Interpolation et optimisation spatio-temporelle pour l'estimation de poses de caméra à obturateur déroulant

Philippe-Antoine Gohard, Bertrand Vandeportaele, Michel Devy

Interpolation et optimisation spatio-temporelle pour l'estimation de poses de caméra à obturateur déroulant

P-A. Gohard[1,2] B. Vandeportaele[1] M. Devy[1]

[1] LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France
[2] Innersense, Ramonville-Saint-Agne, France

{philippe-antoine.gohard, bertrand.vandeportaele, michel.devy}@laas.fr

Résumé

Les applications de réalité augmentée sur téléphones mobiles nécessitent l'adaptation des algorithmes de vision aux caméras à obturateur déroulant présentes sur ces appareils. Cet article propose un nouveau modèle géométrique de caméra à obturateur déroulant utilisant des B-Splines non-uniformes afin d'approximer fidèlement la trajectoire de la caméra au cours du temps. Une méthode d'ajustement de faisceaux intégrant ce modèle est décrite puis testée sur des jeux de données synthétiques. De plus, nous exposons un processus itératif d'optimisation permettant une répartition spatiale et temporelle des poses de contrôle de la courbe B-Spline adaptée à la dynamique de la trajectoire. Les améliorations apportées sont illustrées sur l'approximation de trajectoires simulées et de trajectoires réelles avec exploitation d'une vérité-terrain donnée par un système de Motion Capture.

Mots Clefs

Obturateur déroulant, B-Spline, Ajustement de faisceaux, Localisation et cartographie simultanées

1 Introduction

La réalité augmentée consiste en l'incrustation d'objets virtuels dans une séquence d'images en temps réel. Les applications de cette technologie sont multiples et touchent de plus en plus de domaines. La diffusion massive des téléphones mobiles équipés d'une caméra a permis le déploiement de services grand public exploitant la réalité augmentée. Cette technologie suppose la localisation de la caméra ainsi qu'une modélisation partielle de la scène observée afin de pouvoir afficher des objets virtuels de manière physiquement cohérente avec l'environnement.
Cependant, les caméras à bas-coût équipant les téléphones mobiles sont généralement dotées de propriétés spécifiques issues de leurs conceptions. Ces caméras, dites à obturateur déroulant, produisent des distorsions de l'image dans le cas d'un mouvement relatif entre la caméra et la scène. Ces distorsions, si elles ne sont pas prises en compte dans la modélisation de la caméra, perturbent les algorithmes nécessaires à la réalité augmentée.

1.1 État de l'art

Les algorithmes de cartographie et localisation simultanées (SLAM) sont largement utilisés dans le contexte de la réalité augmentée afin d'estimer la position et l'orientation d'une caméra en mouvement dans une scène réelle dont le modèle est également estimé par le SLAM. Ces informations sont utilisées pour incruster ou projeter des objets virtuels dans l'image de manière réaliste et cohérente par rapport à la scène observée.
Initialement, le problème du SLAM visuel en ligne (calcul des poses et du modèle en temps réel) a été résolu à l'aide d'un filtre de Kalman étendu (EKF) comme démontré par [1]. Malgré la simplicité et l'efficacité offertes par les approches basées filtrage, des méthodes plus robustes basées optimisation s'appuyant sur l'ajustement de faisceaux furent plus tard implémentées en temps réel, grâce à l'augmentation des capacités de calcul et de mémoire.
Afin de pouvoir minimiser une erreur de reprojection dans un temps satisfaisant, les premières méthodes temps-réel basées optimisation utilisaient des fenêtres glissantes d'images [11] ou un sous ensemble d'images dénommées images clés[7]. [16] établit la supériorité de ces méthodes dans la plupart des configurations ; elles sont par ailleurs popularisées par de nombreuses implémentations Open Source : ORB-SLAM [12], LSD-SLAM [3], SVO [4], etc.

1.2 Modèles de caméras

Les algorithmes de SLAM visuel nécessitent l'utilisation d'un modèle géométrique adapté à la caméra réelle afin de pouvoir projeter un point de la scène dans l'image. Le modèle fréquemment utilisé est le modèle sténopé, qui suppose la caméra statique durant le temps d'intégration sur les photosites du capteur d'image. Cette hypothèse est réaliste pour des caméras à obturateur global (OG). Dans notre contexte de réalité augmentée sur smartphone ou tablette, la large majorité des caméras disponibles sont des caméras à obturateur déroulant (OD). Ces caméras, à la différence des caméras OG, exposent et transfèrent chaque ligne de l'image à des instants différents.Cet effet est également présent sur les appareils photographiques équipés d'un obturateur mécanique utilisant deux rideaux, l'un découvrant

le capteur, et l'autre le recouvrant, afin que chaque ligne de l'image soit exposée durant une même durée. L'ouverture entre les deux rideaux permet de contrôler la durée d'exposition de chaque ligne, rendant possible l'obtention de temps de pose très court pour une faible ouverture. Dans le cas d'une caméra statique et d'une scène fixe, il n'y a pas de différence entre des images acquises par des caméras OG et OD. En revanche, un mouvement relatif entre une caméra OD et la scène durant l'exposition de l'image entraîne des distorsions étant donné que chaque ligne est exposée depuis un point de vue différent. Ceci implique que chaque ligne de l'image doit être traitée comme une image 1D avec sa propre pose caméra.

1.3 SLAM pour caméra OD

Si elles sont exploitées sur des séquences d'images acquises par une caméra OD, les méthodes SLAM utilisant un modèle OG risquent de produire des modèles de l'environnement et des trajectoires incohérentes. [8] furent parmi les premiers à tenir compte de l'OD pour adapter PTAM sur smartphone ; dans leur approche, la vitesse angulaire de la caméra (supposée constante dans une image) est estimée afin de rectifier les mesures, les rendant utilisables avec un simple modèle de caméra OG.

[5], s'inspirant des travaux de [10], proposèrent un modèle de caméra basé sur l'interpolation linéaire et l'utilisèrent avec succès dans un ajustement de faisceaux global. Le mouvement de la caméra est interpolé linéairement en découplant les interpolations de la rotation (SLERP) et de la translation (linéaire).

[13] suggérèrent l'utilisation des B-Splines afin d'avoir une représentation en temps continu de la trajectoire caméra. A l'instar de [5], ils choisirent d'interpoler indépendamment la rotation et la translation.

D'après [15], ces formulations sont sujettes à des problèmes pouvant être résolus en utilisant des B-Splines cumulatives afin d'interpoler directement sur la variété. Dans [17], nous avions montré que la distribution temporelle uniforme (DTU) des poses de contrôle (abrégées PC dans la suite) des B-Splines, utilisée par [15], entraîne un lissage de la trajectoire ou une redondance inutile d'informations. Nous proposions d'utiliser une distribution temporelle non uniforme (DTNU) et suggérions des méthodes dynamiques pour la génération des PC. Les travaux présentés ici sont une extension de [17], décrivant le modèle de B-Spline non uniforme adapté à la problématique et ajoutant l'horodatage des PC au processus d'optimisation. Un rapide résumé des notions et notations est donné dans la section suivante.

2 Notations et Contexte Scientifique

Dans cet article, les 6 degrés de liberté (6DOF) des paramètres extrinsèques d'une caméra seront exprimés par une matrice de transformation 4×4 représentant la position et l'orientation de la caméra dans le repère monde (passant du repère caméra vers le repère monde). Cette matrice $T_w \in \mathbb{SE}3$ est paramétrée par une translation a et une matrice de rotation R :

$$\mathbf{T}_w = \begin{pmatrix} \mathbf{R} & \mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix}, \mathbf{T}_w \in \mathbb{SE}3, \mathbf{R} \in \mathbb{SO}3, \mathbf{a} \in \mathbb{R}^3 \quad (1)$$

Une telle transformation (pose) est représentée dans l'algèbre de Lie $\mathfrak{se}(3)$ sous la forme d'un vecteur à 6 dimensions $\xi = [w, a]^T \in \mathfrak{se}(3)$ où $w = [\omega_0, \omega_1, \omega_2]^T$ et $a = [a_0, a_1, a_2]^T$ sont respectivement les composantes en rotation et translation. Elle est également décrite par une matrice Ω de taille 4×4 obtenue en appliquant l'opérateur $[\cdot]_\wedge$ sur ξ sachant que $[w]_\times$ représente la matrice antisymétrique 3×3 de w :

$$\Omega = [\xi]_\wedge = \begin{pmatrix} [w]_\times & \mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix} \quad (2)$$

La fonction logarithmique d'une matrice permet de faire correspondre une matrice de pose depuis $\mathbb{SE}3$ vers son espace tangent $\mathfrak{se}(3)$ localement défini Euclidien, permettant la composition des transformations par simple addition. L'application exponentielle, étant l'opération inverse de l'application logarithmique, permet de reprojeter un vecteur de dimension 6 depuis l'espace tangent $\mathfrak{se}(3)$ vers $\mathbb{SE}(3)$. Ces deux applications peuvent être résumées de la manière suivante pour des rotations d'amplitude inférieure à π :

$$\begin{aligned} T_w &= exp(\Omega) \\ \Omega &= log(T_w) \end{aligned} \quad (3)$$

2.1 B-Splines Cumulatives

Comme énoncé précédemment, une caméra OD acquiert les lignes d'une image à des dates différentes. Une estimée de la pose caméra pour chaque ligne est donc nécessaire afin de projeter des points 3D de la scène dans l'image. Cette estimée peut être obtenue par le biais d'une modélisation de la trajectoire en temps continu. La pose caméra associée à une ligne est alors interpolée à la date d'exposition de la ligne à partir de PC horodatées à des instants t_i, contrôlant localement la trajectoire. Ces PC sont définies dans le repère monde et notées $T_{w,i}$. Différentes méthodes d'interpolation peuvent être mises en œuvre (linéaire, B-Spline, Bézier) suivant le cas d'utilisation considéré.

La forme standard des B-Splines, qui exprime l'interpolation par une somme pondérée de PC n'est pas adaptable à l'espace des transformations rigides $\mathbb{SE}(3)$ où les éléments se composent par multiplication matricielle. [15] proposent d'utiliser la forme cumulative des B-Splines pour leur adaptabilité à l'interpolation sur la variété $\mathbb{SE}(3)$. Ils choisissent un modèle cubique des B-Splines, nécessitant $k = 4$ PC pour l'interpolation afin d'obtenir une continuité C^2 de la trajectoire, ce qui va permettre l'interpolation des vitesses et accélérations.

Dans le cas des B-Splines cumulatives cubiques, les fonctions de base cumulatives $\tilde{B}_k(t)$ pondérant les variations de

poses sont exprimées par la k^{ieme} composante du vecteur $\tilde{B}(t)$ décrit par :

$$\tilde{B}(t) = \frac{1}{6}\begin{pmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{pmatrix} \quad (4)$$

où $u(t) = \frac{t-t_{i+1}}{\Delta t}$ le temps uniforme entre les deux PC $T_{w,i+1}, T_{w,i+2}$ qui encadrent le temps d'interpolation t. Δt définit l'intervalle temporel entre les PC, intervalle qui est considéré constant dans le cas des B-Splines uniformes.
L'interpolation sur la variété $\mathbb{SE}3$ est définie comme une composition de variations de PC $\Omega_{j-1,j} = log(T_{w,j-1}^{-1} T_{w,j})$ pondérées par les fonctions de base $\tilde{B}(t)$ appliquée à une pose de référence $T_{w,i}$. Ainsi, dans le cas des B-Splines cumulatives cubiques $k = 4$, l'interpolation de la pose $T_w(t)$ entre deux PC $T_{w,i+1}$ et $T_{w,i+2}$ nécessite les 4 PC avoisinantes $T_{w,i..i+3}$. La fonction d'interpolation est alors exprimée par :

$$\mathbf{T}_w(t) = \mathbf{T}_{w,i} \prod_{j=i+1}^{i+k-1} \exp(\tilde{B}_j(t)\mathbf{\Omega}_{j-1,j}) \quad (5)$$

3 Distribution temporelle non-uniforme des PC

Cet article étend les travaux décrits dans [17], où nous proposions l'utilisation d'une distribution temporelle non uniforme (DTNU) des PC. La formulation non-uniforme octroie la possibilité de mieux répartir les PC dans le temps afin de s'adapter à la dynamique de la trajectoire pour en permettre une modélisation plus efficace.
La distribution temporelle non uniforme suppose la relaxation de la contrainte sur la constance de l'intervalle temporel entre les PC Δ_t. A la place est défini un intervalle temporel $\Delta t_{i-1,i}$ relatif aux horodatages des deux PC $T_{w,i-1}$ $T_{w,i}$.
Par abus de langage, nous appellerons B-Spline non-uniforme les B-Splines dont les PC sont espacés de manière non-uniforme dans le temps. Pour ce cas particulier, les fonctions de base sont définies par une relation de récurrence légèrement différente de celle de De Boor [2], voir [6] :

$$B_{i,k}(t) = \frac{t-t_i}{t_{i+k-1}-t_i} B_{i,k-1}(t) + \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} B_{i+1,k-1}(t) \quad (6)$$

Avec la condition d'arrêt définie par

$$B_{i,1}(t) = \begin{cases} 1 & \text{si} \quad t_i < t < t_{i+1} \\ 0 & \text{sinon} \end{cases} \quad (7)$$

Cette version non-uniforme des B-Splines nécessite pour l'interpolation à un temps t (tel que $t_{i+2} < t < t_{i+3}$) les 4 PC ($T_{w,i}$ à $T_{w,i+3}$) ainsi que les horodatages de 6 PC ($T_{w,i}$ à $T_{w,i+5}$).

La formule de récurrence permet de calculer successivement les fonctions de base pour différents degrés. En développant les calculs à la manière de [18] ou en utilisant les matrices de Toeplitz[14], il est possible d'obtenir une forme matricielle similaire à celle utilisée par [15] pour les B-Splines cubiques. A l'inverse de la formulation pour les B-Splines uniformes, la matrice des coefficients utilisée dans le calcul des fonctions de base devient dépendante de l'horodatage des PC.
L'expression de temps uniforme entre les deux PC du segment d'interpolation considéré est donnée par :

$$u(t) = \frac{t-t_{i+2}}{t_{i+3}-t_{i+2}} \quad (8)$$

Les fonctions de base pour l'interpolation non uniforme s'exprime alors par :

$$B(t) = M\begin{pmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{pmatrix} \quad (9)$$

Les détails des coefficients de la matrice M sont donnés dans [18], il est cependant important de noter ici que M dépend des horodatages des 6 PC $[t_i..t_{i+5}]$.
C étant la matrice cumulative des coefficients obtenue à partir de M, la version cumulative des fonctions de base est définie par :

$$C_{i,j} = \sum_{l=i}^{k} M_{l,j} \quad , \quad \tilde{B}(t) = C\begin{pmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{pmatrix} \quad (10)$$

La figure 1 représente un exemple des 4 fonctions de base non uniformes non-cumulatives et cumulatives en fonction du temps. A l'inverse de la formulation uniforme, les fonctions de base diffèrent pour chaque tronçon d'interpolation du fait de l'intervalle temporel non-uniforme des PC.
La matrice cumulative des coefficients C ne dépendant pas de t, il est possible de retrouver les dérivées des fonctions de base en fonction de t à l'aide d'une formulation similaire à celle des B-Splines uniformes :

$$\dot{\tilde{B}}(t) = \frac{1}{\Delta t_{i+2,i+3}} C\begin{bmatrix} 0 \\ 1 \\ 2u \\ 3u^2 \end{bmatrix}, \ddot{\tilde{B}}(t) = \frac{1}{\Delta t_{i+2,i+3}^2} C\begin{bmatrix} 0 \\ 0 \\ 2 \\ 6u \end{bmatrix} \quad (11)$$

Les dérivées première et seconde de la trajectoire s'expriment alors de la manière suivante :

$$\dot{\mathbf{T}}_w(t) = A_0(\dot{A}_1 A_2 A_3 + A_1 \dot{A}_2 A_3 + A_1 A_2 \dot{A}_3) \quad (12)$$

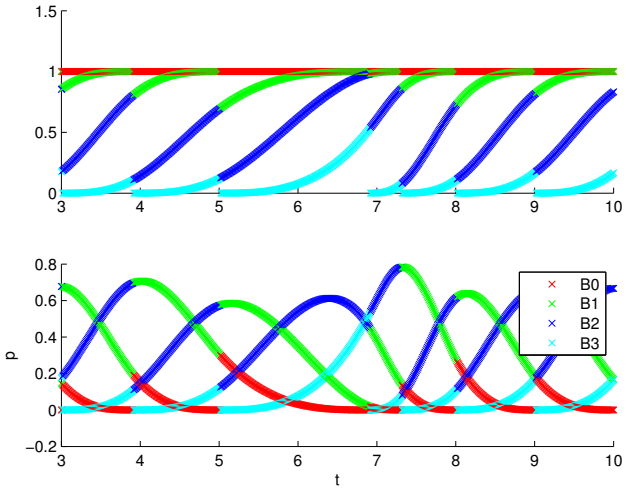

FIGURE 1 – Cette figure décrit l'évolution de la valeur des $k = 4$ fonctions de base non-uniformes cumulatives $\tilde{B}_{0..k}(t)$(haut) et non cumulatives $B_{0..k}(t)$(bas) en fonction du temps pour une DTNU des PC. A chaque fonction de base correspond une couleur (rouge, vert, bleu, cyan), illustrant l'influence d'une PC(bas) ou d'une variation de PC(haut) sur la trajectoire interpolée au cours du temps.

$$\ddot{\mathbf{T}}_w(t) = A_0 \begin{pmatrix} \ddot{A}_1 A_2 A_3 + A_1 A_2 \ddot{A}_3 + A_1 A_2 \ddot{A}_3) + \\ 2 * (\dot{A}_1 \dot{A}_2 A_3 + \dot{A}_1 A_2 \dot{A}_3 + A_1 \dot{A}_2 \dot{A}_3) \end{pmatrix} \quad (13)$$

$$A_j = \exp(\tilde{B}_j(t)\mathbf{\Omega}_{j-1,j}), \dot{A}_j = A_j \dot{\tilde{B}}_j(t)\mathbf{\Omega}_{j-1,j} \quad (14)$$

$$\ddot{A}_j = \dot{A}_j \dot{\tilde{B}}_j(t)\mathbf{\Omega}_{j-1,j} + A_j \ddot{\tilde{B}}_j(t)\mathbf{\Omega}_{j-1,j} \quad (15)$$

Dans la section suivante sont adaptées différentes méthodes d'optimisation des PC utilisant ce modèle de B-Spline non-uniforme. Nous y détaillons également un processus d'optimisation spatio-temporelle rendu possible par la relaxation de la contrainte uniforme.

4 PnP et Ajustement de faisceaux

4.1 Modèle de camera OD

La définition en temps continu de la trajectoire de la caméra permet le calcul des paramètres extrinsèques de la caméra OD pour chaque ligne de l'image. Afin de pouvoir projeter un point 3D X_w de la scène vers l'image en fonction de t, la pose caméra $T_w(t)$ doit être interpolée.
Soit K la matrice des paramètres intrinsèques et $\pi(.)$ l'opération de déshomogénéisation passant de $\mathbb{P}2$ vers $\mathbb{R}2$. La trajectoire temporelle de la projection de X_w sur l'image de la caméra OD est alors définie par :

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \pi([K|0](T_w(t))^{-1} X_w) \quad (16)$$

De plus, une ligne de la caméra OD n'est exposée qu'à une certaine date. La ligne y exposée en fonction de t peut être exprimée comme fonction des dates de départ s et de fin e de l'exposition de l'image ainsi que de la hauteur de l'image en pixel h :

$$y(t) = h\frac{(t-s)}{(e-s)} \quad (17)$$

La valeur de t pour laquelle les fonctions de projection (eq. 16) et d'exposition (eq. 17) coïncident peut être estimée en intersectant les courbes correspondantes. Nous référons les lecteurs à [15] pour les détails concernant cette optimisation.

4.2 PnP avec caméra OD

L'algorithme du PnP (Perspective-N-Points) permet d'estimer les paramètres extrinsèques d'une ou plusieurs caméras à partir de n correspondances entre des points 3D connus de la scène P_k et leurs observations 2D $p_{i,k}$ dans les images.
A partir de ces correspondances et d'une estimée initiale de la pose caméra T_w (supposée pour l'instant constante pour l'image i) peut être calculée une erreur de reprojection fonction du point k :

$$Err(i,k) = p_{i,k} - \pi([K|0]T_w^{-1}P_k) \quad (18)$$

En adaptant la formulation du PnP aux caméras OD, on obtient une formulation de l'erreur dépendante de la date d'exposition de la ligne sur laquelle est projeté le point :

$$Err(i,k) = p_{i,k} - \pi([K|0]T_w(t)^{-1}P_k) \quad (19)$$

Dans le cas du PnP OD, l'ensemble des paramètres θ à optimiser est un ensemble de PC et non plus directement des poses caméras. L'ensemble des paramètres raffinés $\hat{\theta}$ est obtenu en minimisant la somme des carrés des erreurs de reprojection :

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_i \sum_k Err(i,k)^2 \quad (20)$$

4.3 Ajustement de faisceaux

L'ajustement de faisceaux est une méthode d'optimisation qui, à l'instar du PnP, permet de raffiner un ensemble de paramètres en minimisant une fonction de coût. A la différence du PnP, l'ajustement de faisceaux intègre également les points 3D de la carte dans les paramètres à optimiser. Les paramètres de l'ajustement de faisceaux sont donc définis par $\theta_b = [\theta, X_0..X_M]$, et sont raffinés par minimisation de l'erreur de reprojection :

$$\hat{\theta_b} = \underset{\theta_b}{\operatorname{argmin}} \sum_i \sum_k Err(i,k)^2 \quad (21)$$

La minimisation est effectuée par l'algorithme de Levenberg-Marquardt. Les paramètres de θ_b peuvent être initialisés à partir d'un SLAM/PnP utilisant un modèle de caméra OG. Les PC θ sont alors initialisées comme étant sur la trajectoire interpolée.

4.4 Optimisation temporelle

L'utilisation d'un modèle de B-Spline non-uniforme permet d'optimiser la composante temporelle des PC. Les horodatages des PC peuvent alors être ajoutés aux vecteurs des paramètres à optimiser, en conservant la même fonction de coût que précédemment. La datation des PC intervient uniquement dans le calcul de la matrice des coefficients M (eq. 9).

Cependant, l'ajout de ces paramètres doit être réalisé avec précautions, la modification de l'horodatage des PC par une mise à jour d'une itération de l'optimisation devant respecter un certain nombre de contraintes. Les intervalles temporels doivent rester positifs, et l'intervalle temporel global d'interpolation doit comprendre l'ensemble des temps d'exposition des lignes.

De plus, ajouter les horodatages des PC à l'optimisation complexifie la variété issue de la fonction de coût et perturbe sa convexité. L'optimisation simultanée des PC et de leurs horodatages peut donc converger vers une solution erronée locale en partant d'une mauvaise estimée initiale, particulièrement à cause de l'apparition de minima locaux.

Pour répondre à ce problème, nous proposons d'alterner des phases d'optimisation spatiale et temporelle des PC. Ainsi la première optimisation, libérée des minima locaux induits par l'ajout des horodatages, converge vers une solution proche. Il est ensuite possible d'appliquer une optimisation uniquement temporelle ou spatio-temporelle de manière à affiner la trajectoire.

5 Tests sur données synthétiques

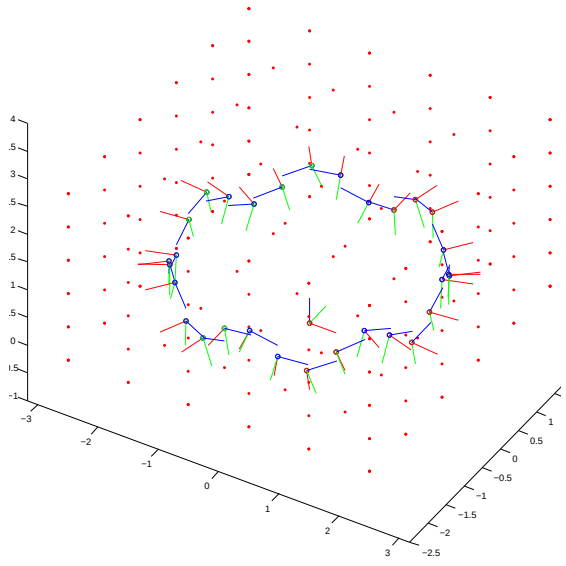

FIGURE 2 – Exemple de PC qui suivent une trajectoire circulaire où la composante en translation z suit une sinusoïde et l'axe z de la caméra est aligné avec la tangente à la courbe. Le nuage de points est affiché en rouge.

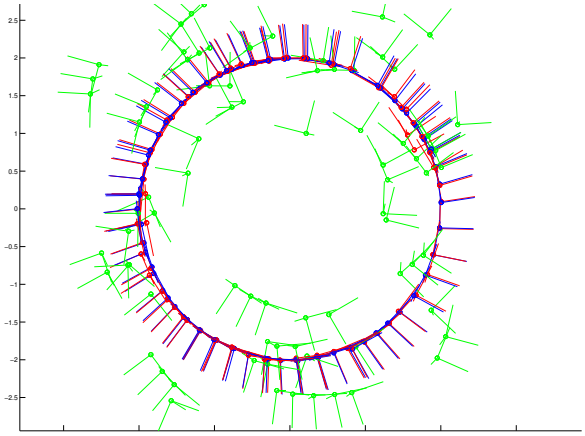

FIGURE 3 – Les poses de caméras interpolées à partir des PC initiales (en vert), des PC optimisées (en bleu) et de la vérité terrain (en rouge).

Nous choisissons ici de travailler sur des données simulées afin de se soustraire aux problématiques de traitement d'image et d'association de données. Les expérimentations se concentrent sur le comportement de notre modèle pour les problèmes du PnP et de l'ajustement de faisceaux.

Dans un premier temps, des PC sont générées suivant différents types de mouvements (linéaires, circulaires ou aléatoires). De plus, un nuage de points 3D représentant un modèle géométrique d'environnement simple est créé (voir fig. 2).

Les PC sont initialisées au niveau des poses interpolées (rappelons que la trajectoire ne passe généralement pas par les PC). Les PC et les points 3D sont ensuite bruités. Les intervalles temporels entre les PC sont générés aléatoirement afin d'obtenir une DTNU permettant de tester et valider notre modèle.

Nous choisissons arbitrairement la fréquence de la caméra ($fps = 3$), ainsi que les temps de départ et de fin auxquels la caméra acquière des images. Nous interpolons à l'aide de ces paramètres les poses de caméras et nous projetons les points dans les images avec un modèle similaire à celui décrit dans [17] et [15]. Ces données sont également bruitées, puis utilisées comme mesures dans un schéma d'optimisation. Les données sont générées sous MATLAB.

5.1 PnP

Dans le cas du PnP, nous utilisons des poses interpolées à partir des PC bruitées comme estimées initiales des PC. Une centaine d'images interpolées observant un nuage de points non bruités sont générées à partir de 60 PC bruitées. Le PnP est résolu par notre optimiseur développé en C++ et basé sur g2o [9].

Sur la figure 3 les poses de caméras initiales (en vert) convergent après optimisation (en bleu) vers la vérité terrain (en rouge). Cinq itérations d'optimisation suffisent à obtenir des poses de caméras fidèles à la vérité terrain malgré des poses initiales très bruitées, montrant le caractère

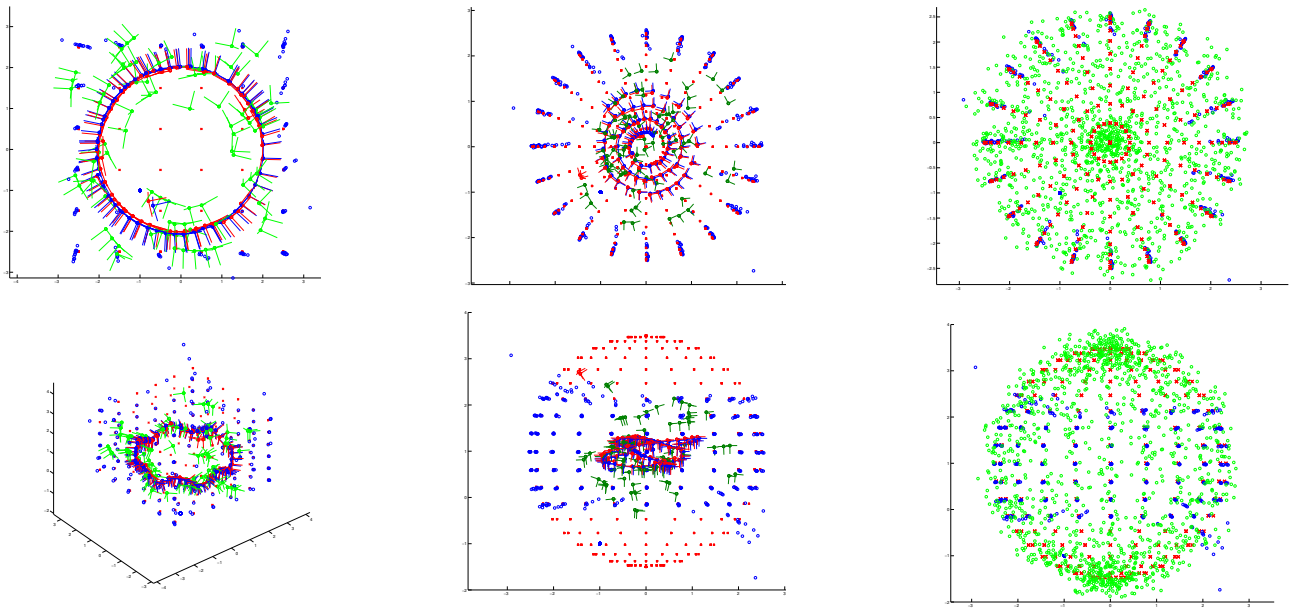

FIGURE 4 – Vues de dessus (en haut) et de côté (en bas) des résultats de l'optimisation non-uniforme. Les poses de caméras interpolées à partir des PC initiales (vert), des PC optimisées (bleu) et à partir de la vérité terrain (en rouge) sont affichées aux côtés des points optimisés de la carte. A droite le nuage de points initial (vert), optimisé (bleu) et la vérité terrain (rouge).

largement convexe de la fonction de coût.

5.2 Ajustement de faisceaux

Pour tester l'ajustement de faisceaux, nous utilisons les mêmes trajectoires simulées que pour le PnP, en intégrant cette fois les points bruités à l'initialisation qui seront raffinés par l'optimisation. L'ajustement de faisceaux est également résolu par notre optimiseur utilisant g2o [9]. Les résultats obtenus après optimisation sont exposés dans la figure 4 où l'on observe la convergence des points et des caméras optimisés (en bleu) vers la vérité terrain (en rouge). La colonne de droite montre uniquement les nuages de points bruités initiaux (en vert), optimisés (en bleu) et de la vérité terrain (en rouge).

6 Tests sur données réelles

Afin de motiver notre méthode ajoutant la composante temporelle des PC à l'optimisation, nous avons dans un premier temps cherché à adapter une trajectoire interpolée sur la trajectoire réelle d'une caméra tenue à la main. Les données de cette trajectoire servant de vérité terrain sont obtenues par un système de capture de mouvements (MOCAP VICON) fournissant une trajectoire échantillonnée à une fréquence de 200hz. Diverses séquences ont été réalisées en suivant le mouvement d'une caméra tenue à la main, avec des dynamiques différentes en rotation et translation.

6.1 Optimisation spatio-temporelle des PC

Nous tentons de faire correspondre aux trajectoires réelles une trajectoire interpolée avec un modèle uniforme et un modèle non-uniforme de B-Spline et suivant différentes démarches d'optimisation des PC. Les PC sont initialisées sur la trajectoire MOCAP à des temps uniformément répartis. L'erreur $Err_T \in \mathbb{R}6$ entre les poses interpolées $T_w(t)$ et la vérité terrain $T_{GT}(t)$ est minimisée pour l'ensemble des horodatages t_i de la trajectoire MOCAP :

$$Err_T = \sum_i log(T_{GT}(t_i)^{-1} T_w(t_i)) \qquad (22)$$

L'optimisation est réalisée par l'algorithme Levenberg-Marquardt avec un calcul des jacobiennes aux différences finies. Cette optimisation permet de raffiner spatialement les PC. Comme décrit dans [17], une PC est ensuite ajoutée entre les deux PC où l'erreur après optimisation est la plus élevée.

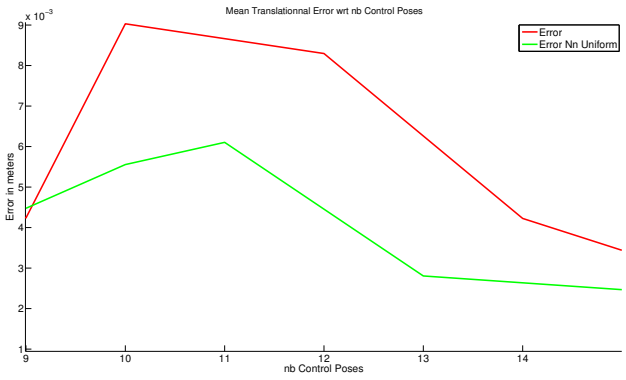

FIGURE 5 – L'erreur moyenne sur la translation en fonction du nombre de PC avec une distribution uniforme (en rouge) et non-uniforme (en vert).

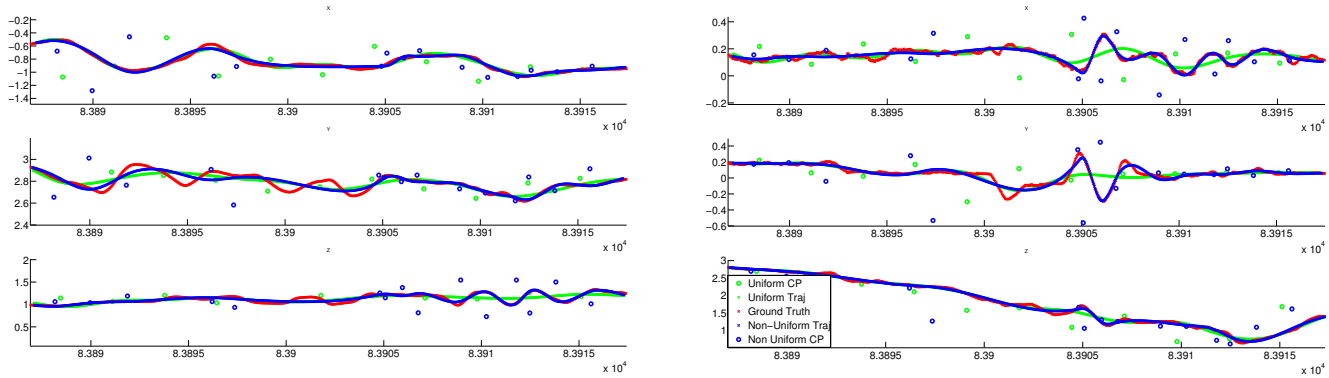

FIGURE 6 – Les composantes (X, Y, et Z) en translation (gauche) et en rotation (droite) des trajectoires interpolées après optimisation spatiale uniforme (en vert) et après 3 cycles d'optimisation spatio-temporelle (en bleu) en fonction du temps. La vérité terrain est illustrée en rouge.

La figure 5 illustre l'erreur moyenne en translation obtenue après optimisation spatiale en fonction du nombre de PC, pour une distribution uniforme (en vert) et non uniforme (en rouge). Comme attendu, à nombre de PC égal, il est possible de modéliser plus fidèlement une trajectoire avec une distribution non uniforme des PC qu'avec une distribution uniforme en ajoutant des PC où l'information manque.

Cette méthode ajoute des PC à un temps arbitraire entre les horodatages des deux PC environnantes, généralement $\frac{t_{i+2}+t_{i+3}}{2}$. La distribution temporelle est alors affinée en optimisant les horodatages des PC.

Comme énoncé précédemment, l'ajout des horodatages des PC à l'optimisation perturbe la convexité de la fonction de coût. L'optimisation offre dans la majorité des cas un gain substantiel qui se traduit par une diminution significative de l'erreur. Cependant de meilleurs résultats sont obtenus en suivant un schéma d'optimisation alterné. Une première optimisation spatiale est réalisée jusqu'à convergence. On répète ensuite n fois les trois opérations suivantes :

— Optimisation temporelle
— Ajout d'une PC où l'erreur est la plus élevée
— Optimisation spatiale

La figure 6 illustre les composantes de la trajectoire interpolée obtenues après optimisation uniforme spatiale (en vert) et après $n = 3$ cycles d'optimisation spatio-temporelle (en bleu), ainsi que la vérité terrain (en rouge). La répartition temporelle et spatiale des PC s'adapte automatiquement à la dynamique de la trajectoire, ce qui est observable notamment sur les composantes en rotation (à droite fig. 6) et la composante z en translation (bas-gauche fig. 6) des trajectoires, où l'interpolation uniforme échoue à modéliser les oscillations rapides de la trajectoire.

Ainsi, pour une optimisation globale, il est approprié de privilégier dans un premier temps une optimisation spatiale des PC, pour ensuite réaliser des cycles d'optimisation spatio-temporelle afin de raffiner la trajectoire interpolée.

Cette expérience démontre qu'il est possible d'adapter la disposition spatio-temporelle des PC à la dynamique d'une trajectoire réelle par le biais d'un processus d'optimisation. Le modèle d'interpolation cubique permet une modélisation fidèle de la trajectoire. Des modèles quadratique ou linéaire peuvent être suffisants selon le type de trajectoire, mais la possibilité d'interpoler de manière continue les accélérations rend le modèle cubique préférable pour les applications avec données inertielles. La fonction de coût utilisée ici est relative à une vérité terrain qui n'est pas accessible dans le cas d'un processus de SLAM. Des tests approfondis doivent être menés en intégrant l'optimisation temporelle dans un ajustement de faisceaux. Passer de la minimisation d'une erreur sur $\mathfrak{se}(3)$ à une erreur sur $\mathbb{R}2$ peut éventuellement poser problème pour des configurations dégénérées qu'il est nécessaire d'évaluer.

7 Conclusion et futurs travaux

Dans cet article nous avons proposé un modèle de B-Spline cumulative non uniforme permettant de conserver la continuité C2 de la trajectoire et d'assurer un parcours correct de la trajectoire dans le temps. Ce modèle a été testé pour ajuster des trajectoires interpolées sur des trajectoires réelles avec différentes démarches d'optimisation. Nous avons montré l'apport de l'optimisation des horodatages des PC pour l'interpolation de trajectoires réelles. Nous avons également proposé un ajustement de faisceaux utilisant le modèle de B-Spline non uniforme et exposé son comportement sur des données simulées.

Le modèle de trajectoire en temps continu utilisant une DTNU des PC pour une caméra ou un robot permet de réduire le coût mémoire et calculatoire, en concentrant l'information uniquement où elle est nécessaire. Pour des trajectoires très linéaires, par exemple celle d'automobiles, un grand nombre de poses de caméra associées à des images clés peuvent être paramétrées par un faible nombre de PC. Pour des trajectoires à dynamiques variables, il est possible de modéliser fidèlement les mouvements rapides par une augmentation locale de la distribution des PC.

Pour modéliser ce type de trajectoire avec une DTU des PC, l'échantillonnage des PC doit s'adapter à la plus forte dynamique de la trajectoire, entraînant une redondance de l'information pour les portions à faible dynamique. Le contrôle local de l'échantillonnage permis par la DTNU des PC autorise une modélisation fine des mouvements rapides sans accroître dramatiquement le coût calculatoire.

Ces travaux sont encore prospectifs, et des tests avec un processus complet de SLAM intégrant notre modèle non-uniforme devront être conduits sur des données réelles. Nous souhaitons également enrichir la paramétrisation des amers de la scène (segments, droites, plans) et adapter leurs modèles d'observation aux caméras OD. S'appuyant sur la continuité C2 offerte par notre modèle, il est envisagé d'optimiser les données inertielles [15], permettant l'auto-étalonnage de la caméra ainsi que l'obtention de l'échelle absolue de la scène.

8 Remerciements

Cette recherche est financée par CIFRE ANRT 2014/1503 et le projet régional FEDER-FSE Midi-Pyrenées et Garonne REALISM n.15056690.

Références

[1] DAVISON, A. J. Real-time simultaneous localisation and mapping with a single camera. In *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France* (2003), pp. 1403–1410.

[2] DE BOOR, C. On calculating with b-splines. *Journal of Approximation Theory 6*, 1 (1972), 50 – 62.

[3] ENGEL, J., SCHÖPS, T., AND CREMERS, D. LSD-SLAM : Large-scale direct monocular SLAM. In *ECCV* (September 2014).

[4] FORSTER, C., CARLONE, L., DELLAERT, F., AND SCARAMUZZA, D. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Proceedings of Robotics : Science and Systems* (Rome, Italy, July 2015).

[5] HEDBORG, J., FORSSÉN, P. E., FELSBERG, M., AND RINGABY, E. Rolling shutter bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (June 2012), pp. 1434–1441.

[6] KIM, M.-J., KIM, M.-S., AND SHIN, S. Y. A general construction scheme for unit quaternion curves with simple high order derivatives. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95* (1995), 369–376.

[7] KLEIN, G., AND MURRAY, D. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)* (Nara, Japan, November 2007).

[8] KLEIN, G., AND MURRAY, D. Parallel tracking and mapping on a camera phone. In *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)* (Orlando, October 2009).

[9] KUEMMERLE, R., GRISETTI, G., STRASDAT, H., KONOLIGE, K., AND BURGARD, W. g2o : A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai, China, May 2011), pp. 3607–3613.

[10] MEINGAST, M., GEYER, C., AND SASTRY, S. Geometric models of rolling-shutter cameras. *CoRR abs/cs/0503076* (2005).

[11] MOURAGNON, E., LHUILLIER, M., DHOME, M., DEKEYSER, F., AND SAYD, P. Real time localization and 3d reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006), vol. 1, pp. 363–370.

[12] MUR-ARTAL, R., MONTIEL, J. M. M., AND TARDÓS, J. D. Orb-slam : A versatile and accurate monocular slam system. *IEEE Transactions on Robotics 31*, 5 (Oct 2015), 1147–1163.

[13] OTH, L., FURGALE, P., KNEIP, L., AND SIEGWART, R. Rolling shutter camera calibration. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on* (June 2013), pp. 1360–1367.

[14] QIN, K. General matrix representations for B-splines. *The Visual Computer 16*, 3 (2000), 177–186.

[15] STEVEN LOVEGROVE, ALONSO PATRON-PEREZ, G. S. Spline fusion : A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *Proceedings of the British Machine Vision Conference* (2013), BMVA Press.

[16] STRASDAT, H., MONTIEL, J., AND DAVISON, A. J. Real-time monocular slam : Why filter ? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (2010), IEEE, pp. 2657–2664.

[17] VANDEPORTAELE, B., GOHARD, P.-A., DEVY, M., AND COUDRIN, B. Pose interpolation for rolling shutter cameras using non uniformly time-sampled b-splines. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 6 : VISAPP, (VISIGRAPP 2017)* (2017), INSTICC, SciTePress, pp. 286–293.

[18] YANG, H., YUE, W., HE, Y., HUANG, H., AND XIA, H. The Deduction of Coefficient Matrix for Cubic Non-Uniform B-Spline Curves. *2009 First International Workshop on Education Technology and Computer Science*, 3 (2009), 607–609.