Ajustement de faisceaux : Matrice jacobienne et Complément de Schur

1 Algorithme d'ajustement de faisceaux

A l'issue de la mise en correspondances, l'information disponible est représentée sous la forme de « chemins » (« tracks » en anglais). Un « chemin » est propre à un point 3D et indique pour chaque image (si le point 3D est vu dans cette image) l'indice du point 2D auquel il correspond (parmi tous les points détectés dans l'image). Pour chaque image $\mathtt{I}_j$, si C_j points 3D sont vus dans cette image, nous disposons de deux vecteurs p2DId et p3DId de taille C_j. L'élément p2DId (c) indique l'indice du point 2D parmi tous les points ayant été détectés dans $\mathtt{I}_j$, alors que p3DId (c) indique l'indice du point 3D.

Ainsi l'algorithme d'ajustement de faisceaux consiste à minimiser la **somme des erreurs de reprojection**, ce qui conduit à la fonction de coût suivante :

$$\min_{\substack{\{\mathtt{R}_{wj},\mathbf{t}_{wj}\}_{j=0,M-1} \\ \{\mathbf{u}_i^w\}_{i=0\ldots N-1}}} \sum_{j=0}^{M-1}\sum_{c=0}^{C_j-1} \left\| \mathbf{p}_{j,\text{p2DId}(c)} - \mathtt{K}_j \pi\left(\mathtt{R}_{wj}^\top\left(\mathbf{u}^w_{\text{p3DId}(c)} - \mathbf{t}_{wj}\right)\right)\right\|_2^2 \tag{1}$$

où C_j est le nombre de points 3D vus dans l'image $\mathtt{I}_j$, $\pi(\cdot)$ est la fonction de projection, $\mathtt{K}_j$ est la matrice de calibration linéaire de la caméra j et $\mathbf{p}_{j,\text{p2DId}(c)}$ est un point 2D (qui dans un cas idéal devrait parfaitement correspondre à la reprojection du point 3D $\mathbf{u}^w_{\text{p3DId}(c)}$) dans l'image j. En pratique, la fonction de coût précédente est souvent minimisée en utilisant un algorithme de Levenberg-Marquardt.

2 Calcul des matrices jacobiennes

Afin d'implémenter un algorithme de Levenberg-Marquardt, il est nécessaire de connaître l'expression de la dérivée de $\mathtt{K}\pi\left(\mathtt{R}_{wj}^\top(\mathbf{u}_i^w - \mathbf{t}_{wj})\right)$ par rapport aux paramètres $\{\mathtt{R}_{wj},\mathbf{t}_{wj}\}_{j=0,M-1}$ et $\{\mathbf{u}_i^w\}_{i=0\ldots N-1}$. Utilisons les notations suivantes pour définir les incréments estimés à chaque itération du Levenberg-Marquardt : $\mathtt{R}_{wj}^{(l+1)} = \mathtt{R}_{wj}^{(l)}\text{expm}\left(\left[\boldsymbol{\delta}_{\mathtt{R}_{wj}}\right]^\wedge\right)$, $\mathbf{t}_{wj}^{(l+1)} = \mathbf{t}_{wj}^{(l)} + \boldsymbol{\delta}_{\mathbf{t}_{wj}}$ et $\mathbf{u}_i^{w,(l+1)} = \mathbf{u}_i^{w,(l)} + \boldsymbol{\delta}_{\mathbf{u}_i^w}$.

Remarque : Une matrice de rotation n'étant pas un élément d'un espace euclidien mais d'un groupe de Lie, l'incrément $\boldsymbol{\delta}_{\mathtt{R}_{wj}}$ ne peut pas être additif et doit être adapté à la géométrie de ce groupe de Lie. Ainsi $\boldsymbol{\delta}_{\mathtt{R}_{wj}}$ est un vecteur de taille 3 (car une matrice de rotation a 3 degrés de liberté) et se convertit en une matrice de rotation grâce à la fonction exponentielle de matrice.

La linéarisation de l'erreur de reprojection s'écrit alors (en omettant les indices l et $l+1$) :

$$\mathbf{p}_{j,i} - \mathtt{K}\pi\left(\left(\mathtt{R}_{wj}\mathrm{expm}\left(\left[\boldsymbol{\delta}_{\mathtt{R}_{wj}}\right]^{\wedge}\right)\right)^{\top}\left(\mathbf{u}_i^w + \boldsymbol{\delta}_{\mathbf{u}_i^w} - \mathbf{t}_{wj} - \boldsymbol{\delta}_{\mathbf{t}_{wj}}\right)\right) \approx \mathbf{r}_{j,i} - \mathtt{J}_{j,i}\boldsymbol{\delta} \tag{2}$$

où nous avons utilisé les notations suivantes :

$$\mathbf{r}_{j,i} = \mathbf{p}_{j,i} - \mathtt{K}\pi\left(\mathtt{R}_{wj}^{\top}\left(\mathbf{u}_i^w - \mathbf{t}_{wj}\right)\right), \tag{3}$$

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_{\mathtt{R}_{w0}} & \boldsymbol{\delta}_{\mathbf{t}_{w0}} & \boldsymbol{\delta}_{\mathtt{R}_{w1}} & \boldsymbol{\delta}_{\mathbf{t}_{w1}} & \cdots & \boldsymbol{\delta}_{\mathtt{R}_{w(M-1)}} & \boldsymbol{\delta}_{\mathbf{t}_{w(M-1)}} & \boldsymbol{\delta}_{\mathbf{u}_0^w} & \boldsymbol{\delta}_{\mathbf{u}_1^w} & \cdots & \boldsymbol{\delta}_{\mathbf{u}_{(N-1)}^w} \end{bmatrix}^{\top} \tag{4}$$

$$\mathtt{J}_{j,i} = \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix}\mathtt{J}_{\pi}\begin{bmatrix} \mathtt{J}_{\mathtt{R}_{w0}}^{i,j} & \mathtt{J}_{\mathbf{t}_{w0}}^{i,j} & \mathtt{J}_{\mathtt{R}_{w1}}^{i,j} & \mathtt{J}_{\mathbf{t}_{w1}}^{i,j} & \cdots & \mathtt{J}_{\mathtt{R}_{w(M-1)}}^{i,j} & \mathtt{J}_{\mathbf{t}_{w(M-1)}}^{i,j} & \mathtt{J}_{\mathbf{u}_0^w}^{i,j} & \mathtt{J}_{\mathbf{u}_1^w}^{i,j} & \cdots & \mathtt{J}_{\mathbf{u}_{(N-1)}^w}^{i,j} \end{bmatrix}, \tag{5}$$

$$\mathtt{J}_{\pi} = \frac{\partial}{\partial \boldsymbol{\delta}_{\mathbf{u}_i^j}}\pi\left(\mathbf{u}_i^j + \boldsymbol{\delta}_{\mathbf{u}_i^j}\right)\Big|_{\boldsymbol{\delta}_{\mathbf{u}_i^j}=\mathbf{0}} = \begin{bmatrix} \frac{1}{\mathbf{u}_i^j(z)} & 0 & -\frac{\mathbf{u}_i^j(x)}{\mathbf{u}_i^j(z)^2} \\ 0 & \frac{1}{\mathbf{u}_i^j(z)} & -\frac{\mathbf{u}_i^j(y)}{\mathbf{u}_i^j(z)^2} \end{bmatrix}, \tag{6}$$

$$\begin{aligned} \mathtt{J}_{\mathtt{R}_{wk}}^{i,j} &= \frac{\partial}{\partial \boldsymbol{\delta}_{\mathtt{R}_{wk}}}\left(\mathtt{R}_{wj}\mathrm{expm}\left(\left[\boldsymbol{\delta}_{\mathtt{R}_{wj}}\right]^{\wedge}\right)\right)^{\top}\left(\mathbf{u}_i^w - \mathbf{t}_{wj}\right)\Big|_{\delta_{\mathtt{R}_{wk}}=\mathbf{0}} \\ &= \begin{cases} \mathbf{0} & \text{si } k \neq j \\ \begin{bmatrix} \mathtt{G}_x^{\top}\mathbf{u}_i^j & \mathtt{G}_y^{\top}\mathbf{u}_i^j & \mathtt{G}_z^{\top}\mathbf{u}_i^j \end{bmatrix} & \text{si } k = j \end{cases}, \end{aligned} \tag{7}$$

$$\mathtt{G}_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \mathtt{G}_y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \mathtt{G}_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{8}$$

$$\begin{aligned} \mathtt{J}_{\mathbf{t}_{wk}}^{i,j} &= \frac{\partial}{\partial \boldsymbol{\delta}_{\mathbf{t}_{wk}}}\mathtt{R}_{wj}^{\top}\left(\mathbf{u}_i^w - \mathbf{t}_{wj} - \boldsymbol{\delta}_{\mathbf{t}_{wj}}\right)\Big|_{\boldsymbol{\delta}_{\mathbf{t}_{wk}}=\mathbf{0}} \\ &= \begin{cases} \mathbf{0} & \text{si } k \neq j \\ -\mathtt{R}_{wj}^{\top} & \text{si } k = j \end{cases}, \end{aligned} \tag{9}$$

$$\begin{aligned} \mathtt{J}_{\mathbf{u}_k^w}^{i,j} &= \frac{\partial}{\partial \boldsymbol{\delta}_{\mathbf{u}_k^w}}\mathtt{R}_{wj}^{\top}\left(\mathbf{u}_i^w + \boldsymbol{\delta}_{\mathbf{u}_i^w} - \mathbf{t}_{wj}\right)\Big|_{\boldsymbol{\delta}_{\mathbf{t}_{wk}}=\mathbf{0}} \\ &= \begin{cases} \mathbf{0} & \text{si } k \neq i \\ \mathtt{R}_{wj}^{\top} & \text{si } k = i \end{cases}. \end{aligned} \tag{10}$$

3 Implémentation efficace utilisant le complément de Schur

Remarquons que la matrice $\mathtt{J}_{j,i}$ (12) se décompose en deux parties. La $6M$ premières colonnes contiennent les dérivées par rapport aux caméras, alors les autres colonnes contiennent les dérivées par rapport aux points 3D. De plus, parmi les 6M premières colonnes, seulement 6 sont non nulles, et parmi les autres, seulement 3 sont non nulles :

$$\mathtt{J}_{j,i} = \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \mathtt{J}_\pi \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathtt{J}^{i,j}_{\mathtt{P}_{wj}} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathtt{J}^{i,j}_{\mathbf{u}^w_i} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}, \tag{11}$$

où nous avons défini la jacobienne d'une pose de caméra : $\mathtt{J}^{i,j}_{\mathtt{P}_{wj}} = \begin{bmatrix} \mathtt{J}^{i,j}_{\mathtt{R}_{wj}} \mathtt{J}^{i,j}_{\mathbf{t}_{wj}} \end{bmatrix}$. Ainsi $\mathtt{J}_{j,i}$ contient beaucoup de zéros. On dit qu'elle est *parcimonieuse*.

Au cœur de l'algorithme de Levenberg-Marquardt, nous devons résoudre un système linéaire de la forme :

$$(\mathtt{J}^\top \mathtt{J} + \lambda \mathtt{Id})\boldsymbol{\delta} = \mathtt{J}^\top \mathbf{r} \tag{12}$$

où $\mathtt{J}$ est une matrice concaténant les $\mathtt{J}_{j,i}$, de taille $2\sum_{j=1}^{M} C_j \times (6M + 3N)$, et $\mathbf{r}$ est un vecteur concaténant les $\mathbf{r}_{j,i}$, de taille $(2\sum_{j=1}^{M} C_j \times 1)$. Le système linéaire précédent est de taille $(6M + 3N) \times (6M + 3N)$, ce qui peut devenir très grand en pratique car le nombre N de points 3D peut facilement être de l'ordre de centaines de milliers. Résoudre un tel système linéaire sans tenir compte de sa structure parcimonieuse sera long et très inefficace. De plus, en fonction du nombre N de points 3D et du nombre M de caméras, stocker la matrice $\mathtt{J}^\top \mathtt{J} + \lambda \mathtt{Id}$ peut rapidement devenir problématique.

La matrice $\mathtt{J}$ préserve la structure parcimonieuse de $\mathtt{J}_{j,i}$ précédemment décrite que nous écrirons :

$$\mathtt{J} = [\mathtt{J}_\mathtt{P}\ \mathtt{J}_\mathtt{U}] \tag{13}$$

où $\mathtt{J}_\mathtt{P}$ est une matrice parcimonieuse de taille $2\sum_{j=1}^{M} C_j \times 6M$ et $\mathtt{J}_\mathtt{U}$ est une matrice parcimonieuse de taille $2\sum_{j=1}^{M} C_j \times 3N$. $\mathtt{J}^\top \mathtt{J}$ s'écrit alors :

$$\mathtt{J}^\top \mathtt{J} = \begin{bmatrix} \mathtt{J}_\mathtt{P}^\top \\ \mathtt{J}_\mathtt{U}^\top \end{bmatrix} [\mathtt{J}_\mathtt{P}\ \mathtt{J}_\mathtt{U}] = \begin{bmatrix} \mathtt{J}_\mathtt{P}^\top \mathtt{J}_\mathtt{P} & \mathtt{J}_\mathtt{P}^\top \mathtt{J}_\mathtt{U} \\ \mathtt{J}_\mathtt{U}^\top \mathtt{J}_\mathtt{P} & \mathtt{J}_\mathtt{U}^\top \mathtt{J}_\mathtt{U} \end{bmatrix}, \tag{14}$$

et le système linéaire se réécrit :

$$\begin{bmatrix} \mathtt{J}_\mathtt{P}^\top \mathtt{J}_\mathtt{P} + \lambda \mathtt{Id} & \mathtt{J}_\mathtt{P}^\top \mathtt{J}_\mathtt{U} \\ \mathtt{J}_\mathtt{U}^\top \mathtt{J}_\mathtt{P} & \mathtt{J}_\mathtt{U}^\top \mathtt{J}_\mathtt{U} + \lambda \mathtt{Id} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_\mathtt{P} \\ \boldsymbol{\delta}_\mathtt{U} \end{bmatrix} = \begin{bmatrix} \mathtt{J}_\mathtt{P}^\top \mathbf{r} \\ \mathtt{J}_\mathtt{U}^\top \mathbf{r} \end{bmatrix}. \tag{15}$$

$\mathtt{J}_\mathtt{P}^\top \mathtt{J}_\mathtt{P}$ est bloc-diagonale :

$$\mathtt{J}_\mathtt{P}^\top \mathtt{J}_\mathtt{P} = \begin{bmatrix} \sum_i \left(\mathtt{J}^{i,0}_{\mathtt{P}_{w0}}\right)^\top \mathtt{J}^{i,0}_{\mathtt{P}_{w0}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sum_i \left(\mathtt{J}^{i,1}_{\mathtt{P}_{w1}}\right)^\top \mathtt{J}^{i,1}_{\mathtt{P}_{w1}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \sum_i \left(\mathtt{J}^{i,M-1}_{\mathtt{P}_{w(M-1)}}\right)^\top \mathtt{J}^{i,M-1}_{\mathtt{P}_{w(M-1)}} \end{bmatrix}. \tag{16}$$

Cette structure reflète le fait que dans le cas où les points 3D sont figés, alors chaque caméra est indépendante.

$\mathtt{J}_{\mathtt{U}}^{\top}\mathtt{J}_{\mathtt{U}}$ est également bloc-diagonale :

$$\mathtt{J}_{\mathtt{U}}^{\top}\mathtt{J}_{\mathtt{U}} = \begin{bmatrix} \sum_j \left(\mathtt{J}_{\mathbf{u}_0^w}^{0,j}\right)^{\top} \mathtt{J}_{\mathbf{u}_0^w}^{0,j} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sum_j \left(\mathtt{J}_{\mathbf{u}_1^w}^{1,j}\right)^{\top} \mathtt{J}_{\mathbf{u}_1^w}^{1,j} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \sum_j \left(\mathtt{J}_{\mathbf{u}_{N-1}^w}^{(N-1),j}\right)^{\top} \mathtt{J}_{\mathbf{u}_{N-1}^w}^{(N-1),j} \end{bmatrix}. \tag{17}$$

Cette structure reflète le fait que dans le cas où les caméras sont figées, alors chaque point 3D est indépendant.

L'utilisation du complément de Schur permet de résoudre efficacement le système (15). En introduisons les notations suivantes, $\mathtt{H}_{\mathtt{PP}} = \mathtt{J}_{\mathtt{P}}^{\top}\mathtt{J}_{\mathtt{P}}$, $\mathtt{H}_{\mathtt{UU}} = \mathtt{J}_{\mathtt{U}}^{\top}\mathtt{J}_{\mathtt{U}} + \lambda\mathtt{Id}$, $\mathtt{H}_{\mathtt{UP}} = \mathtt{J}_{\mathtt{U}}^{\top}\mathtt{J}_{\mathtt{P}} = \mathtt{H}_{\mathtt{UP}}^{\top}$, $\mathbf{b}_{\mathtt{P}} = \mathtt{J}_{\mathtt{P}}^{\top}\mathbf{r}$ et $\mathbf{b}_{\mathtt{U}} = \mathtt{J}_{\mathtt{U}}^{\top}\mathbf{r}$, (15) se réécrit :

$$\begin{bmatrix} \mathtt{H}_{\mathtt{PP}} + \lambda\mathtt{Id} & \mathtt{H}_{\mathtt{UP}}^{\top} \\ \mathtt{H}_{\mathtt{UP}} & \mathtt{H}_{\mathtt{UU}} + \lambda\mathtt{Id} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_{\mathtt{P}} \\ \boldsymbol{\delta}_{\mathtt{U}} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mathtt{P}} \\ \mathbf{b}_{\mathtt{U}} \end{bmatrix}, \tag{18}$$

donnant lieu aux deux équations suivantes :

$$\left(\mathtt{H}_{\mathtt{PP}} + \lambda\mathtt{Id}\right)\boldsymbol{\delta}_{\mathtt{P}} + \mathtt{H}_{\mathtt{UP}}^{\top}\boldsymbol{\delta}_{\mathtt{U}} = \mathbf{b}_{\mathtt{P}}. \tag{19}$$

et

$$\mathtt{H}_{\mathtt{UP}}\boldsymbol{\delta}_{\mathtt{P}} + \left(\mathtt{H}_{\mathtt{UU}} + \lambda\mathtt{Id}\right)\boldsymbol{\delta}_{\mathtt{U}} = \mathbf{b}_{\mathtt{U}}, \tag{20}$$

L'équation (20) se réécrit :

$$\boldsymbol{\delta}_{\mathtt{U}} = \left(\mathtt{H}_{\mathtt{UU}} + \lambda\mathtt{Id}\right)^{-1}\left(\mathbf{b}_{\mathtt{U}} - \mathtt{H}_{\mathtt{UP}}\boldsymbol{\delta}_{\mathtt{P}}\right). \tag{21}$$

En injectant (21) dans (19), nous obtenons :

$$\underbrace{\left(\mathtt{H}_{\mathtt{PP}} + \lambda\mathtt{Id} - \mathtt{H}_{\mathtt{UP}}^{\top}\left(\mathtt{H}_{\mathtt{UU}} + \lambda\mathtt{Id}\right)^{-1}\mathtt{H}_{\mathtt{UP}}\right)}_{\mathtt{A}}\boldsymbol{\delta}_{\mathtt{P}} = \underbrace{\mathbf{b}_{\mathtt{P}} - \mathtt{H}_{\mathtt{UP}}^{\top}\left(\mathtt{H}_{\mathtt{UU}} + \lambda\mathtt{Id}\right)^{-1}\mathbf{b}_{\mathtt{U}}}_{\mathbf{d}}. \tag{22}$$

La matrice $\mathtt{A} = \mathtt{H}_{\mathtt{PP}} + \lambda\mathtt{Id} - \mathtt{H}_{\mathtt{UP}}^{\top}\left(\mathtt{H}_{\mathtt{UU}} + \lambda\mathtt{Id}\right)^{-1}\mathtt{H}_{\mathtt{UP}}$ s'appelle le complément de Schur. Remarquons que le système linéaire précédent (22) n'est plus que de taille $6M \times 6M$, sa résolution est donc beaucoup plus efficace que (15). Ainsi au lieu de résoudre (15) directement, il est beaucoup plus efficace de découper les choses en deux étapes :

1. Calculer $\boldsymbol{\delta}_{\mathtt{P}}$ en résolvant (22),
2. Calculer $\boldsymbol{\delta}_{\mathtt{U}}$ en calculant (21). Remarquons que $\mathtt{H}_{\mathtt{UU}}$ étant bloc-diagonale, $\mathtt{H}_{\mathtt{UU}}^{-1}$ est également bloc-diagonale et peut être calculée très efficacement (inversion de matrices 3×3).

En ce qui concerne la résolution du système linéaire (22), il faut :

1. Construire $\mathtt{A}$ et $\mathbf{d}$
2. Calculer $\boldsymbol{\delta}_{\mathtt{P}}$ en résolvant numériquement le système linéaire (22).

La construction de $\mathtt{A}$ et $\mathbf{d}$ peut se faire de plusieurs manières :

— de la plus simple (mais encore coûteuse en mémoire) où on calcule et **stocke les blocs** de $\mathtt{H}_{\mathtt{UU}}$ et $\mathtt{H}_{\mathtt{PP}}$, et où on calcule et **stocke** $\mathtt{H}_{\mathtt{UP}}$

— à la plus complexe (mais la plus économe en mémoire) où on calcule et **stocke les blocs** de $\mathtt{H}_{\mathtt{UU}}$ et $\mathtt{H}_{\mathtt{PP}}$ mais où on **ne stocke pas** $\mathtt{H}_{\mathtt{UP}}$.

Ici nous commencerons par la première approche et laisserons la deuxième en option si vous avez le temps. Remarquons que si l'objectif principal était d'implémenter la méthode la plus économe en mémoire, il serait sage de commencer par implémenter la première méthode pour s'assurer que tout fonctionne avant de passer à l'optimisation mémoire.