Une longueur de pas optimale au sens de l'erreur de reprojection algébrique pour l'ajustement de faisceaux
Algebraic Line Search for Bundle Adjustment

J. Michot[1] A. Bartoli[2] F. Gaspard[1]

[1] CEA, LIST, Laboratoire Systèmes de Vision Embarqués,
Boîte Courrier 94, Gif-sur-Yvette, F-91191 France ;
{julien.michot,francois.gaspard}@cea.fr
[2] LASMEA, UMR 6602 CNRS/UBP, Clermont-Ferrand, France
adrien.bartoli@gmail.com

Résumé

L'ajustement de faisceaux repose sur des techniques de minimisation non linéaires au sens des moindres carrés, comme Levenberg-Marquardt ou Gauss-Newton. Ils fournissent itérativement des déplacements relatif dans l'espace des paramètres à optimiser. Les techniques de recherche de la longueur de pas (Line Search) visent à déterminer un pas de déplacement efficace pour chaque déplacement. Nous proposons dans cet article une nouvelle technique ad hoc de Line Search pour l'ajustement de faisceaux. L'idée principale est de déterminer la longueur idéale du pas de déplacement par une approximation de l'erreur de reprojection, en substituant la distance euclidienne par la distance algébrique. Notre méthode est comparée avec plusieurs algorithmes de minimisation non-linéaires dans différentes configurations, sur des données synthétiques et réelles. Elle améliore la convergence de la minimisation de manière efficace et rapide à chaque itération et réduit le temps de minimisation de 10% en moyenne.

Mots Clef

Line Search, moindres carrés non-linéaires, ajustement de faisceaux, *Structure-from-Motion*.

Abstract

Bundle Adjustment is based on nonlinear least squares minimization techniques, such as Levenberg-Marquardt and Gauss-Newton. It iteratively computes local parameters increments. Line Search techniques aims at providing an efficient magnitude for these increments, called the step length. In this paper, a new ad hoc Line Search technique for solving bundle adjustment is proposed. The main idea is to determine an efficient step length using an approximation of the cost function based on an algebraic distance. Our method is compared to different nonlinear optimization algorithms and Line Search techniques under several conditions, on real and synthetic data. The method improves the minimization process, significantly decreasing the reprojection error at each iteration. It reduces the global computation time by 10% in average.

Keywords

Line Search, non-linear square roots, bundle adjustment, Structure-from-Motion.

1 Introduction

Depuis quelques décennies les travaux de recherche dans le domaine de la vision par ordinateur se sont fortement orientés vers les problèmes de reconstruction 3D. Dans de nombreux problèmes liés à l'image ou à la vidéo, le calcul de la structure 3D de la scène filmée ainsi que de la localisation de la caméra est essentiel. Ce calcul, effectué à partir des images 2D, est connu sous le nom de *Structure-from-Motion* (SfM). Une introduction au problème de la reconstruction 3D à partir d'une séquence d'images est donné par Pollefeys *et al.*, voir [17].

L'ajustement de faisceaux est une optimisation non-linéaire couramment employée dans les problèmes de reconstruction pour raffiner un modèle initial. Il s'agit d'optimiser un ensemble de paramètres modélisant la structure 3D d'une scène (primitives 3D), les positions et orientations des caméras (paramètres extrinsèques), et éventuellement les paramètres intrinsèques des caméras (focale, point principal, etc.).

Habituellement, les techniques de minimisation de Gauss-Newton, comme le Levenberg-Marquardt [8, 11], sont utilisées pour minimiser l'erreur de reprojection. Ce critère est défini par la distance géométrique entre les points détectés dans les images et leur reprojection [22]. Depuis quelques années des stratégies ont été développées afin de rendre cette optimisation non linéaire rapide et efficace. Il est même aujourd'hui possible d'employer des ajustements de faisceaux locaux dans des application temps-réel [13]. Nous proposons une nouvelle stratégie pour réduire le temps de calcul, afin de réaliser plus d'itérations et être ainsi plus précis.

Un des inconvénients de la méthode Levenberg-Marquardt est qu'elle ne fournit qu'une direction de déplacement dans l'espace des paramètres, mais pas nécessairement la longueur optimale du pas à réaliser. L'objectif des techniques de *Line Search* est donc de déterminer la longueur de pas la plus efficace, à chaque itération de l'algorithme. De nombreux travaux ont été effectués sur cette recherche (entre autres [1, 9, 12, 16, 3]). La plupart de ces méthodes sont itératives et sont particulièrement coûteuses en temps de calcul.

Nous proposons dans cet article une nouvelle technique de Recherche de la longueur de pas non itérative, pouvant être rapidement implantée dans un ajustement de faisceaux traditionnel. En approximant l'erreur de reprojection par une distance algébrique, nous sommes en mesure de calculer une longueur efficace du pas de déplacement. Nos expérimentations montrent que cette longueur, optimale en distance algébrique, est de plus performante en distance euclidienne. Nous présentons deux variations : l'ALS (*Algebraic Line Search*) global et l'ALS à deux dimensions. La première calcule une longueur de pas globale, pour tous les paramètres, en résolvant un polynôme de degré 3. La seconde définit deux longueurs de pas : une pour les paramètres des caméras et une autre pour la scène. Celle-ci requière la résolution d'un polynôme de degré 5. Les routines proposées sont efficaces, rapides et elles peuvent être employées dans différents contextes : SLAM visuels, calcul de pose, triangulation ou tout autre problème nécessitant la minimisation de l'erreur de reprojection.

Organisation de l'article. La section 2 définie les problèmes de *Structure-from-Motion* et d'ajustement de faisceaux. Nous décrivons ensuite notre approche, la longueur de pas algébrique. Enfin, la dernière section présente les résultats de la méthode sur des données réelles et simulées.

Notations. Nous adoptons les notations suivantes : les scalaires sont en italique, *e.g.* x, les vecteur en gras, *e.g.* $\mathbf{p}$ $\mathbf{P}$ et les matrices en caractères sans sérif *e.g.* M. I_3 est la matrice identité 3×3. $d(\mathbf{q}, \mathbf{q}')$ est la distance Euclidienne, non linéaire en coordonnées homogènes puisque $d^2(\mathbf{q}, \mathbf{q}') = \|\Psi(\mathbf{q}) - \Psi(\mathbf{q}')\|^2$ avec $\Psi(\mathbf{q}) = \frac{1}{q_3}\begin{pmatrix} q_1 \\ q_2 \end{pmatrix}$.

2 Contexte et travaux antérieurs

2.1 Formulation du problème

Nous considérons le problème de *Structure-from-motion*.

***Structure-from-Motion*.** Le problème de reconstruction consiste à déterminer de manière précise toutes les positions et orientations des caméras et l'ensemble des points 3D $\mathbf{Q}j$ de la scène à partir d'un ensemble de vues (images 2D). Les matrices de projection des caméras sont notées P_i, avec $i = 1 \dots n$. Elles peuvent être décomposées en métrique en $\mathsf{P}_i^M = \mathsf{P}_i\mathsf{H} = \mathsf{K}_i(\mathsf{R}_i|\mathbf{t}_i)$, où K_i contient les paramètres intrinsèques de la caméra i et R_i et $\mathbf{t}_i$ représentent l'orientation et la position des caméras dans le repère global. H est une homographie permettant de passer d'un espace projectif à un espace métrique. Nous considérons les cas non calibrés et calibrés. Pour ce dernier, nous supposons que les matrices de paramètres intrinsèques des caméras sont constantes, connues et identiques : $\mathsf{K}_i = \mathsf{K}$. Le vecteur $\mathbf{x}$ de paramètres à optimiser comporte donc les points $\mathbf{Q}_j$ et soit les matrices P_i dans le cas non-calibré, soit les vecteurs $\mathbf{R}_i$ et $\mathbf{t}_i$ dans les ajustements calibrés.

Fonctions de coût. Dans l'ajustement de faisceaux, la fonction de coût $f(\mathbf{x})$ associée à l'algorithme d'optimisation calcule l'erreur de reprojection. Il s'agit d'une minimisation au sens des moindres carrés de la distance entre les observations 2D (mesures sur les images) et les reprojections théoriques. La caractérisation de la distance peut varier. La fonction classique modélisant l'erreur géométrique est la suivante :

$$\varepsilon(\mathbf{x}) = \sum_{i,j} v_{ij}\, d^2\left(\mathbf{q}_{ij}, \mathbf{x}\right), \tag{1}$$

où $\mathbf{q}_{ij}$ est l'observation du point 3D $\mathbf{Q}_j$ sur l'image (caméra) P_i et $v_{ij} = 1$ si l'observation existe et 0 dans le cas contraire.

Une autre fonction de coût proposé dans la littérature [6] est basée sur la distance algébrique suivante :

$$\tilde{d}(\mathbf{q}, \mathbf{q}') = \|\mathsf{S}[\mathbf{q}]_\times \mathbf{q}'\|, \tag{2}$$

avec $\mathsf{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ et $[\mathbf{q}]_\times$ est la matrice associée au produit vectoriel. Il est convenu [6, 7] que sous une normalisation appropriée des données, les fonctions de coût algébriques donnent des résultats satisfaisants, eu égard au fait que cette distance ne possède pas réellement de signification géométrique ou statistique. La fonction de coût algébrique s'écrit alors :

$$\tilde{\varepsilon}(\mathbf{x}) = \sum_{i,j} v_{ij} \left\|\mathsf{S}[\mathbf{q}_{ij}]_\times \mathbf{x}\right\|^2. \tag{3}$$

2.2 Séquentiel et *batch Structure-from-Motion*

On peut distinguer deux principales configurations de SfM en fonction du schéma d'obtention des données images : le *batch* et le séquentiel SfM.

***Structure-from-Motion* sequentiel.** Avec le SfM séquentiel, les images sont traitées séquentiellement, l'une après l'autre. Le *framework* classique de SfM séquentiel comporte les étapes suivantes :

Détection et appariement des points d'intérêt. Pour reconstruire une scène filmée et localiser la caméra, il est nécessaire dans un premier temps de relier les images entre-elles, au moyen des matrices fondamentales par exemple. Cette étape est réalisée à l'aide d'un appariement de points d'intérêt dans les premières images.

Extraction des images clés. Le déplacement d'une caméra entre deux vues est généralement trop limité pour pouvoir appliquer les algorithmes de reconstruction. Il faut donc sélectionner un ensemble d'images "clés", distantes les unes

par rapport autres mais partageant un nombre suffisant de points d'intérêt.

Reconstruction initiale et localisation. Avec l'ensemble des points d'intérêt et des images clés précédemment extraits, il est possible de déterminer la structure initiale de la scène 3D et de la trajectoire de la caméra. Ce premier modèle est ensuite raffiné au moyen d'un ajustement de faisceaux [17].

Resection-Intersection. Pour chaque nouvelle image clé détectée, le déplacement de la caméra peut être identifié grâce aux points 3D précédents (resection). Et les nouveaux points 3D peuvent être calculés avec ce nouveau déplacement (intersection). Un raffinement local est ensuite périodiquement appliqué sur les dernières poses de la caméra et sur la scène pour réduire les erreurs cumulées. De plus, il est nécessaire de supprimer les *outliers* des données en utilisant par exemple un algorithme de RANSAC [2].

***Batch Structure-from-Motion*.** Contrairement aux SfM séquentiels, le *Batch SfM* considère l'ensemble des données d'entrée (toutes les images) pour reconstruire la scène et la trajectoire de la caméra.

Les méthodes de factorisation classiques sont basées sur une décomposition en valeur singulières (SVD) de la matrice de mesures [21, 19], mais sont particulièrement sensibles aux manques de données et aux outliers. Les méthodes de *Factorizations* hiérarchiques ou contraintes [23, 20] sont plus robustes sur ces critères. Elles considèrent non plus l'ensemble de la matrice de mesures, mais des sous-ensembles complets pour retrouver la structure de la scène.

2.3 Ajustement de faisceaux

Pour optimiser un premier modèle initial de scène et de localisation de caméra, l'ajustement de faisceaux effectue une minimisation au sens des moindres carrés.

Optimisation non-linéaire. Nous considérons le problème suivant :

$$\min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}). \tag{4}$$

où

$$f(\mathbf{x}) = \frac{1}{2}\left\|d(\mathbf{x})\right\|^2.$$

et $f : \mathbb{R}^p \to \mathbb{R}$ est une fonction non-convexe, continue et dérivable deux fois. On recherche alors un optimum $\mathbf{x}^*$ de f, tel que :

$$\nabla f(\mathbf{x}^*) = 0. \tag{5}$$

Le problème est habituellement résolu par un algorithme itératif de minimisation qui génère une série de déplacements $\mathbf{x}_0, \mathbf{x}_1, \dots \mathbf{x}_k$. La série converge vers un minimum local $\mathbf{x}^*$: $f(\mathbf{x}_{k\to\infty}) \to f(\mathbf{x}^*)$. La séquence $\mathbf{x}_k$ est déterminée par :

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \boldsymbol{\delta}_k, \tag{6}$$

où $\boldsymbol{\delta}_k$ est la direction proposée par l'optimisation non contrainte à l'itération k et α_k la longueur du pas du déplacement à entreprendre dans cette direction.

Plusieurs stratégies ont été proposées afin de déterminer le minimum global d'une fonction non-linéaire (méthodes de (quasi-)Newton [9], de descente de gradient, Levenberg-Marquardt [7, 8, 14], des régions de confiance de Powell [18, 16, 10], ...)

Levenberg-Marquardt. La technique de minimisation Levenberg-Marquardt (LM) est une combinaison des méthodes de descente de gradient et de Gauss-Newton. L'algorithme détermine le déplacement $\boldsymbol{\delta}_k$ en résolvant l'équation suivante :

$$\left(\mathrm{J}_k^\top \mathrm{J}_k + \lambda_k \mathrm{I}_3\right) \boldsymbol{\delta}_k = -\mathbf{g}_k \tag{7}$$

où $\mathrm{J}_k = \frac{\partial d}{\partial \mathbf{x}_k}(\mathbf{x}_k)$, $\mathbf{g}_k = \mathrm{J}_k^\top d(\mathbf{x}_k)$ et λ_k est le paramètre de *damping*. C'est ce dernier paramètre qui autorise l'algorithme LM à choisir le comportement de minimisation le plus adapté, suivant le contexte de la solution courante, entre la Descente de Gradient et la méthode de Gauss-Newton.

Plusieurs heuristiques de mise à jour de ce paramètre existent. Lors de nos expérimentations, nous avons choisi les règles décrites par Hartley et Zisserman dans [7] ainsi que les méthodes proposées par Nielsen dans [14].

Recherche de la longueur de pas (*Line Search*). Puisque la minimisation de Levenberg-Marquardt ne s'attache pas à calculer la distance optimale du déplacement à réaliser suivant la direction $\boldsymbol{\delta}_k$, les techniques de *Line Search* tentent de déterminer cette longueur de pas minimisant le plus efficacement possible la fonction objective. Il existe deux types de *Line Search*, les méthodes exactes et inexactes.

Les méthodes exactes sont définies par :

$$\alpha_k = \arg\min_{\alpha_k > 0} f(\mathbf{x}_k + \alpha_k \boldsymbol{\delta}_k). \tag{8}$$

Malheureusement la majorité des problèmes de vision possèdent des fonctions objectives f non linéaires et l'équation (8) devient alors difficile à résoudre. De ce cas, il est beaucoup plus intéressant de faire appel aux algorithmes de recherche inexacte de la longueur de pas.

Pour trouver la meilleure longueur de pas suivant une direction définie par un algorithme de minimisation non-linéaire, la plupart des méthodes de *Line Search* inexactes s'initialisent sur une première valeur du pas et raffinent ensuite cette proposition itérativement à l'aide de différents critères, comme par exemple les critères de Goldstein ou de Wolfe [15].

Un des algorithmes les plus courants est l'algorithme de *Backtracking*. Initialisé sur une première valeur de pas, il génère et évalue ensuite à chaque itération une proposition, et fournit en fin d'execution la meilleure proposition minimisant l'erreur finale.

De nombreuses techniques de *Line Search*, comme notamment [1, 12, 16] ou [3], dérivent de la technique dite du *Backtracking*. Après avoir réduit un intervalle de recherche, elles se proposent de chercher la meilleur longueur de pas à l'aide de différentes heuristiques. Frandsen

et al. [3] proposent deux techniques de *Line Search* itératives (*Exact* et *Soft LS*) grâce à une interpolation quadratique de la fonction objective. La méthode proposée s'avère être très efficace mais aussi très lente étant donnée la discrétisation de l'espace. Liu et Nocedal [9] font appel aux critères de Wolfe dans le cadre d'une minimisation *quasi*-Newtonienne limitée (*L-BFGS*), Hager et Zhang dans [4] étudient la convergence de la méthode de gradient conjugué munie d'un *Line Search* basé sur les conditions de Wolfe. Nous renvoyons le lecteur au chapitre de Nocedal et Wright [15, Chap 3] sur les méthodes classiques de *Line Search* couramment employées en optimisation nonlinéaire.
Ces différentes techniques de recherche de la longueur de pas sont itératives et, puisque pour chaque proposition la fonction de coût est évaluée, elles ne peuvent être employées dans des algorithmes à contraintes temps-réel. La complexité de la fonction à minimiser dans l'ajustement de faisceaux dépend du nombre de points 3D de la scène ainsi que des caméras du modèle à raffiner. En pratique, leur nombre est important, ces méthodes sont par conséquent peu utilisées dans les ajustement de faisceaux temps-réel.

3 *Line Search* algébrique (ALS)

Dans cette partie nous présentons notre contribution : utiliser la distance algébrique dans une technique de Recherche de la longueur de pas.
Notre idée **n'est pas** d'utiliser la distance algébrique en tant que fonction de coût de l'ajustement de faisceaux, nous gardons pour cela la distance géométrique. La distance algébrique est employée seulement pour trouver une longueur de pas efficace.
Nous distinguons les ajustements calibrés et non calibrés. La principale différence réside dans la paramétrisation de la caméra. Soit on intègre la position et l'orientation de la caméra dans le cas calibré, soit on utilise directement les éléments des matrices P_i dans le cas non calibré.

3.1 *Line Search* algébrique non-calibré

Nous étudions deux approches différentes de l'ALS. L'ALS Global est une technique de *Line Search* qui vise à rechercher une longueur de pas efficace pour l'ensemble des paramètres (caméra et scène). L'ALS à deux dimensions détermine quant-à-lui deux longueurs de pas, l'une pour les caméras et l'autre pour la structure de la scène.

Line Search algébrique Global (G-ALS). Une fois que l'algorithme d'optimisation nous fournit une direction ($\boldsymbol{\delta} = \left\{\boldsymbol{\delta}_{\mathsf{P}_i} = vect(\Delta_{P_i}), \boldsymbol{\delta}_{\mathbf{Q}_j}\right\}$), il reste à définir la distance de déplacement α le long de cette direction. En considérant un nouveau déplacement (équation (6)), l'équation (3) devient

$$\varepsilon_{alg}(\alpha) = \sum_{i,j} \left\|\mathsf{S}[\mathbf{q}_{ij}]_\times(\mathsf{P}_i + \alpha\Delta_{P_i})(\mathbf{Q}_j + \alpha\boldsymbol{\delta}_{\mathbf{Q}_j})\right\|^2 \tag{9}$$

Puisque nous cherchons le pas α dans la direction $\boldsymbol{\delta}$ qui minimise $\varepsilon_{alg}(\alpha)$, les valeurs optimales sont données par les racines du polynôme (10), figure 1.
En approximant ainsi la fonction classique de reprojection par une distance algébrique (approximation quadratique) nous simplifions le problème et le rendons résolvable littéralement. La résolution du polynôme (10) de degré 3 en α nous fournit donc une ou trois solutions et, dans la majorité des cas, nous obtenons une seule solution réelle.
Cette méthode de calcul du déplacement - optimal en distance algébrique - est simple, rapide et approxime relativement bien la fonction de coût géométrique. Contrairement aux méthodes classiques de *Line Search*, aucune itération n'est nécessaire, ce qui rend cette technique rapide.

Line Search algébrique à deux dimensions (Two-way ALS). Étant donné la nature différente des deux types de données du modèle (les caméras et les points 3D de la scène), il semble intéressant de séparer la recherche de la longueur optimale du pas. Nous cherchons dès lors deux déplacements α_{cam} et α_{str}, les pas optimaux pour respectivement les déplacements des caméras et de la scène (points 3D). Cela revient à déterminer le minimum $\{\alpha^*_{cam}, \alpha^*_{str}\}$ de la fonction objective algébrique :

$$\begin{array}{l}\varepsilon_{alg}(\{\alpha^*_{cam}, \alpha^*_{str}\}) = \\ \sum_{i,j} \left\|\mathsf{S}\left[\mathbf{q}_{ij}\right]_\times \left(\mathsf{P}_i + \alpha^*_{cam}\Delta_{P_i}\right)\left(\mathbf{Q}_j + \alpha^*_{str}\boldsymbol{\delta}_{\mathbf{Q}_j}\right)\right\|^2\end{array} \tag{11}$$

La recherche du minimum global $\{\alpha^*_{cam}, \alpha^*_{str}\}$ peut être effectuée de manière rapide. Les solutions de α_{cam} sont les racines d'un polynôme de degré 5, et α_{str} se déduit de α_{cam}. Les détails des calculs sont fournis en Annexe 1.

3.2 *Line Search* algébrique calibré

Avec des ajustements métriques, les paramètres des caméras ne sont plus définis par la matrice P_i, mais par des informations d'orientation et de position : $\boldsymbol{\delta} = \left\{\boldsymbol{\delta}_{r_i}, \boldsymbol{\delta}_{t_i}, \boldsymbol{\delta}_{Q_j}\right\}$. Nous utilisons un paramétrisation en petits angles d'Euler pour les déplacements d'orientation. Les orientations globales de la caméra sont représentées par des matrices de rotation $\mathsf{R}_i \in SO(3)$. Notre règle de mise à jour est donc $\mathsf{R}_i^{k+1} = \mathsf{R}_i^k[\boldsymbol{\delta}_{r_i}]$, où $[\mathbf{v}]$ transforme le vecteur d'angles $\mathbf{v}$ en une matrice de rotation. Dans ce contexte et pour conserver la forme de l'équation (9), nous définissons $\Delta^k_{P_i}$ de l'équation de déplacement $\mathsf{P}_i^{k+1} = \mathsf{P}_i^k + \alpha\Delta^k_{P_i}$ tel que :

$$\Delta^k_{P_i} \approx \mathsf{K}\mathsf{R}_i^k([\boldsymbol{\delta}_{r_i^k}]_\times|\boldsymbol{\delta}_{t_i^k}), \tag{12}$$

puisque

$$\begin{array}{rcl}\mathsf{P}_i^{k+1} & = & \mathsf{P}_i^k \begin{pmatrix}[\alpha\boldsymbol{\delta}_{r_i^k}] & \alpha\boldsymbol{\delta}_{t_i^k} \\ \mathbf{0} & 1\end{pmatrix} \\ & = & \mathsf{K}(\mathsf{R}_i^k[\alpha\boldsymbol{\delta}_{r_i^k}]|\mathbf{t}_i^k + \alpha\mathsf{R}_i^k\boldsymbol{\delta}_{t_i^k}),\end{array}$$

et en utilisant l'approximation classique des petits angles $[\mathbf{v}] \approx \mathrm{I}_3 + [\mathbf{v}]_\times$, nous obtenons

$$\begin{array}{rcl}\mathsf{P}_i^{k+1} & \approx & \mathsf{K}(\mathsf{R}_i^k + \alpha\mathsf{R}_i^k[\boldsymbol{\delta}_{r_i^k}]_\times|\mathbf{t}_i^k + \alpha\mathsf{R}_i^k\boldsymbol{\delta}_{t_i^k}) \\ & = & \mathsf{P}_i^k + \alpha\mathsf{K}\mathsf{R}_i^k([\boldsymbol{\delta}_{r_i^k}]_\times|\boldsymbol{\delta}_{t_i^k}).\end{array}$$

$$\frac{\partial\tilde{\varepsilon}}{\partial\alpha} = \sum_{i,j} 2(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}\boldsymbol{\delta}_{Q_j})^\top(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}\boldsymbol{\delta}_{Q_j})\alpha^3 + \sum_{i,j} 3\left(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,(\Delta_{P_i}\mathbf{Q}_j + \mathsf{P}_i\boldsymbol{\delta}_{Q_j})\right)^\top \mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}\boldsymbol{\delta}_{Q_j}\alpha^2$$
$$+\sum_{i,j}\left(\left(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\mathsf{P}_i\mathbf{Q}_j\right)^\top \mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}\boldsymbol{\delta}_{Q_j} + \left(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,(\Delta_{P_i}\mathbf{Q}_j + \mathsf{P}_i\boldsymbol{\delta}_{Q_j})\right)^2\right)\alpha + \sum_{i,j}\left(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\mathsf{P}_i\mathbf{Q}_j\right)^\top \mathsf{S}\,[\mathbf{q}_{ij}]_\times\,(\Delta_{P_i}\mathbf{Q}_j + \mathsf{P}_i\boldsymbol{\delta}_{Q_j}) \tag{10}$$

Figure 1 – Le polynôme de degré 3 utilisé par le Global ALS

Cette approximation de $\Delta_{P_i}^k$ nous permet de conserver les équations (9) et (11) dans le cas calibré. Nous appliquons ensuite les même procédures de calcul de la longueur de pas que dans le cas non calibré.
Un algorithme général de Structure-from-Motion pour des ajustements calibré ou non, est donné en Table 1.

Heuristiques. Il est important de noter que la distance algébrique (qui n'est pas réellement une distance) n'est qu'une approximation de la distance euclidienne puisque :

$$d_{euc}(\mathbf{X}, \widetilde{\mathbf{X}}) = \frac{d_{alg}(\mathbf{X}, \widetilde{\mathbf{X}})}{w\widetilde{w}} \tag{13}$$

avec $\mathbf{X}^\top = (x, y, w)$ et $\widetilde{\mathbf{X}}^\top = (\widetilde{x}, \widetilde{y}, \widetilde{w})$.
En conséquence, les minima des fonctions de coût associées ne sont pas obligatoirement identiques. L'expérience montre cependant qu'avec une normalisation adaptée, les minima sont toutefois proches.
Nous avons donc expérimenté deux heuristiques (`G-ALS` et `T-ALS`) employant la distance algébrique. Ces deux heuristiques vérifient dans un premier temps que la simple distance unité n'est pas meilleure que la distance proposée par l'heuristique. En effet, cette distance unité représente le pas idéal de la minimisation de Gauss-Newton. Nous retrouvons ce comportement avec le LM lorsque le paramètre de damping est très faible, c'est-à-dire lorsque nous sommes près de la solution.
D'autre part, nous limitons le domaine de définition de la longueur du pas afin d'éviter les déplacements trop grands ou trop faibles : $\alpha \in [\alpha_{min}, \alpha_{max}]$ avec $0 < \alpha_{min} < \alpha_{max}$.
Line Search Algébrique Global (`G-ALS`) :
$\alpha = \arg\min_\alpha(\varepsilon_{euc}(\alpha = 1), \varepsilon_{euc}(\alpha = \alpha^*))$ où α^* est la longueur de pas calculée par l'ALS global.
Line Search Algébrique à deux dimensions (`T-ALS`) :
$\alpha = \arg\min_\alpha(\varepsilon_{euc}(\alpha = \{1, 1\}), \varepsilon_{euc}(\alpha = \{\alpha^*_{cam}, \alpha^*_{str}\}))$ Cette dernière heuristique reprend l'idée d'ALS en deux dimensions de l'équation (11).

4 Résultats expérimentaux

Dans cette partie nous présentons les résultats de notre méthode obtenus sur des données réelles et simulées.

4.1 Contexte de l'évaluation

L'ensemble de nos essais ont été réalisés sous MATLAB 7.5.0.

***Line Search* Algébrique** : intégration dans le framework [7, p.574]
Ajouter les étapes suivantes :
vii'. Calculer les coefficients des équations (9) (Global ALS) ou (16) (Two-way ALS, Annexe 1)
vii". Résoudre le polynôme, tester et conserver la meilleure distance pour le déplacement (en incluant le pas unité)
vii". Multiplier le déplacement $\boldsymbol{\delta} = (\boldsymbol{\delta}_a^\top, \boldsymbol{\delta}_b^\top)$ par la ou les distances ainsi calculées par l'ALS

Table 1 – Implémentation de notre *Line Search* Algébrique dans l'ajustement de faisceaux de [7, p.574] (Algorithm A4.1).

Une normalisation isotrope ([5]) a été appliquée sur chaque observation, puis rectifiée pour l'affichage des résultats (*RMS* : Erreur moyenne de reprojection).
Pour chaque jeu de données nous comparons notre technique de *Line Search* avec différents algorithmes de minimisation non-linéaires.

1. `BA-Classic` est une implémentation de l'ajustement de faisceaux de [7] utilisant Levenberg-Marquardt
2. `BA-Nielsen` est la technique proposée par Nielsen dans [14]. Celui-ci utilise aussi Levenberg-Marquardt mais avec notamment une règle distincte de mise à jour du paramètre de *damping*.
3. `BA-Powel-Lourakis` est un ajustement de faisceaux basé sur les régions de confiance [10].
4. `LS-FJNT` est l'algorithme de *Line Search* de [3] décrit en section 2.3.
5. `G-ALS` et `T-ALS` sont les heuristiques basées sur le *Line Search* Algébrique, décris aux sections 3.1 et 3.2.

4.2 Données simulées

Nous générons aléatoirement 500 points 3D dans un cube de 6 mètres, et 30 caméras positionnées autour à une distance de 20 mètres par rapport au centre du cube. Les caméras possèdent des paramètres intrinsèques identiques et connus (focale : 1000 pixels, au centre de l'image, les pixels sont carrés), nous appliquons ensuite les projections 2D sur des images (640×480).

L'ajout d'un bruit gaussien sur les points 3D et sur les poses des caméras nous fournit notre vecteur initial $\mathbf{x}_0$ de l'ajustement de faisceaux. La Figure 2(a) illustre la configuration de la scène simulée.

Les résultats (Figure 2) montrent que sur des données synthétiques, notre *Line Search* Algébrique choisit une longueur de pas de déplacement efficace puisque dès les premières itérations de l'algorithme de minimisation, l'erreur de reprojection est moins importante qu'avec les autres algorithmes. Il est intéressant de noter que l'ALS est d'autant plus efficace que nous sommes loin du minimum de la fonction de coût. On retrouve ici le fait que la partie GN du LM détermine déjà efficacement la longueur du pas, le pas unité. Comme prévu, la technique itérative `LS-FJNT` est beaucoup plus performante que les autres méthodes, mais elle est aussi beaucoup plus lente. Le gain moyen en RMS pour les 4 premières itérations est de 11% pour l'ALS global et 15% pour le `T-ALS`, avec un écart type de 4%.

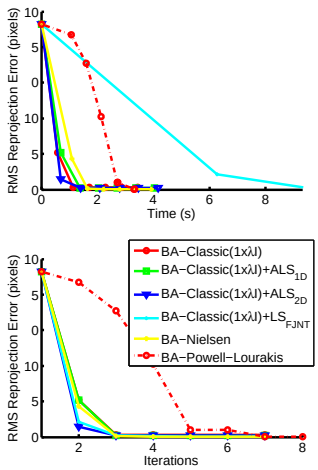


Figure 3 – RMS (pixels) de reprojection suivant le temps et les itérations, pour un ajustement projectif.

La Figure 3 présente une comparaison des différents algorithmes de minimisation avec et sans *Line Search* sur un ajustement projectif. La technique de l'ALS en deux dimensions (`T-ALS`) s'avère être très efficace puisque dès la première itération la minimisation est fortement améliorée. Le gain dans cet exemple est de plus supérieur aux résultats de la méthode `LS-FJNT`. Dans la majorité des tests effectués sur des données simulées, nos heuristiques de *Line Search* `G-ALS` et `T-ALS` ont montré de bons comportements. On peut noter de plus que l'ALS est aussi plus efficace sur les premières itérations de la minimisation puisque comme pour la plupart des techniques de *Line Search*, le ratio gain en RMS sur le temps de calcul additionnel est plus intéressant pour les premières itérations.

4.3 Données réelles

Les jeux de données réelles ont été obtenus à l'aide une caméra de type *pinhole* placée sur un véhicule se déplaçant dans une ville (Figure 5(a)) sur 500 mètres. Un exemple de reconstruction 3D est présenté en Figure 4.

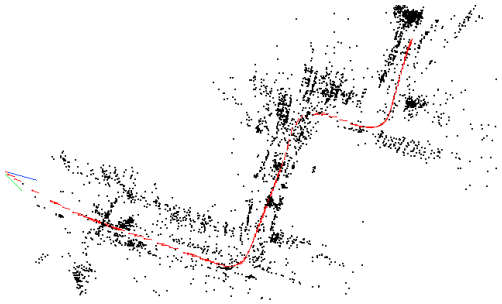
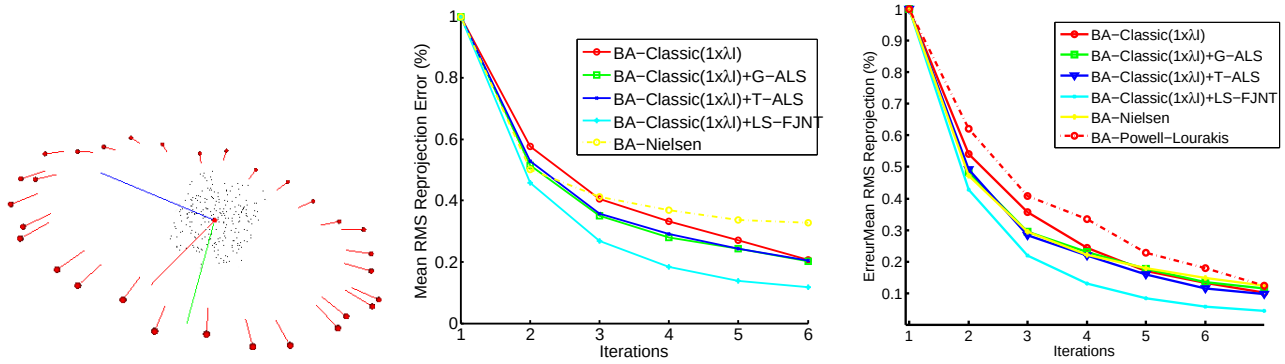


Figure 4 – Reconstruction 3D de la ville et localisation du véhicule.

Le calcul initial (précédant l'optimisation) des poses des caméras et de la structure 3D de la scène a été effectué par la méthode du SLAM incrémental temps réel de Mouragnon-Lhuiller [13].
Nous appliquons ensuite un ajustement de faisceaux global, c'est-à-dire sur toutes les données (150 caméras et 8000 points 3D), ou incrémental, i.e. seulement sur les N dernières données (les $N = 15$ plus récentes caméras, avec les $M = 300$ derniers points 3D).
Avec des ajustements de faisceaux incrémentaux, l'erreur de reprojection est raffinée au fur et à mesure des images clés. Comme le montre la figure 5(c), dans ce type de raffinement les techniques de *Line Search* ne semblent pas être adaptées. En effet, le Levenberg-Marquardt acquière dans ces conditions un comportement de type Gauss-Newton puisque l'état courant périodiquement rapproché du minimum. Comme mentionné plus haut, l'efficacité de l'ALS est d'autant plus importante que le LM prend le comportement d'une descente de gradient.
Lorsque l'on considère des ajustements globaux (résultats présentés en Figure 5(b)), ou hiérarchiques, le *Line Search* Algébrique apporte réellement un gain à l'optimisation, la minimisation converge plus rapidement. Les heuristiques `G-ALS` et `T-ALS` définissent une longueur de pas relativement efficace et minimisent ainsi d'autant plus l'erreur de reprojection. Elles ont de plus souvent le même comportement que le `LS-FJNT` dans les ajustements métriques, mais sont bien plus rapides. Le *Line Search* Algébrique apporte en moyenne un gain de 10% sur le temps total de minimisation avec un écart type de 5%, pour les deux heuristiques sur nos jeux de données réelles.

5 Conclusion

Cet article propose une nouvelle approche au problème de *Line Search* pour l'ajustement de faisceaux. L'idée principale consiste à utiliser une fonction objective basée sur la distance algébrique seulement lors de la recherche de



(a) (b) (c)

Figure 2 – Résultats sur les données synthétiques (a). Les figures (b) et (c) montrent l'évolution des RMS moyens normalisés par rapport au RMS initial, sur 20 simulations, pour des ajustements projectifs (b) et métriques (c).

la longueur du pas. Nos expérimentations, conduites sur des données réelles et synthétiques, montrent qu'à chaque itération de l'algorithme de minimisation, nos heuristiques fournissent une distance de déplacement efficace pour un faible temps de calcul. Cette méthode peut être aisément appliquée aux problèmes minimisant l'erreur de reprojection, dans des ajustements de faisceaux globaux ou hiérarchiques. Les résultats montrent cependant que de meilleurs résultats sont obtenus dans les ajustements dont la solution initiale est loin de la solution du problème. Les futurs travaux s'attacheront à expérimenter cette approche en temps réel pour des applications de type SLAM.

Annexe 1

Nous présentons ici la résolution du T-ALS. Nous cherchons les distances optimales $\{\alpha^*_{cam}, \alpha^*_{str}\}$ pour les déplacements de la caméra Δ_P et de la structure de la scène $\boldsymbol{\delta}_Q$. L'erreur algébrique que nous minimisons est la suivante :

$$\tilde{\varepsilon}(\{\alpha_{cam}, \alpha_{str}\}) = \sum_{i,j} \left\| \mathsf{S}\,[\mathbf{q}_{ij}]_\times \left(\mathsf{P}_i + \alpha_{cam}\Delta_{P_i}\right)\left(\mathbf{Q}_j + \alpha_{str}\boldsymbol{\delta}_{Q_j}\right)\right\|^2. \tag{14}$$

Les dérivées partielles en α_{cam} et α_{str} nous donnent le système suivant :

$$\frac{\partial\tilde{\varepsilon}}{\partial\alpha_{str}} = 0 \quad \text{and} \quad \frac{\partial\tilde{\varepsilon}}{\partial\alpha_{cam}} = 0,$$

sous une forme développée :

$$\begin{array}{l} \sum_{i,j} 2a + 2c\alpha_{cam} + 2d\alpha^2_{cam} + 2\alpha_{str}\alpha_{cam}(2e + f\alpha_{cam}) + 2g\alpha_{str} = 0 \\ \sum_{i,j} 2b + 2c\alpha_{str} + 2e\alpha^2_{str} + 2\alpha_{str}\alpha_{cam}(2d + f\alpha_{str}) + 2h\alpha_{cam} = 0 \end{array} \tag{15}$$

avec

$$\begin{array}{l} a = \sum_{i,j}(\mathsf{P}_i\mathbf{Q}_j)^\top(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\mathsf{P}_i\boldsymbol{\delta}_{Q_j}) \\ b = \sum_{i,j}(\mathsf{P}_i\mathbf{Q}_j)^\top(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}\mathbf{Q}_j) \\ c = \sum_{i,j}(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\mathsf{P}_i\boldsymbol{\delta}_{Q_j})^\top(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}\mathbf{Q}_j) + (\mathsf{P}_i\mathbf{Q}_j)^\top(\Delta_{P_i}\boldsymbol{\delta}_{Q_j}) \\ d = \sum_{i,j}(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}\mathbf{Q}_j)^\top(\Delta_{P_i}\boldsymbol{\delta}_{Q_j}) \\ e = \sum_{i,j}(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\mathsf{P}_i\boldsymbol{\delta}_{Q_j})^\top(\Delta_{P_i}\boldsymbol{\delta}_{Q_j}) \\ f = \sum_{i,j}(\Delta_{P_i}\boldsymbol{\delta}_{Q_j})^\top(\Delta_{P_i}\boldsymbol{\delta}_{Q_j}) \\ g = \sum_{i,j}(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\mathsf{P}_i\boldsymbol{\delta}_{Q_j})^\top(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\mathsf{P}_i\boldsymbol{\delta}_{Q_j}) \\ h = \sum_{i,j}(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}Q)^\top(\mathsf{S}\,[\mathbf{q}_{ij}]_\times\,\Delta_{P_i}\mathbf{Q}_j) \end{array} \tag{16}$$

La résolution symbolique de ce système de polynômes sous MAPLE nous donne les solutions suivantes :

$$\{\alpha^*_{cam}, \alpha^*_{str}\} = \{p, \frac{-a - cp - dp^2}{g + 2ep + fp^2}\} \tag{17}$$

avec $p = RootsOf(F(X))$ et $F : \mathbb{R} \rightarrow \mathbb{R}$ est un polynôme de degré 5 :

$$\begin{array}{r} F(X) = (-hf^2 + d^2f)X^5 \\ +(3ed^2 - 4hef + cdf - bf^2)X^4 \\ +(-4he^2 - 4bef - 2hfg + 4cde + 2d^2g)X^3 \\ +(c^2e - 4heg - 4be^2 + 2ead - 2bfg - caf + 3cdg)X^2 \\ +(c^2g + 2dag - 4beg - fa^2 - hg^2)X \\ -bg^2 - ea^2 + cag \end{array}$$

Références

[1] M. Al-Baali and R. Fletcher. An efficient line search for nonlinear least squares. *J. Optim. Theory Appl.*, 48(3) :359–377, 1986.

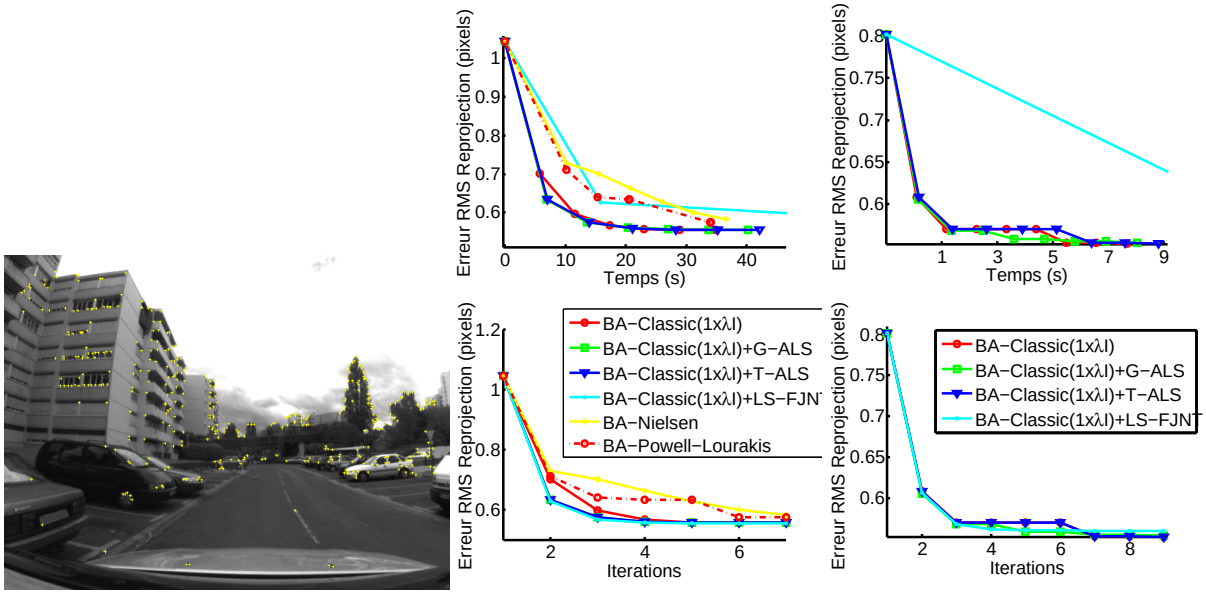
[2] M. A. Fischler and R. C. Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6) :381–395, 1981.

[3] P. E. Frandsen, K. Jonasson, H. B. Nielsen, and O. Tingleff. *Unconstrained Optimization*. 1999.

[4] W. Hager and H. Zhang. Algorithm 851 : CG DESCENT, a conjugate gradient method with guaranteed descent. *ACM Trans. Math. Softw.*, 32(1) :113–137, 2006.

[5] R. Hartley. In defence of the 8-point algorithm. In *ICCV '95 : Proceedings of the Fifth International Conference on Computer Vision*, page 1064, Washington, DC, USA, 1995. IEEE Computer Society.

[6] R. Hartley. Minimizing algebraic error. In *ICCV '98 : Proceedings of the Sixth International Conference on*



(a) (b) (c)

Figure 5 – Résultats sur des données réelles (a). Les figures (b) et (c) montrent l'évolution du RMS (pixels) par rapport au temps de calcul (en haut) et par rapport aux itérations (en bas), pour des ajustements globaux (b) et locaux (c)

Computer Vision, page 469, Washington, DC, USA, 1998. IEEE Computer Society.

[7] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, Cambridge, UK, first edition, 2003.

[8] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, Jul. 1944.

[9] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(3) :503–528, 1989.

[10] M. Lourakis and A. Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment ? *ICCV*, 2005.

[11] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11 :431, 441, 1963.

[12] J. Moré and D. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, 20(3) :286–307, 1994.

[13] E. Mouragnon, M. Lhuiller, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion. In *BMVC*, 2007.

[14] H. B. Nielsen. Damping parameter in marquardt's method. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, apr 1999.

[15] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research, New york, 1999.

[16] J. Nocedal and Y. Yuan. Combining trust region and line search techniques. Technical report, Advances in Nonlinear Programming, (Kluwer), 1992.

[17] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3) :207–232, 2004.

[18] M. J. D. Powell. A hybrid method for nonlinear equations. *Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz (Ed.)*, pages 87–114, 1970.

[19] P Sturm and B Triggs. A factorization based algorithm for multi-image projective structure and motion. pages 709–720. Springer-Verlag, 1996.

[20] J-P Tardif, A. Bartoli, M. Trudeau, and S. Guilbert, N.and Roy. Algorithms for batch matrix factorization with application to structure-from-motion. 2007.

[21] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography : a factorization method. *Int. J. Comput. Vision*, 9(2) :137–154, 1992.

[22] B. Triggs, P. F. Mclauchlan, R. Hartley, and A. W. Fitzgibbon. Bundle adjustment – a modern synthesis. *Lecture Notes in Computer Science*, 1883 :298+, January 2000.

[23] Bill Triggs. Linear projective reconstruction from matching tensors. *Image and Vision Computing*, 15 :617–625, 1997.