
= L ∗ √D ∗ √D ∗ L^T
let C = L ∗ √D
then **_A_** = **C ∗ C^T**
aside: L is invertible if none of its diagonal elements are 0.

for _A_=L ∗ L*
(_A_)^-1 = (L*) ∗ (L ∗ L*)^-1

but usually, for A ∗ x= b:
(1) A=L ∗ L*
(2) L ∗ y = b ==> y via forward subst
(3) L* ∗ x = y ==> x via backward subst

http://users.ics.forth.gr/~argyros/mypapers/2004_08_tr340_forth_sba.pdf

The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm†

Manolis I.A. Lourakis and Antonis A. Argyros

In all cases, the function pointed to by `proj` is assumed to estimate in `xij` the projection in image `j` of the point `i`. Arguments `aj` and `bi` are respectively the parameters of the `j`-th camera and `i`-th point. In other words, `proj` implements the parameterizing function $\mathbf{Q}()$. Similarly, `projac` is assumed to compute in `Aij` and `Bij` the functions $\frac{\partial \mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)}{\partial \mathbf{a}_j}$ and $\frac{\partial \mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)}{\partial \mathbf{b}_i}$, i.e. the jacobians with respect to `aj` and `bi` of the projection of point `i` in image `j`. If `projac` is NULL, the jacobians are

The employed world coordinate frame is taken to be aligned with the initial camera location. All subsequent camera motions are defined relative to the initial location, through the combination of a 3D rotation and a 3D translation. A 3D rotation by an angle θ about a unit vector $\mathbf{u} = (u_1, u_2, u_3)^T$ is represented by the quaternion $\mathbf{R} = (\cos(\frac{\theta}{2}),\ u_1 \sin(\frac{\theta}{2}),\ u_2 \sin(\frac{\theta}{2}),\ u_3 \sin(\frac{\theta}{2}))$ [26]. A 3D translation is defined by a vector $\mathbf{t}$. A 3D point is represented by its Euclidean coordinate vector $\mathbf{M}$. Thus, the parameters of each camera j and point i are $\mathbf{a}_j = (\mathbf{R}_j, \mathbf{t}_j{}^T)^T$ and $\mathbf{b}_i = \mathbf{M}_i$, respectively. With the previous definitions, the predicted projection of point i on image j is

$$\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i) = \mathbf{K}\ (\mathbf{R}_j\ \mathbf{N}_i\ \mathbf{R}_j^{-1}\ +\ \mathbf{t}_j), \tag{28}$$

where $\mathbf{K}$ is the 3×3 intrinsic camera calibration matrix and $\mathbf{N}_i = (0,\ \mathbf{M}_i{}^T)$ is the vector quaternion corresponding to the 3D point $\mathbf{M}_i$. The expression $\mathbf{R}_j\ \mathbf{N}_i\ \mathbf{R}_j^{-1}$ corresponds to point $\mathbf{M}_i$ rotated by an angle θ_j about unit vector $\mathbf{u}_j$, as specified by the quaternion $\mathbf{R}_j$. Source file `eucsbademo.c` accompanying the `sba` package im-

http://users.ics.forth.gr/~argyros/mypapers/2004_08_tr340_forth_sba.pdf

The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm†

Manolis I.A. Lourakis and Antonis A. Argyros

algorithm[3]. This procedure can be embedded into the LM algorithm of section 2 at the point indicated by the rectangular box in Fig. 1, leading to a sparse bundle adjustment algorithm.

Figure 2: Algorithm for solving the sparse normal equations arising in generic bundle adjustment; see text for details.

Input: The current parameter vector partitioned into m camera parameter vectors $\mathbf{a}_j$ and n 3D point parameter vectors $\mathbf{b}_i$, a function $\mathbf{Q}$ employing the $\mathbf{a}_j$ and $\mathbf{b}_i$ to compute the predicted projections $\hat{\mathbf{x}}_{ij}$ of the i-th point on the j-th image, the observed image point locations $\mathbf{x}_{ij}$ and a damping term μ for LM.

Output: The solution δ to the normal equations involved in LM-based bundle adjustment.

Algorithm:

Compute the derivative matrices $\mathbf{A}_{ij} := \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j} = \frac{\partial \mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)}{\partial \mathbf{a}_j}$, $\mathbf{B}_{ij} := \frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i} = \frac{\partial \mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)}{\partial \mathbf{b}_i}$
and the error vectors $\epsilon_{ij} := \mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij}$,
where i and j assume values in $\{1, \ldots, n\}$ and $\{1, \ldots, m\}$ respectively.

Compute the following auxiliary variables:

$$\mathbf{U}_j := \sum_i \mathbf{A}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \mathbf{A}_{ij} \quad \mathbf{V}_i := \sum_j \mathbf{B}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \mathbf{B}_{ij} \quad \mathbf{W}_{ij} := \mathbf{A}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \mathbf{B}_{ij}$$

$$\epsilon_{\mathbf{a}_j} := \sum_i \mathbf{A}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij} \qquad \epsilon_{\mathbf{b}_i} := \sum_j \mathbf{B}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij}$$

Augment $\mathbf{U}_j$ and $\mathbf{V}_i$ by adding μ to their diagonals to yield $\mathbf{U}_j^*$ and $\mathbf{V}_i^*$.

Compute $\mathbf{Y}_{ij} := \mathbf{W}_{ij} \mathbf{V}_i^{*-1}$.

Compute $\delta_{\mathbf{a}}$ from $\mathbf{S}\,(\delta_{\mathbf{a}_1}{}^T, \delta_{\mathbf{a}_2}{}^T, \ldots, \delta_{\mathbf{a}_m}{}^T)^T = (\mathbf{e}_1{}^T, \mathbf{e}_2{}^T, \ldots, \mathbf{e}_m{}^T)^T$,
where $\mathbf{S}$ is a matrix consisting of $m \times m$ blocks; block jk is defined by
$\mathbf{S}_{jk} = \delta_{jk} \mathbf{U}_j^* - \sum_i \mathbf{Y}_{ij} \mathbf{W}_{ik}{}^T$, where δ_{jk} is Kronecker's delta
and
$\mathbf{e}_j = \epsilon_{\mathbf{a}_j} - \sum_i \mathbf{Y}_{ij}\, \epsilon_{\mathbf{b}_i}$.

Compute each $\delta_{\mathbf{b}_i}$ from the equation $\delta_{\mathbf{b}_i} = \mathbf{V}_i^{*-1}\,(\epsilon_{\mathbf{b}_i} - \sum_j \mathbf{W}_{ij}{}^T\, \delta_{\mathbf{a}_j})$.

Form δ as $(\delta_{\mathbf{a}}{}^T,\ \delta_{\mathbf{b}}{}^T)^T$.

http://users.ics.forth.gr/~argyros/mypapers/2004_08_tr340_forth_sba.pdf

The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm†

Manolis I.A. Lourakis and Antonis A. Argyros

Figure 1: Levenberg-Marquardt non-linear least squares algorithm; see text and [16, 20] for details. The reason for enclosing a statement in a rectangular box will be explained in section 3.

Input: A vector function $f : \mathcal{R}^m \rightarrow \mathcal{R}^n$ with $n \geq m$, a measurement vector $\mathbf{x} \in \mathcal{R}^n$ and an initial parameters estimate $\mathbf{p}_0 \in \mathcal{R}^m$.
Output: A vector $\mathbf{p}^+ \in \mathcal{R}^m$ minimizing $||\mathbf{x} - f(\mathbf{p})||^2$.
Algorithm:

$k := 0$; $\nu := 2$; $\mathbf{p} := \mathbf{p}_0$;
$\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_\mathbf{p} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_\mathbf{p}$;
stop:=$(||\mathbf{g}||_\infty \leq \varepsilon_1)$; $\mu := \tau * \max_{i=1,\ldots,m}(A_{ii})$;
while (not stop) and $(k < k_{max})$
    $k := k + 1$;
    repeat
        $\boxed{\text{Solve } (\mathbf{A} + \mu\mathbf{I})\delta_\mathbf{p} = \mathbf{g};}$
        if $(||\delta_\mathbf{p}|| \leq \varepsilon_2||\mathbf{p}||)$
            stop:=true;
        else
            $\mathbf{p}_{new} := \mathbf{p} + \delta_\mathbf{p}$;
            $\rho := (||\epsilon_\mathbf{p}||^2 - ||\mathbf{x} - f(\mathbf{p}_{new})||^2)/(\delta_\mathbf{p}^T(\mu\delta_\mathbf{p} + \mathbf{g}))$;
            if $\rho > 0$
                $\mathbf{p} = \mathbf{p}_{new}$;
                $\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_\mathbf{p} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_\mathbf{p}$;
                stop:=$(||\mathbf{g}||_\infty \leq \varepsilon_1)$;
                $\mu := \mu * \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$; $\nu := 2$;
            else
                $\mu := \mu * \nu$; $\nu := 2 * \nu$;
            endif
        endif
    until $(\rho > 0)$ or (stop)
endwhile