Projective Bundle Adjustment from Random Initialization using the Variable Projection Method : Supplementary Document

Je Hyeong Hong[1], Christopher Zach[2], Andrew Fitzgibbon[3] and Roberto Cipolla[1]

[1]University of Cambridge, UK ** {jhh37,rc10001}@cam.ac.uk
[2]Toshiba Research Europe, Cambridge, UK christopher.m.zach@gmail.com
[3]Microsoft, Cambridge, UK awf@microsoft.com

Abstract. This document illustrates theoretical and implementation details of the main paper. All symbols used and all algorithms compiled in [**?**] are listed. We present an analytic form of each required derivative in detail, and summarize ways to constrain camera and point scales as well as gauge freedom by applying the Riemannian manifold optimization framework. At the end of the document, we also make a brief comparison between our variable projection (RW) projective bundle algorithms and those implemented using the Ceres-solver [1].

1 Nomenclature

A list of symbols used in [2] can be found in Table 1.

2 List of algorithms

A full list of compiled algorithms can be found in Table 2.

3 Analytic expressions of the required derivatives

In [2], we proposed to formulate the projective bundle adjustment problem as follows:

$$\min_{\{\mathtt{P}_i\}\{\tilde{\mathbf{x}}_j\}} \sum_{\{i,j\}\in\Omega} \|\boldsymbol{\pi}\left(\mathtt{P}_i\tilde{\mathbf{x}}_j\right) - \tilde{\mathbf{m}}_{ij}\|_2^2 \tag{1}$$

$$= \min_{\{\mathbf{p}_i\},\{\mathbf{q}_i\},\{s_i\},\{\mathbf{x}_j\},\{t_j\}} \sum_{(i,j)\in\Omega} \|\boldsymbol{\varepsilon}_{ij}(\mathbf{p}_i,\mathbf{q}_i,s_i,\mathbf{x}_j,t_j,\mu_i)\|_2^2, \tag{2}$$

where the unknowns are listed in Table 1.

In §5 of [2], we showed that only three derivatives are required to compile all algorithms listed in Table 2. In this section, we present the analytic form of these derivatives obtained by using the techniques illustrated in [3–5].

** Much of the work was done while the first author was an intern at Toshiba Research Europe.

3.1 Jacobian with respect to the camera parameters

As shown in Table 3 of [2], the Jacobian with respect to the camera parameters ($\partial\boldsymbol{\varepsilon}/\partial\tilde{\mathbf{p}}$) consists of three smaller Jacobians which are $\partial\boldsymbol{\varepsilon}/\partial\mathbf{p}$, $\partial\boldsymbol{\varepsilon}/\partial\mathbf{q}$ and $\partial\boldsymbol{\varepsilon}/\partial\mathbf{s}$. We first obtain the analytic derivatives of a single residual $\boldsymbol{\varepsilon}_{ij}$ with respect to $\mathbf{p}_i$, $\mathbf{q}_i$ and s_i:

$$\frac{\partial\boldsymbol{\varepsilon}_{ij}}{\partial\mathbf{p}_i} = \frac{\partial\boldsymbol{\pi}_{ij}}{\partial\mathbf{p}_i} = \frac{1}{\mu_i(\mathbf{q}_i^\top\mathbf{x}_j)+s_it_j}\frac{\partial\begin{bmatrix}\mathbf{p}_{i1:}^\top\tilde{\mathbf{x}}_j\\ \mathbf{p}_{i2:}^\top\tilde{\mathbf{x}}_j\end{bmatrix}}{\partial\mathbf{p}_i} = \frac{\begin{bmatrix}\tilde{\mathbf{x}}_j^\top & \\ & \tilde{\mathbf{x}}_j^\top\end{bmatrix}}{\mu_i(\mathbf{q}_i^\top\mathbf{x}_j)+s_it_j} \tag{3}$$

$$\frac{\partial\boldsymbol{\varepsilon}_{ij}}{\partial\mathbf{q}_i} = \frac{\partial\boldsymbol{\pi}_{ij}}{\partial\mathbf{q}_i} = \frac{\partial}{\partial\mathbf{q}_i}\left[\frac{1}{\mu_i(\mathbf{q}_i^\top\mathbf{x}_j)+s_it_j}\right]\begin{bmatrix}\mathbf{p}_{i1:}^\top\tilde{\mathbf{x}}_j\\ \mathbf{p}_{i2:}^\top\tilde{\mathbf{x}}_j\end{bmatrix} = \frac{-\mu_i\begin{bmatrix}\mathbf{p}_{i1:}^\top\tilde{\mathbf{x}}_j\\ \mathbf{p}_{i2:}^\top\tilde{\mathbf{x}}_j\end{bmatrix}\mathbf{x}_j^\top}{(\mu_i(\mathbf{q}_i^\top\mathbf{x}_j)+s_it_j)^2} \tag{4}$$

$$\frac{\partial\boldsymbol{\varepsilon}_{ij}}{\partial s_i} = \frac{\partial\boldsymbol{\pi}_{ij}}{\partial s_i} = \frac{\partial}{\partial s_i}\left[\frac{1}{\mu_i(\mathbf{q}_i^\top\mathbf{x}_j)+s_it_j}\right]\begin{bmatrix}\mathbf{p}_{i1:}^\top\tilde{\mathbf{x}}_j\\ \mathbf{p}_{i2:}^\top\tilde{\mathbf{x}}_j\end{bmatrix} = \frac{-t_j\begin{bmatrix}\mathbf{p}_{i1:}^\top\tilde{\mathbf{x}}_j\\ \mathbf{p}_{i2:}^\top\tilde{\mathbf{x}}_j\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top\mathbf{x}_j)+s_it_j)^2} \tag{5}$$

Using above results, we can construct $\partial\boldsymbol{\varepsilon}_{i:}/\partial\mathbf{p}_i$, $\partial\boldsymbol{\varepsilon}_{i:}/\partial\mathbf{q}_i$ and $\partial\boldsymbol{\varepsilon}_{i:}/\partial s_i$ by stacking all the residuals of visible points for frame i. Further noting that $\boldsymbol{\varepsilon}_{i:}$ only depends on camera i leads to

$$\frac{\partial\boldsymbol{\varepsilon}}{\partial\mathbf{p}} = \begin{bmatrix}\frac{\partial\boldsymbol{\varepsilon}_{1:}}{\partial\mathbf{p}_1} & \cdots & \frac{\partial\boldsymbol{\varepsilon}_{1:}}{\partial\mathbf{p}_f}\\ \vdots & \ddots & \vdots\\ \frac{\partial\boldsymbol{\varepsilon}_{f:}}{\partial\mathbf{p}_1} & \cdots & \frac{\partial\boldsymbol{\varepsilon}_{f:}}{\partial\mathbf{p}_f}\end{bmatrix} = \begin{bmatrix}\frac{\partial\boldsymbol{\varepsilon}_{1:}}{\partial\mathbf{p}_1} & & \\ & \ddots & \\ & & \frac{\partial\boldsymbol{\varepsilon}_{f:}}{\partial\mathbf{p}_f}\end{bmatrix} =: \texttt{blkdiag}\left(\frac{\partial\boldsymbol{\varepsilon}_{1:}}{\partial\mathbf{p}_1},\cdots,\frac{\partial\boldsymbol{\varepsilon}_{f:}}{\partial\mathbf{p}_f}\right), \tag{6}$$

where `blkdiag` is a standard MATLAB function for creating a block-diagonal matrix from input submatrices. Similarly, we obtain

$$\frac{\partial\boldsymbol{\varepsilon}}{\partial\mathbf{q}} = \texttt{blkdiag}\left(\frac{\partial\boldsymbol{\varepsilon}_{1:}}{\partial\mathbf{q}_1},\cdots,\frac{\partial\boldsymbol{\varepsilon}_{f:}}{\partial\mathbf{q}_f}\right) \tag{7}$$

$$\frac{\partial\boldsymbol{\varepsilon}}{\partial\mathbf{s}} = \texttt{blkdiag}\left(\frac{\partial\boldsymbol{\varepsilon}_{1:}}{\partial s_1},\cdots,\frac{\partial\boldsymbol{\varepsilon}_{f:}}{\partial s_f}\right). \tag{8}$$

We now have sufficient building blocks for computing $\mathtt{J}_{\tilde{\mathbf{p}}}$.

3.2 Jacobian with respect to the point parameters

As shown in Table 3 of [2], the Jacobian with respect to the point parameters ($\partial\boldsymbol{\varepsilon}/\partial\tilde{\mathbf{x}}$) consists of two smaller Jacobians which are $\partial\boldsymbol{\varepsilon}/\partial\mathbf{x}$ and $\partial\boldsymbol{\varepsilon}/\partial\mathbf{t}$. Similar to §3.1, we first obtain the analytic derivatives of a single residual $\boldsymbol{\varepsilon}_{ij}$ with respect to $\mathbf{x}_j$ and t_j:

$$\frac{\partial \boldsymbol{\varepsilon}_{ij}}{\partial \mathbf{x}_j} = \frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \frac{\partial \begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{\partial \mathbf{x}_j} + \frac{\partial}{\partial \mathbf{x}_j} \left[\frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \right] \begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \tag{9}$$

$$= \frac{\begin{bmatrix} p_{i1} & p_{i2} & p_{i3} \\ p_{i5} & p_{i6} & p_{i7} \end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} - \frac{\mu_i \begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \mathbf{q}_i^\top}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2} \tag{10}$$

$$\frac{\partial \boldsymbol{\varepsilon}_{ij}}{\partial t_j} = \frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \frac{\partial \begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{\partial t_j} + \frac{\partial}{\partial t_j} \left[\frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \right] \begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \tag{11}$$

$$= \frac{\begin{bmatrix} p_{i4} \\ p_{i8} \end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} - \frac{s_i \begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2}. \tag{12}$$

This then yields

$$\frac{\partial \boldsymbol{\varepsilon}_{ij}}{\partial \tilde{\mathbf{x}}_j} = \begin{bmatrix} \frac{\partial \boldsymbol{\varepsilon}_{ij}}{\partial \mathbf{x}_j} & \frac{\partial \boldsymbol{\varepsilon}_{ij}}{\partial t_j} \end{bmatrix} = \frac{\begin{bmatrix} \mathbf{p}_{i1:}^\top \\ \mathbf{p}_{i2:}^\top \end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} - \frac{\begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \begin{bmatrix} \mu_i \mathbf{q}_i^\top & s_i \end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2}. \tag{13}$$

Using the above result, we can construct $\partial \boldsymbol{\varepsilon}_{:j} / \partial \tilde{\mathbf{x}}_j$ and by stacking all the residuals of visible frames for point j. Further noting that $\boldsymbol{\varepsilon}_{:j}$ only depends on point j leads to

$$\frac{\partial \boldsymbol{\varepsilon}_{pm}}{\partial \tilde{\mathbf{x}}} = \begin{bmatrix} \frac{\partial \boldsymbol{\varepsilon}_{:1}}{\partial \tilde{\mathbf{x}}_1} & \cdots & \frac{\partial \boldsymbol{\varepsilon}_{:1}}{\partial \tilde{\mathbf{x}}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{\varepsilon}_{:n}}{\partial \tilde{\mathbf{x}}_1} & \cdots & \frac{\partial \boldsymbol{\varepsilon}_{:n}}{\partial \tilde{\mathbf{x}}_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{\varepsilon}_{:1}}{\partial \tilde{\mathbf{x}}_1} & & \\ & \ddots & \\ & & \frac{\partial \boldsymbol{\varepsilon}_{:n}}{\partial \tilde{\mathbf{x}}_n} \end{bmatrix} =: \texttt{blkdiag} \left(\frac{\partial \boldsymbol{\varepsilon}_{:1}}{\partial \tilde{\mathbf{x}}_1}, \cdots, \frac{\partial \boldsymbol{\varepsilon}_{:n}}{\partial \tilde{\mathbf{x}}_n} \right), \tag{14}$$

where `blkdiag` is a standard MATLAB function for creating a block-diagonal matrix from input submatrices. We can then obtain $\mathtt{J}_{\tilde{\mathbf{x}}} := \mathtt{K}_{p2f} [\partial \boldsymbol{\varepsilon}_{pm} / \partial \tilde{\mathbf{x}}]$, where the permutation matrix $\mathtt{K}_{p2f}$ is used to reorder the residuals in the camera-major form.

3.3 Additional derivative for RW1

For the RW1 algorithm, we need to compute $\partial [\mathtt{J}_{\tilde{\mathbf{x}}}^\dagger] \boldsymbol{\varepsilon} / \partial \tilde{\mathbf{p}}$. The derivative of matrix pseudo-inverse is given in the supplementary document of [5]. Applying this yields:

$$\partial [\mathtt{J}_{\tilde{\mathbf{x}}}^\dagger] \boldsymbol{\varepsilon} = (-\mathtt{J}_{\tilde{\mathbf{x}}}^\dagger \partial [\mathtt{J}_{\tilde{\mathbf{x}}}] \mathtt{J}_{\tilde{\mathbf{x}}}^\dagger + (\mathtt{J}_{\tilde{\mathbf{x}}}^\top \mathtt{J}_{\tilde{\mathbf{x}}})^{-1} \partial [\mathtt{J}_{\tilde{\mathbf{x}}}]^\top (\mathtt{I} - \mathtt{J}_{\tilde{\mathbf{x}}} \mathtt{J}_{\tilde{\mathbf{x}}}^\dagger)) \boldsymbol{\varepsilon} \tag{15}$$

$$= (\mathtt{J}_{\tilde{\mathbf{x}}}^\top \mathtt{J}_{\tilde{\mathbf{x}}})^{-1} \partial [\mathtt{J}_{\tilde{\mathbf{x}}}]^\top \boldsymbol{\varepsilon} - \mathtt{J}_{\tilde{\mathbf{x}}}^\dagger \partial [\mathtt{J}_{\tilde{\mathbf{x}}}] \mathtt{J}_{\tilde{\mathbf{x}}}^\dagger \boldsymbol{\varepsilon} - (\mathtt{J}_{\tilde{\mathbf{x}}}^\top \mathtt{J}_{\tilde{\mathbf{x}}})^{-1} \partial [\mathtt{J}_{\tilde{\mathbf{x}}}]^\top \mathtt{J}_{\tilde{\mathbf{x}}} \mathtt{J}_{\tilde{\mathbf{x}}}^\dagger \boldsymbol{\varepsilon}. \tag{16}$$

We note that $\mathtt{J}_{\tilde{\mathbf{x}}}^\dagger \boldsymbol{\varepsilon} = \boldsymbol{\Delta} \tilde{\mathbf{x}}$, which is the last inner iteration update for points $(\tilde{\mathbf{x}})$. At this inner iteration stage, the points should have already converged to $\tilde{\mathbf{x}}^*(\tilde{\mathbf{p}})$ using a

second-order solver and therefore $\boldsymbol{\Delta}\tilde{\mathbf{x}}$ should be below the tolerance level and close to 0. Hence, (16) simplies to

$$\lim_{\boldsymbol{\Delta}\tilde{\mathbf{x}}\to 0}\left[\partial[\mathtt{J}^{\dagger}_{\tilde{\mathbf{x}}}]\boldsymbol{\varepsilon}\right] = (\mathtt{J}^{\top}_{\tilde{\mathbf{x}}}\mathtt{J}_{\tilde{\mathbf{x}}})^{-1}\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^{\top}\boldsymbol{\varepsilon}. \tag{17}$$

In essence, we need to be able to compute $\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^{\top}\boldsymbol{\varepsilon}/\partial\tilde{\mathbf{p}}$.

We start by deriving $\partial[\partial\boldsymbol{\varepsilon}_{ij}/\partial\tilde{\mathbf{x}}_j]^{\top}\boldsymbol{\varepsilon}_{ij}$ with respect to the i-th camera parameters.

$$\frac{\partial}{\partial\mathbf{p}_i}\left[\frac{\partial\boldsymbol{\varepsilon}_{ij}}{\partial\tilde{\mathbf{x}}_j}\right]^{\top}\boldsymbol{\varepsilon}_{ij} = \frac{\partial}{\partial\mathbf{p}_i}\left[\frac{\begin{bmatrix}\mathbf{p}^{\top}_{i1:}\\ \mathbf{p}^{\top}_{i2:}\end{bmatrix}}{\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j} - \frac{\begin{bmatrix}\mathbf{p}^{\top}_{i1:}\tilde{\mathbf{x}}_j\\ \mathbf{p}^{\top}_{i2:}\tilde{\mathbf{x}}_j\end{bmatrix}\begin{bmatrix}\mu_i\mathbf{q}_i^{\top} & s_i\end{bmatrix}}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2}\right]^{\top}\boldsymbol{\varepsilon}_{ij} \tag{18}$$

$$= \frac{1}{\partial\mathbf{p}_i}\left[\frac{[\boldsymbol{\varepsilon}_{ij1}\partial\mathbf{p}_{i1:}+\boldsymbol{\varepsilon}_{ij2}\partial\mathbf{p}_{i2:}]}{\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j}\right. \tag{19}$$

$$\left. - \frac{\begin{bmatrix}\mu_i\mathbf{q}_i\\ s_i\end{bmatrix}[\boldsymbol{\varepsilon}_{ij1}\tilde{\mathbf{x}}_j^{\top}\partial\mathbf{p}_{i1:}+\boldsymbol{\varepsilon}_{ij2}\tilde{\mathbf{x}}_j^{\top}\partial\mathbf{p}_{i2:}]}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2}\right] \tag{20}$$

$$= \frac{\boldsymbol{\varepsilon}_{ij}^{\top}\otimes\mathtt{I}_4}{\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j} - \frac{\begin{bmatrix}\mu_i\mathbf{q}_i\\ s_i\end{bmatrix}(\boldsymbol{\varepsilon}_{ij}^{\top}\otimes\tilde{\mathbf{x}}_j^{\top})}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2} \tag{21}$$

$$\frac{\partial}{\partial\mathbf{q}_i}\left[\frac{\partial\boldsymbol{\varepsilon}_{ij}}{\partial\tilde{\mathbf{x}}_j}\right]^{\top}\boldsymbol{\varepsilon}_{ij} = \frac{\partial}{\partial\mathbf{q}_i}\left[\frac{\begin{bmatrix}\mathbf{p}^{\top}_{i1:}\\ \mathbf{p}^{\top}_{i2:}\end{bmatrix}}{\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j} - \frac{\begin{bmatrix}\mathbf{p}^{\top}_{i1:}\tilde{\mathbf{x}}_j\\ \mathbf{p}^{\top}_{i2:}\tilde{\mathbf{x}}_j\end{bmatrix}\begin{bmatrix}\mu_i\mathbf{q}_i^{\top} & s_i\end{bmatrix}}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2}\right]^{\top}\boldsymbol{\varepsilon}_{ij} \tag{22}$$

$$= -\frac{\mu_i\begin{bmatrix}\mathbf{p}_{i1:} & \mathbf{p}_{i2:}\end{bmatrix}\boldsymbol{\varepsilon}_{ij}\mathbf{x}_j^{\top}}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2} - \frac{\begin{bmatrix}\mathbf{p}^{\top}_{i1:}\tilde{\mathbf{x}}_j & \mathbf{p}^{\top}_{i2:}\tilde{\mathbf{x}}_j\end{bmatrix}\boldsymbol{\varepsilon}_{ij}\begin{bmatrix}\mu_i\mathtt{I}_3\\ \mathbf{0}_{1\times 4}\end{bmatrix}}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2} \tag{23}$$

$$+ \frac{2\mu_i\begin{bmatrix}\mathbf{p}^{\top}_{i1:}\tilde{\mathbf{x}}_j & \mathbf{p}^{\top}_{i2:}\tilde{\mathbf{x}}_j\end{bmatrix}\boldsymbol{\varepsilon}_{ij}\begin{bmatrix}\mu_i\mathbf{q}\\ s_i\end{bmatrix}\mathbf{x}_j^{\top}}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^3} \tag{24}$$

$$\frac{\partial}{\partial s_i}\left[\frac{\partial \boldsymbol{\varepsilon}_{ij}}{\partial \tilde{\mathbf{x}}_j}\right]^\top \boldsymbol{\varepsilon}_{ij} = \frac{\partial}{\partial s_i}\left[\frac{\begin{bmatrix}\mathbf{p}_{i1:}^\top \\ \mathbf{p}_{i2:}^\top\end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} - \frac{\begin{bmatrix}\mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j\end{bmatrix}\begin{bmatrix}\mu_i \mathbf{q}_i^\top & s_i\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2}\right]^\top \boldsymbol{\varepsilon}_{ij} \tag{25}$$

$$= -\frac{t_j \begin{bmatrix}\mathbf{p}_{i1:} & \mathbf{p}_{i2:}\end{bmatrix} \boldsymbol{\varepsilon}_{ij}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2} - \frac{\begin{bmatrix}\mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j & \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j\end{bmatrix} \boldsymbol{\varepsilon}_{ij} \begin{bmatrix}\mathbf{0}_{3\times 1} \\ 1\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2} \tag{26}$$

$$+ \frac{2t_j \begin{bmatrix}\mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j & \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j\end{bmatrix} \boldsymbol{\varepsilon}_{ij} \begin{bmatrix}\mu_i \mathbf{q} \\ s_i\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^3}. \tag{27}$$

Then, by noting the block-diagonal structure of $\partial \boldsymbol{\varepsilon}_{pm}/\partial \tilde{\mathbf{x}}$ in (14), we have

$$\frac{\partial}{\partial \mathbf{p}}[\mathrm{J}_{\tilde{\mathbf{x}}}]^\top \boldsymbol{\varepsilon} = \frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial \boldsymbol{\varepsilon}_{pm}}{\partial \tilde{\mathbf{x}}}\right]^\top \boldsymbol{\varepsilon}_{pm} = \begin{bmatrix}\frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial \boldsymbol{\varepsilon}_{:1}}{\partial \tilde{\mathbf{x}}_1}\right]^\top \boldsymbol{\varepsilon}_{:1} \\ \vdots \\ \frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial \boldsymbol{\varepsilon}_{:n}}{\partial \tilde{\mathbf{x}}_n}\right]^\top \boldsymbol{\varepsilon}_{:n}\end{bmatrix} = \begin{bmatrix}\sum_{i\in\Omega_1} \frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial \boldsymbol{\varepsilon}_{i1}}{\partial \tilde{\mathbf{x}}_1}\right]^\top \boldsymbol{\varepsilon}_{i1} \\ \vdots \\ \sum_{i\in\Omega_n} \frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial \boldsymbol{\varepsilon}_{in}}{\partial \tilde{\mathbf{x}}_n}\right]^\top \boldsymbol{\varepsilon}_{in}\end{bmatrix} \tag{28}$$

$$= \begin{bmatrix}\sum_{i\in\Omega_1} \frac{\partial}{\partial \mathbf{p}_1}\left[\frac{\partial \boldsymbol{\varepsilon}_{i1}}{\partial \tilde{\mathbf{x}}_1}\right]^\top \boldsymbol{\varepsilon}_{i1} & \cdots & \sum_{i\in\Omega_1} \frac{\partial}{\partial \mathbf{p}_f}\left[\frac{\partial \boldsymbol{\varepsilon}_{i1}}{\partial \tilde{\mathbf{x}}_1}\right]^\top \boldsymbol{\varepsilon}_{i1} \\ \vdots & \ddots & \vdots \\ \sum_{i\in\Omega_n} \frac{\partial}{\partial \mathbf{p}_1}\left[\frac{\partial \boldsymbol{\varepsilon}_{in}}{\partial \tilde{\mathbf{x}}_n}\right]^\top \boldsymbol{\varepsilon}_{in} & \cdots & \sum_{i\in\Omega_n} \frac{\partial}{\partial \mathbf{p}_f}\left[\frac{\partial \boldsymbol{\varepsilon}_{in}}{\partial \tilde{\mathbf{x}}_n}\right]^\top \boldsymbol{\varepsilon}_{in}\end{bmatrix}, \tag{29}$$

where Ω_j denotes the set of visible frames for point j. In other words, we just need to sum the quantities $\partial[\partial \boldsymbol{\varepsilon}_{ij}/\partial \tilde{\mathbf{x}}_j]^\top \boldsymbol{\varepsilon}_{ij}/\partial \mathbf{p}_i$ (which is derived in (21)) appropriately. Similar expressions can be obtained for $\partial[\mathrm{J}_{\tilde{\mathbf{x}}}]^\top \boldsymbol{\varepsilon}/\partial \mathbf{q}$ and $\partial[\mathrm{J}_{\tilde{\mathbf{x}}}]^\top \boldsymbol{\varepsilon}/\partial \mathbf{s}$, which are altogether combined to form $\partial[\mathrm{J}_{\tilde{\mathbf{x}}}]^\top \boldsymbol{\varepsilon}/\partial \tilde{\mathbf{p}}$.

4 Further details on constraining the scale freedoms

In this section, we describe briefly how to incorporate the Riemannian manifold optimization framework for constraining the scale freedoms.

4.1 Riemannian Manifold optimization

[6] extensively covers optimization strategies on Riemannian manifolds and [7] concisely illustrates how to incorporate the Grassmann manifold projection into optimization. We apply the Riemannian manifold optimization to a homogeneous vector [**?**] as shown below. (The homogeneous vector space is a subset of the Grassmann manifold since vector scaling does not change the column space and hence remains in the equivalence class.) We arbitrarily denote $\mathbf{u}$ as the homogeneous vector here.

1. If the current solution $\mathbf{u}$ is outside the underlying manifold, perform retraction. (For the Grassmann manifold, orthogonalize the parameters for numerical stability.)
2. Project the gradient to the tangent space of current $\mathbf{u}$ by multiplying the tangent space projector to the original gradient ($\mathbf{g}$) and call it $\mathbf{g}_p$.
3. Project the Hessian (or its approximation) to the tangent space of current $\mathbf{u}$ by multiplying the tangent space projector to the derivative of the projected gradient (or its approximation) and call it $\mathtt{H}_p$.
4. Obtain the parameter update by solving the projected sub-problem using a second-order solver:

$$\boldsymbol{\Delta}\mathbf{u} = \underset{\boldsymbol{\delta}\perp\mathbf{u}}{\arg\min}\, \mathbf{g}_p\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top \mathtt{H}_p\boldsymbol{\delta}, \tag{30}$$

   where $\boldsymbol{\delta}\perp\mathbf{u}$ is the linear constraint $\mathbf{u}^\top\boldsymbol{\delta} = 0$.
5. Since $\mathbf{u} + \boldsymbol{\Delta}\mathbf{u}$ is now outside the manifold, perform retraction similar to step 1.

[6] states that orthogonalizing the updated variable using the q-factor is an efficient retraction technique. Since any homogeneous vector has only one column space, this retraction simply becomes a normalization procedure. (i.e. $\mathbf{u} \leftarrow \mathbf{u}/\|\mathbf{u}\|_2$.)

4.2 Tangent space projectors for the homogeneous variable projection algorithms

For the variable projection (RW) algorithms, the point parameters $\tilde{\mathbf{x}}$ are updated in the inner loop and the camera parameters $\tilde{\mathbf{p}}$ are updated in the outer loop. For the homogeneous set of algorithms, the tangent space projector for the inner loop (i.e. over the points) is

$$\mathtt{P}_{r1} = \begin{bmatrix} \tilde{\mathbf{x}}_1^\perp & & \\ & \ddots & \\ & & \tilde{\mathbf{x}}_n^\perp \end{bmatrix} = \mathtt{blkdiag}(\tilde{\mathbf{x}}_1^\perp, \cdots, \tilde{\mathbf{x}}_n^\perp) \in \mathbb{R}^{3n\times 4n}, \tag{31}$$

where $\tilde{\mathbf{x}}_j^\perp$ is the left null space of $\tilde{\mathbf{x}}_j$. Hence, the reduced update is in $\mathbb{R}^{3n}$, which needs to be projected back to $\mathbb{R}^{4n}$ by multiplying $\mathtt{P}_{r1}^\top$ to the reduced update vector.

The tangent space projector for the outer loop (over the cameras) is

$$\mathtt{P}_{r2} = \begin{bmatrix} \tilde{\mathbf{p}}_1^\perp & & \\ & \ddots & \\ & & \tilde{\mathbf{p}}_f^\perp \end{bmatrix} = \mathtt{blkdiag}(\tilde{\mathbf{p}}_1^\perp, \cdots, \tilde{\mathbf{p}}_n^\perp) \in \mathbb{R}^{11f\times 12f}, \tag{32}$$

where $\tilde{\mathbf{p}}_i := [\mathbf{p}_i^\top, \mathbf{q}_i^\top, s_i]^\top$ and $\tilde{\mathbf{p}}_i^\perp$ is the left null space of $\tilde{\mathbf{p}}_i$. Similarly, the reduced update is in $\mathbb{R}^{11f}$, which needs to be projected back to $\mathbb{R}^{12f}$ by multiplying $\mathtt{P}_{r2}^\top$ to the reduced update vector.

4.3 Tangent space projector for the joint algorithms

Assuming that the parameters are stacked as $[\tilde{\mathbf{p}}_1, \cdots, \tilde{\mathbf{p}}_f, \tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_n]$, the corresponding tangent space projector for the homogeneous joint algorithms is

$$\mathtt{P}_{r3} = \begin{bmatrix} \tilde{\mathbf{p}}_1^{\perp} & & & & & \\ & \ddots & & & & \\ & & \tilde{\mathbf{p}}_f^{\perp} & & & \\ & & & \tilde{\mathbf{x}}_1^{\perp} & & \\ & & & & \ddots & \\ & & & & & \tilde{\mathbf{x}}_n^{\perp} \end{bmatrix} \in \mathbb{R}^{(11f+3n)\times(12f+4n)}. \tag{33}$$

Hence, the reduced update will be in $\mathbb{R}^{(11f+3n)}$, which needs to be projected back to $\mathbb{R}^{(12f+4n)}$ by multiplying $\mathtt{P}_{r3}^{\top}$ to the reduced update vector.

5 Further details on constraining the gauge freedom of the variable projection methods

In [2], the gauge freedom is penalized using the rank-filling technique introduced in [8]. This can be regarded as a relaxed form of the Riemannian manifold optimization framework illustrated in §4.1.

Since the subproblem is already projected when constraining the scale freedoms (see §4), we instead add a gauge freedom penalty term to the subproblem which discourages updates along the column space of $\mathtt{P} := [\mathtt{P}_1^{\top}, \cdots, \mathtt{P}_f^{\top}]^{\top}$.

5.1 Homogeneous affine, homogeneous projective and Euclidean projective

As illustrated in Table 2 of §[2], the gauge can be any invertible matrix $\in \mathbb{R}^{4\times4}$. In each iteration, we wish to prevent any update in the direction spanning $\mathrm{col}(\mathtt{P})$.

Let us generalize the update in cameras ($\mathtt{P}$) such that

$$\mathtt{P} + \Delta\mathtt{P} = \mathtt{P}\mathtt{A} + \mathtt{P}^{\perp}\mathtt{B}, \tag{34}$$

where $\mathtt{P}^{\perp}$ is the left null space of $\mathtt{P}$. Then, by definition,

$$\mathtt{P}^{\top}\mathtt{P} + \mathtt{P}^{\top}\Delta\mathtt{P} = \mathtt{P}^{\top}\mathtt{P}\mathtt{A} + \mathtt{P}^{\top}\mathtt{P}^{\perp}\mathtt{B} = \mathtt{P}^{\top}\mathtt{P}\mathtt{A}. \tag{35}$$

Since we wish to prevent any update along the column space of $\mathtt{P}$, this fixes $\mathtt{A}$ to be $\mathtt{I}$, leading to the linear constraint $\mathtt{P}^{\top}\Delta\mathtt{P} = \mathtt{0}$. Hence, instead of solving (45), we solve the relaxed objective

$$\boldsymbol{\Delta}\mathbf{p} = \arg\min_{\boldsymbol{\delta}} \mathbf{g}\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^{\top}\mathtt{H}\boldsymbol{\delta} + \|\mathtt{P}^{\top}\Delta\mathtt{P}\|_2^2, \tag{36}$$

where $\boldsymbol{\Delta}\mathbf{p}$ is $\Delta\mathtt{P}$ vectorized.

5.2 Euclidean affine

As illustrated in Table 2 of §[2], the gauge for an Euclidean affine camera is any invertible matrix $\in \mathbb{R}^{4\times4}$ with the last row set to $[0,0,0,1]$. Similar to 5.1, In each update, we wish to prevent any update in the direction spanning the column space of $\mathtt{P}$.

Let us generalize the update in cameras ($\mathtt{P}$) such that

$$\mathtt{P} + \Delta\mathtt{P} = \mathtt{P}\begin{bmatrix}\mathtt{C} & \mathbf{d}\\ \mathbf{0}^\top & 1\end{bmatrix} + \mathtt{P}^\perp\mathtt{B}, \tag{37}$$

where $\mathtt{P}^\perp$ is the left null space of $\mathtt{P}$. Then, by definition,

$$\mathtt{P}^\top\mathtt{P} + \mathtt{P}^\top\Delta\mathtt{P} = \mathtt{P}^\top\mathtt{P}\mathtt{A} + \mathtt{P}^\top\mathtt{P}^\perp\mathtt{B} = \mathtt{P}^\top\mathtt{P}\mathtt{A}. \tag{38}$$

Now, we can replace $\mathtt{A}$ with the Euclidean affine gauge matrix

$$\mathtt{A} = \begin{bmatrix}\mathtt{C} & \mathbf{d}\\ \mathbf{0}^\top & 1\end{bmatrix} \tag{39}$$

$$\Rightarrow \mathtt{P}^\top\mathtt{P} + \mathtt{P}^\top\Delta\mathtt{P} = \mathtt{P}^\top\mathtt{P}\begin{bmatrix}\mathtt{C} & \mathbf{d}\\ \mathbf{0}^\top & 1\end{bmatrix}. \tag{40}$$

Separating $\mathtt{P}$ into $[\mathtt{P}_{1:3}, \mathbf{p}_4]$ gives

$$\begin{bmatrix}\mathtt{P}_{1:3}^\top\\ \mathbf{p}_4^\top\end{bmatrix}[\mathtt{P}_{1:3}, \mathbf{p}_4] + \begin{bmatrix}\mathtt{P}_{1:3}^\top\\ \mathbf{p}_4^\top\end{bmatrix}[\Delta\mathtt{P}_{1:3}, \boldsymbol{\Delta}\mathbf{p}_4] = \begin{bmatrix}\mathtt{P}_{1:3}^\top\\ \mathbf{p}_4^\top\end{bmatrix}[\mathtt{P}_{1:3}\mathtt{C}, \mathtt{P}_{i:3}\mathbf{d} + \mathbf{p}_4] \tag{41}$$

$$\Rightarrow \begin{bmatrix}\mathtt{P}_{1:3}^\top\mathtt{P}_{1:3} & \mathtt{P}_{1:3}^\top\mathbf{p}_4\\ \mathbf{p}_4^\top\mathtt{P}_{1:3} & \mathbf{p}_4^\top\mathbf{p}_4\end{bmatrix} + \begin{bmatrix}\mathtt{P}_{1:3}^\top\Delta\mathtt{P}_{1:3} & \mathtt{P}_{1:3}^\top\boldsymbol{\Delta}\mathbf{p}_4\\ \mathbf{p}_4^\top\Delta\mathtt{P}_{1:3} & \mathbf{p}_4^\top\boldsymbol{\Delta}\mathbf{p}_4\end{bmatrix} = \begin{bmatrix}\mathtt{P}_{1:3}^\top\mathtt{P}_{1:3}\mathtt{C}, \mathtt{P}_{1:3}^\top\mathtt{P}_{1:3}\mathbf{d} + \mathtt{P}_{1:3}^\top\mathbf{p}_4\\ \mathbf{p}_4^\top\mathtt{P}_{1:3}\mathtt{C}, \mathbf{p}_4^\top\mathtt{P}_{1:3}\mathbf{d} + \mathbf{p}_4^\top\mathbf{p}_4\end{bmatrix} \tag{42}$$

Similar to §5.1, we fix $\mathtt{C} = \mathtt{I}_3$ and $\mathbf{d} = \mathbf{0}$ to prevent the update along the column space of $\mathtt{P}$. This forms 4 linear constraints some of which are inter-dependent. We have empirically found that constraining the following two independent constraints yields the best performance: (applying all 4 seems to over-constrain the problem.)

$$\mathtt{P}_{1:3}^\top\Delta\mathtt{P}_{1:3} = \mathtt{0}, \tag{43}$$

$$\mathbf{p}_4^\top\boldsymbol{\Delta}\mathbf{p}_4 = 0. \tag{44}$$

We now solve the relaxed objective

$$\boldsymbol{\Delta}\mathbf{p} = \arg\min_{\boldsymbol{\delta}} \mathbf{g}\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top\mathtt{H}\boldsymbol{\delta} + \|\mathtt{P}_{1:3}^\top\Delta\mathtt{P}_{1:3}\|_2^2 + \|\mathbf{p}_4^\top\boldsymbol{\Delta}\mathbf{p}_4\|_2^2, \tag{45}$$

where $\boldsymbol{\Delta}\mathbf{p}$ is $\Delta\mathtt{P}$ vectorized.

6 A brief comparison to the Ceres-solver implementation of RW2

The official documentation of the Ceres-solver [1] states that the solver implements the Ruhe and Wedin algorithm 2 (RW2). With this in mind, we also implemented each algorithm in Table 2 using the Ceres-solver in hopes to improve the run speeds. For the VarPro (RW) projective bundle adjustment algorithms, We turned on the inner iterations setting in Ceres. Furthermore, for the set of homogeneous algorithms, We used the homogeneous vector parameterization (which is an instance of the local parameterization in Ceres) supplied by the solver.

Surprisingly, we were unable to duplicate our results with the above solver (v1.12). The critical issue is that AHRW2P and AIRW2P, which uses nonlinear RW2 and linear RW2 respectively, fails to hit the best affine optimum using Ceres when initialized from random starting points. We have observed this issue repeatedly across various datasets used in [2].

Similar phenomenon is also reported in [5], where the matrix factorization algorithms using Ceres with inner iterations showed worse performance compared to the authors' barebone MATLAB implementation of the RW algorithms.

Table 1: A list of symbols used in [2].

Symbol	Dimension	Definition
f	$\in \mathbb{R}$	The number of frames (cameras)
n	$\in \mathbb{R}$	The number of points
k	$\in \mathbb{R}$	The number of non-zero observed projections
f_j	$\in \mathbb{R}$	The number of visible frames for point j
n_i	$\in \mathbb{R}$	The number of visible points for camera i
$\mathtt{P}_i$	$\in \mathbb{R}^{3\times 4}$	The camera matrix for frame i: $\begin{bmatrix} p_{i1} & p_{i2} & p_{i3} & p_{i4} \\ p_{i5} & p_{i6} & p_{i7} & p_{i8} \\ \mu_i q_{i1} & \mu_i q_{i2} & \mu_i q_{i3} & s_i \end{bmatrix}$
$\mathtt{P}$	$\in \mathbb{R}^{3f\times 4}$	$[\mathtt{P}_1^\top, \cdots, \mathtt{P}_f^\top]^\top$
$\mathbf{p}_i$	$\in \mathbb{R}^8$	$[p_{i1}, \cdots, p_{i8}]^\top$
$\mathbf{q}_i$	$\in \mathbb{R}^3$	$[q_{i1}, q_{i2}, q_{i3}]^\top$
s_i	$\in \mathbb{R}$	The scaling factor for camera i
$\mathbf{p}$	$\in \mathbb{R}^{8f}$	$[\mathbf{p}_1^\top, \cdots, \mathbf{p}_f^\top]^\top$
$\mathbf{q}$	$\in \mathbb{R}^{3f}$	$[\mathbf{q}_1^\top, \cdots, \mathbf{q}_f^\top]^\top$
$\mathbf{s}$	$\in \mathbb{R}^{f}$	$[s_1, \cdots, s_f]^\top$
$\tilde{\mathbf{p}}$	$\in \mathbb{R}^{12f}$	$[\mathbf{p}^\top, \mathbf{q}^\top, \mathbf{s}^\top]^\top$
$\tilde{\mathbf{x}}_j$	$\in \mathbb{R}^4$	The homogeneous vector for point j: $[x_{j1}, x_{j2}, x_{j3}, t_j]^\top$
$\mathbf{x}_j$	$\in \mathbb{R}^3$	$[x_{j1}, x_{j2}, x_{j3}]^\top$
t_j	$\in \mathbb{R}$	The scaling factor for point j
$\tilde{\mathbf{x}}$	$\in \mathbb{R}^{4n}$	$[\tilde{\mathbf{x}}_1^\top, \cdots \tilde{\mathbf{x}}_n^\top]^\top$
$\mathbf{x}$	$\in \mathbb{R}^{3n}$	$[\mathbf{x}_1^\top, \cdots \mathbf{x}_n^\top]^\top$
$\mathbf{t}$	$\in \mathbb{R}^{n}$	$[t_1, \cdots t_n]^\top$
$\mathbf{m}_{ij}$	$\in \mathbb{R}^2$	The observed projection of point j in camera i
$\boldsymbol{\pi}_{ij}$	$\in \mathbb{R}^2$	The estimated projection of point j in camera i: $\boldsymbol{\pi}(\mathtt{P}(\mathbf{p}_i, \mathbf{q}_i, s_i, \mu_i), \tilde{\mathbf{x}}(\mathbf{x}_j, t_j)) = \frac{\begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j}$
$\boldsymbol{\varepsilon}_{ij}$	$\in \mathbb{R}^2$	$\boldsymbol{\pi}_{ij} - \mathbf{m}_{ij}$
$\boldsymbol{\varepsilon}_{i:}$	$\in \mathbb{R}^{2n_i}$	The collection of $\{\boldsymbol{\varepsilon}_{ij}\}$ for all points visible in camera i e.g. $[\boldsymbol{\varepsilon}_{i1}^\top, \cdots, \boldsymbol{\varepsilon}_{in}^\top]^\top$
$\boldsymbol{\varepsilon}_{:j}$	$\in \mathbb{R}^{2f_j}$	The collection of $\{\boldsymbol{\varepsilon}_{ij}\}$ for all frames in which point j is visible e.g. $[\boldsymbol{\varepsilon}_{1j}^\top, \cdots, \boldsymbol{\varepsilon}_{fj}^\top]^\top$
$\boldsymbol{\varepsilon}$	$\in \mathbb{R}^{2k}$	The frame-major stacking of the residuals: $[\boldsymbol{\varepsilon}_{1:}^\top, \cdots, \boldsymbol{\varepsilon}_{f:}^\top]^\top$
$\boldsymbol{\varepsilon}_{pm}$	$\in \mathbb{R}^{2k}$	The point-major stacking of the residuals: $[\boldsymbol{\varepsilon}_{:1}^\top, \cdots, \boldsymbol{\varepsilon}_{:n}^\top]^\top$
$\mathtt{K}_{p2f}$	$\in \mathbb{R}^{2k\times 2k}$	The permutation matrix defined such that $\boldsymbol{\varepsilon} = \mathtt{K}_{p2f}\boldsymbol{\varepsilon}_{pm}$
$\mathtt{J}_{\tilde{\mathbf{p}}}$	$\in \mathbb{R}^{2k\times 12f}$	The Jacobian with respect to the camera parameters (The effective number of columns may be smaller.)
$\mathtt{J}_{\tilde{\mathbf{x}}}$	$\in \mathbb{R}^{2k\times 4n}$	The Jacobian with respect to the point parameters (The effective number of columns may be smaller.)

Table 2: A full list of affine and projective bundle adjustment algorithms compiled.

ID	Camera	Form	Strategy	Algorithm	Constrained gauge
AHRW1P	Affine	Homogeneous	Nonlinear VarPro	RW1	16 / 16
AIRW1P	Affine	Euclidean	Linear VarPro	RW1	9 / 12
AHRW2P	Affine	Homogeneous	Nonlinear VarPro	RW2	16 / 16
AIRW2P	Affine	Euclidean	Linear VarPro	RW2	9 / 12
AHRW3P	Affine	Homogeneous	Nonlinear VarPro	RW3 (ALS)	0 / 0
AIRW3P	Affine	Euclidean	Linear VarPro	RW3 (ALS)	0 / 0
AHJP	Projective	Homogeneous	Joint	LM (JTJ)	None
AIJP	Projective	Euclidean	Joint	LM (JTJ)	None
PHRW1P	Projective	Homogeneous	Nonlinear VarPro	RW1	16 / 16
PIRW1P	Projective	Euclidean	Nonlinear VarPro	RW1	16 / 16
PHRW2P	Projective	Homogeneous	Nonlinear VarPro	RW2	16 / 16
PIRW2P	Projective	Euclidean	Nonlinear VarPro	RW2	16 / 16
PHRW3P	Projective	Homogeneous	Nonlinear VarPro	RW3 (ALS)	0 / 0
PIRW3P	Projective	Euclidean	Nonlinear VarPro	RW3 (ALS)	0 / 0
PHJP	Projective	Homogeneous	Joint	LM (JTJ)	None
PIJP	Projective	Euclidean	Joint	LM (JTJ)	None

References

1. Agarwal, S., Mierle, K., Others: Ceres solver. http://ceres-solver.org (2014)
2. Hong, J.H., Zach, C., Fitzgibbon, A.W., Cipolla, R.: Projective bundle adjustment from arbitrary initialization using the variable projection method. In: Proceedings of the 14th European Conference on Computer Vision (ECCV). (2016)
3. Minka, T.P.: Old and new matrix algebra useful for statistics. Technical report, Microsoft Research (2000)
4. Magnus, J.R., Neudecker, H.: Matrix Differential Calculus with Applications in Statistics and Econometrics. 3rd edn. (2007)
5. Hong, J.H., Fitzgibbon, A.W.: Secrets of matrix factorization: Approximations, numerics, manifold optimization and random restarts. In: Proceedings of the 2015 IEEE Internatonal Conference on Computer Vision (ICCV). (2015) 4130–4138
6. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press (2008)
7. Boumal, N., Absil, P.A.: RTRMC: A Riemannian trust-region method for low-rank matrix completion. In: Advances in Neural Information Processing Systems 24 (NIPS 2011). (2011) 406–414
8. Okatani, T., Yoshida, T., Deguchi, K.: Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms. In: Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV). (2011) 842–849