Vers une solution à la dérive du *SLAM* visuel en environnement urbain par une connaissance éparse de l'environnement

Maxime BOUCHER Fakhr-Eddine ABABSA Malik MALLEM

Laboratoire IBISC

Université d'Evry Val d'Essonne

{maxime.boucher, fakhreddine.ababsa, malik.mallem}@ibisc.fr

Résumé

Le SLAM (Simultaneous Localization and Mapping) consiste à assurer une tâche de localisation simultanément à une tâche de construction d'une carte de l'environnement à partir de données télémétriques et/ou d'images fournies par des caméras mobiles. Si le SLAM apparaît aujourd'hui mature, plusieurs problèmes scientifiques demeurent ouverts, et tout particulièrement celui de la dérive. En effet, de par son caractère incrémental, le SLAM est sujet à l'accumulation d'erreurs (aussi bien en localisation qu'en reconstruction) qui proviennent essentiellement de mesures imprécises et qui se propagent au cours du temps. La solution du problème de la dérive ne peut provenir uniquement d'une amélioration/complexification des méthodes incrémentales actuelles, ou d'un apport de connaissance relative. De la connaissance absolue s'avère nécessaire. Dans ce papier, nous proposons de modifier la méthode classique de l'ajustement de faisceaux en incorporant des poses de caméras, et/ou de points 3D, connu(e)s a priori dans le processus d'estimation.

Mots clefs

SLAM, Ajustement de faisceaux.

1 Introduction

La localisation et cartographie simultannée, couramment nommée *SLAM*, est aujourd'hui encore un champ de recherche actif. Cette tâche consiste pour un système mobile à construire un modèle de l'environnement dans lequel il évolue et à s'y localiser. A la confluence de la robotique et de la vision par ordinateur, dans le cas du *SLAM* visuel, le champ a bénéficié d'apports de ces deux communautés. La réalité augmentée, qui se popularise auprès du grand public, est un domaine qui pourrait tirer un grand bénéfice de l'utilisation de systèmes *SLAM*. De plus, les platformes sur lesquelles ses applications sont appelées à se développer, des terminaux mobiles munis de capteurs visuels, inertiels et de localisation, sont particulièrement adaptées à la mise en oeuvre de ces systèmes. Dans la suite de cet article nous nous intéresserons au cas du *SLAM* visuel monoculaire, possiblement augmenté de capteurs supplémentaires (tels qu'un GPS ou une centrale inertielle) dans une optique d'utilisation future en réalité augmentée.

2 Etat de l'art

Notamment grâce à cette confluence de communautés, de nombreuses approches ont été proposées, et deux principales se sont dégagées. La première est celle basée sur le filtrage statistique, filtre de Kalman, ou bayésien, filtrage particulaire. Un résumé en est donné dans [1]. La seconde est basée sur une approche *Structure From Motion (SFM)* adaptée au temps réel. Cette approche héritée de la vision ne se soucie pas dans sa version originale de contraintes de temps. Dans cette optique, des paramètres initiaux de localisation et de cartographie sont estimés indépendamment puis raffinés simultannément en minimisant une fonction de coût. La minimisation de la fonction de coût est réalisée par ajustement de faisceaux [2].

Dans [3], Davison réalisa l'un des premiers *SLAM* en temps réel en optant pour une approche EKF (*Extended Kalman Filter*). Dans [4], Nister fit l'un des premiers pas dans l'adaptation du *SFM* au *SLAM*. Dans [5] l'idée est reprise, approfondie et fixe un nouveau standard dans la façon de réaliser le *SLAM*. Ce standard est confirmé dans [6] où Strasdat démontre qu'à partir d'un certain seuil de capacité calculatoire l'approche par ajustement de faisceaux est la plus précise des deux.

Néanmoins, quelque soit l'approche envisagée, un problème se pose irrémédiablement. Le *SLAM* est un processus fondamentalement incrémental. Des erreurs de mesures, de compromis et d'arrondis se produisent à chaque étape. Comme l'estimation des paramètres du modèle à chaque étape s'appuie sur des mesures de l'étape courante et l'historique des estimations, les erreurs s'accumulent et induisent tôt ou tard un état du système si erroné qu'il apparaît incohérent. C'est un problème très important puisque cela limite la taille des environnements d'utilisation potentiels. La communauté fait généralement référence à cette accumulation d'erreur en parlant de dérive. Elle est mise en évidence dans [3] et [7], pour les approches filtrée et *SFM* respectivement. Cette accumulation d'erreur ne se produit pas dans l'approche *Structure*

From Motion totale ou non temps réel car les paramètres du modèle sont estimés indépendamment puis optimisés conjointement. Dans [5] l'idée forte est de sélectionner des poses *clés*, et de ne réaliser l'ajustement de faisceaux que sur les paramètres des dernières poses clés tout en en gardant quelques unes un peu plus anciennes observatrices. Ainsi il peut satisfaire la contrainte de temps et garantir une cohérence entre les paramètres couramment estimés et ceux estimés lors des étapes précédentes. Cela limite également l'apparition de dérive. Cet ajustement de faisceaux est qualifié de local.
Dans [8], qui s'appuie sur [5], Lothe montre que l'un des phénomènes les plus importants de la dérive est la dérive d'échelle. Cela fait sens puisque, dans le cas monoculaire, la géométrie des vues multiples [9] est définie au facteur d'échelle près. Alors, une solution plusieurs fois investiguée est d'utiliser des capteurs additionnels. Pour l'approche filtrée la fusion de données est assez naturelle, un exemple de réalisation en est de [10]. La fusion de données par ajustement de faisceaux l'est nettement moins. Dans [11], pour une application routière, Eudes se sert de données odométriques pour corriger l'échelle des paramètres estimés avant de les raffiner par ajustement de faisceaux local. Ainsi des capteurs sont utilisés pour pallier la faiblesse intrinsèque de la vision monoculaire qu'est l'estimation du facteur d'échelle, mais on ne peut pas à proprement parler de fusion de données. Il n'y aucune garantie que ces informations soient prises en compte ou encore qu'elles ne soient pas modifiées. Dans [12], Lothe utilise un modèle approximatif de l'environnement pour contraindre la carte de son système au travers d'un ajustement de faisceaux modifié. C'est une première approche de fusion quoique taillée spécifiquement pour cette tâche et contrainte à la navigation routière. Cependant la fusion est réalisée lorsque le système tourne. Il peut donc se perdre lors de longues séquences en lignes droite, et pour limiter cet effet l'auteur estime un facteur d'échelle par identification du plan au sol pour lutter contre la dérive d'échelle. Cette approche est alors peu robuste aux mauvaises associations et peut se perdre.
En s'inspirant de [13], dans [14, 15] Michot réalise une fusion de données par ajustement de faisceaux en incorporant les données supplémentaires (gyroscopiques) dans le vecteur d'erreurs minimisé, l'ajustement de faisceaux devient alors multi-objectifs. Il constate que le choix de la pondération à affecter aux nouvelles données, lorsque leur variance est inconnue ou varie avec le temps, est non trivial et coûteux. Ce coût croît exponentiellement avec le nombre de poids à estimer. Finalement, dans [16] Lhuillier introduit deux nouveaux types d'ajustement faisceaux destinés à la fusion de données. Ils consistent en un ajustement de faisceaux local classique suivi, en quelque sorte, d'une étape de post-processing pour incorporer les données supplémentaires. Il montre que ces nouveaux ajustements de faisceaux réalisent une meilleure fusion de données que la précédente approche dans une application avec des données GPS. Les approches [15] et [16] sont génériques et peuvent être utilisées avec une grande variété de capteurs. Cependant, les processus restant intrinsèquement incrémentaux, l'accumulation d'erreur est toujours possible. Leurs limitations proviennent essentiellement des capteurs utilisés. Les informations gyroscopiques sont notamment sujettes à la dérive de ses axes et étant relatives ne peuvent enrayer la dérive. Les informations GPS sont quant à elles absolues et à ce sujet susceptibles de pouvoir enrayer la dérive. Mais leur qualité peut fluctuer et est généralement dépendante de l'environnement, ainsi les environnements urbains complexes ou intérieurs en tirent forcément moins parti que les environnements ouverts.

Parallèlement, les environnements urbains sont de plus en plus modélisés et parfois avec une grande précision. On peut notamment citer [17], dans lequel Agarwal *et al* reconstruisent des monuments et quartiers historiques de Rome à partir de photos collectées sur une banque de photos en ligne. La reconstruction est réalisée selon un schéma *Structure From Motion* avec ajustement de faisceaux total. Dans [18], Irschara utilise un modèle tel que [17] en fournit et par indexation des images utilisées pour effectuer la reconstruction et celle de sa caméra, il apparie les points du modèles avec ceux de son image et effectue une localisation très précise.

3 Solution proposée

Nous proposons d'investiguer la possibilité d'ajouter une connaissance éparse mais précise de l'environnement urbain pour aller à l'encontre du phénomène de dérive. Nous rappelons que le domaine d'application ciblé par nos travaux est la réalité augmentée. Dans un contexte de système mobile porté à la main par l'utilisateur les degrés de liberté du mouvement ne sont pas contraints et le système est susceptible d'être soumis à des accélérations parfois brutales. Ces conditions sont très propices à la formation de dérive.
Nous faisons l'hypothèse que dans le futur il sera de plus en plus probable pour un système *SLAM* en environnement urbain de croiser, ponctuellement, un monument fidèlement reconstruit [17] et disponible dans une base de donnée. Les informations que l'on peut tirer d'un tel modèle peuvent être utilisées pour relocaliser le système et fusionnées pour propager la connaissance précise nouvellement acquise dans l'historique des paramètres. Nous faisons donc également l'hypothèse que ces modèles seront géolocalisés, ne serait ce que grâce à la méthode de fusion de données GPS proposée par [16]. La propagation de la connaissance est importante en général puisque les poses passées sont susceptibles d'être utilisées lors d'une fermeture de boucle. Elle l'est encore plus dans le cadre d'une application en réalité augmentée : dans le cas où un objet virtuel serait déposé par l'utilisateur juste avant une reconnaissance de monument, la relocalisation est susceptible d'introduire un déplacement du virtuel par rapport au réel. La

propagation de connaissance en réduisant l'erreur passée permet de l'atténuer.
L'approche que nous proposons se situe entre celle de [5] et celle de [18]. Elle pourrait venir en complément de la fusion d'autres données dans le cas où celles ci ne seraient pas disponibles ou de mauvaise qualité. A la reconnaissance d'un modèle nous relocalisons le système grâce aux appariements 2D-3D selon la méthode de [19]. Ensuite, la nouvelle pose calculée et les points détectés sont incorporés à un ajustement de faisceaux comme paramètres fixes et les quelques dernières caméras et points de la carte comme paramètres variables. C'est, en pratique similaire à [5] lorsqu'il utilise des poses passées.
Soient P les points de la carte optimisés, P_a les points fixes (ancrés), C les poses de caméra optimisées, C_o les poses de caméra non optimisées (observatrices) regroupant indistinctement des poses passées et celle(s) calculée(s) lors d'une reconnaissance de scène et enfin x_j^i la mesure d'un point P_j dans la caméra C_i.
La fonction de coût minimisée est :

$$f(C,P) = \sum_{C_i \in \{C_o;C\}} \sum_{P_j \in \{P_a;P\}} d(h(C_i,P_j), x_j^i)^2$$

Où $d(.)$ est une mesure de distance et $h(C_i, P_j)$ la fonction de projection du point P_j dans la caméra C_i.
Les paramètres fixes ne peuvent pas varier mais contribuent à la formation du vecteur d'erreur. Alors, la minimisation de celui ci a pour conséquence de faire tendre les paramètres de l'historique récent vers de nouveaux plus cohérents avec l'état courant du système. L'ajustement de faisceaux sert alors de vecteur de propagation de connaissance.
Notre approche se base sur un *SLAM* classique, du type de [5], partiellement reprise dans [20]. On rappelle le principe de [5] :
Une caméra est dite clé lorsqu'elle est la dernière de la séquence à avoir un minimum de points d'intérêt appariés avec la précédente caméra clé. La première caméra clé est la première de la séquence. Nous utilisons pour détecteur et descripteur des points d'intérêt les *SIFT* de [21], et les appariements sont filtrés par *RANSAC* [22]. L'estimation de pose à partir de correspondances 2D-3D est réalisée selon l'algorithme de [19]. Pour réaliser les ajustements de faisceaux nous utilisons la bibliothèque *Simple Sparse Bundle Adjustment* [1].
Tout d'abord la carte nécessite d'être initialisée. La phase d'initialisation consiste en la détection des trois premières caméras clés. Une fois ces caméras détectées, la pose de la troisième est estimée au facteur près selon l'algorithme de [23], puis les points appariés entre cette dernière et la première sont triangulés. Alors, la pose de la seconde est estimée à partir de la correspondance entre les points détectés et leur triangulation. Puis sa pose est raffinée par ajustement de faisceaux.
Ensuite le système entre dans la boucle principale. Chaque nouvelle caméra est appariée avec la dernière caméra clé. Si le nombre de points appariés entre les deux est supérieur au seuil (défini à 300 dans [5]), la pose de la nouvelle est estimée à partir des correspondances 2D-3D. Sinon, le traitement est interrompu et la caméra précédente devient clé. Sa pose, et celles des deux précédentes clés, sont raffinées par un ajustement de faisceaux pour lequel les sept clés précédant les caméras ajustées sont observatrices. Le traitement de la dernière caméra reprend du début.

Algorithm 1

définir N, n, seuil
. capturer la première image, l'enregistrer comme clé, définir l'origine du repère
while $\#clés < 3$ **do**
  . capturer la dernière image
  . détecter les points d'intérêt
  . apparier avec la dernière clé
  . filtrer les appariements par [22]
  if $\#inliers < seuil$ **then**
    . définir la caméra précédente comme clé
  else
    . estimer la pose selon [23]
  if $\#clés == 3$ **then**
    . initialiser la carte
    . estimer la pose de la clé 2 par [19]
    . raffiner la pose de la clé 2 et la carte par ajustement de faisceaux
while $il\ reste\ des\ images\ à\ traiter$ **do**
  . capturer la dernière image
  . détecter les points d'intérêt
  . apparier avec la dernière clé
  . filtrer les appariements par [22]
  if $\#inliers > seuil$ **then**
    . estimer la pose de la caméra par [19]
    . passer à l'image suivante
  . définir la caméra précédente comme clé
  . observatrices = [clé(fin -N-n+1)..clé(fin -n)]
  . variables = [clé(fin -n+1)..clé(fin)]
  if Reconnaissance de la scène **then**
    . corriger la pose de la dernière clé
    . observatrices = [clé(fin -N-n+1)..clé(fin -n), clé(fin)]
    . variables = [clé(fin -n+1)..clé(fin-1)]
  . Ajustement de faisceaux(observatrices, variables)
  . trianguler les nouveaux points
  . compléter la carte
  . apparier la dernière image avec la dernière clé
  . filtrer les appariements et estimer la pose

Notre proposition consiste en l' ajout d'une étape facultative à ce système : régulièrement, une détection de scène est réalisée et suivie d'une relocalisation en cas de succès. Cette détection est faite à chaque ajout de pose clé, car l'ajout d'une pose clé signifie que la scène observée a signi-

1. http ://www.inf.ethz.ch/personal/chzach/opensource.html

ficativement changé, cela permet de ne pas surcharger la machine avec des calculs non nécessaires et enfin car l'ajout d'une pose clé comme la reconnaissance d'une scène doivent être suivis d'une étape de réduction d'erreur. Alors, une fusion d'information basée sur un ajustement de faisceaux est appliquée.
Une spécification de notre algorithme en pseudo-code est donnée dans l'algorithme 1. Notre étape est identifiée par un changement de couleur.

4 Protocole expérimental

Parce que l'innovation de notre démarche ne repose pas sur la reconnaissance de scène, nous mettons en place un protocole expérimental simplifié.
Nous prenons un flux vidéo en entrée de notre algorithme servant à contruire un modèle de l'environnement qui servira de vérité terrain en estimant les paramètres des poses de caméras clés. Ceci est réalisé selon un schéma *SLAM Structure From Motion* dans lequel chaque nouvelle pose est raffinée en prenant en compte l'ensemble des précédentes commes observatrices. Ainsi les caméras clés sont détectées selon le même principe que dans notre application *SLAM*. Cela permet de simuler une reconnaissance de scène aisément. Trois vues de notre scène d'expérimentation sont visibles à la figure 1, la reconstruction de référence de la scène est disponible à la figure 3 et une visualisation de la seule trajectoire de référence est montrée à la figure 2.

Figure 1 – *Quelques vues de la scène expérimentale*

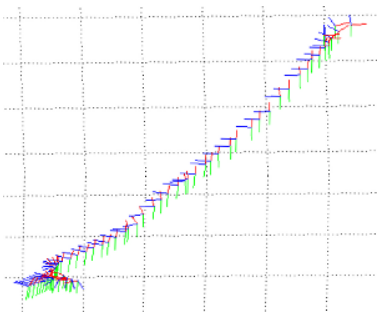

Figure 2 – *La trajectoire de référence, l'axe de visée des caméras est en bleu*

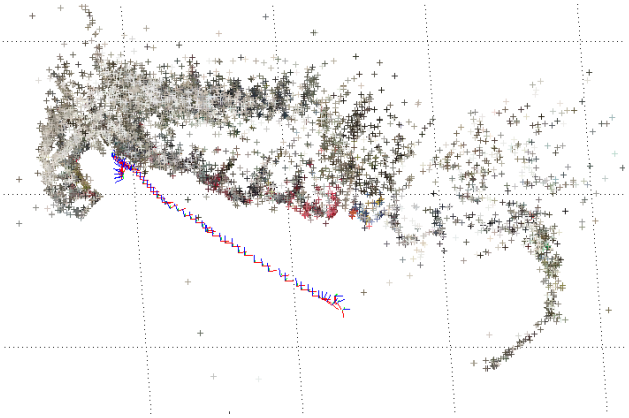

Figure 3 – *La reconstruction de référence*

Avec le même flux, nous pouvons ensuite exécuter une version dégradée, comparativement à celle de [5], de l'algorithme. En parallèle, à chaque détection de caméra clé, nous simulons une reconnaissance de la scène grâce à son identifiant et estimons la réduction d'erreur que l'application de notre solution apporte mais nous ne la prenons pas en compte dans la suite. La relocalisation consiste à affecter les paramètres de la pose de référence à la pose courante.
Dans la version dégradée de l'algorithme, seule la dernière caméra clé est optimisée et seules les deux précédentes sont observatrices. Dans la version expérimentale notre correction prend la dernière caméra clé comme seul paramètre fixe et les deux précédentes comme paramètres variables.
Nous considérons comme critère de qualité des paramètres de poses estimés la distance du centre optique à celui de la caméra correspondante dans le modèle. La pose d'une caméra étant représentée par la matrice $[R, T]$, on note $[R_{ref}, T_{ref}]$ la pose de la caméra de référence et $[R_e, T_e]$ la pose de la caméra estimée, l'erreur que nous mesurons est $\varepsilon_e = ||T_{ref} - T_e||_2$.

5 Résultats

Ayant constaté que le processus d'optimisation peut converger vers une solution aberrante, alors même qu'il est raisonnablement initialisé, nous faisons l'hypothèse que la distance entre deux caméras clés suit une loi normale. Nous ajoutons la condition que la norme de la translation après optimisation soit inférieure à deux fois l'écart type empirique de la séquence pour prendre en compte le résultat de l'ajustement de faisceaux. L'estimation du système dans la version dégradée ne se perd ainsi quasiment plus, néanmoins les paramètres estimés sont de mauvaise qualité. Nous avons alors appliqué notre solution sur ces données et avons comparé les différences d'erreur avec la version sans correction. Soient ε_{deg} l'erreur de la version dégradée, ε_{rel1} et ε_{rel2} les erreurs lorsque la relocalisation est effectuée respectivement une et deux clés plus tard, les figures 4 et 5 représentent $\varepsilon_{deg} - \varepsilon_{rel1}$ et $\varepsilon_{deg} - \varepsilon_{rel2}$ re-

spectivement en fonction des caméras clés.

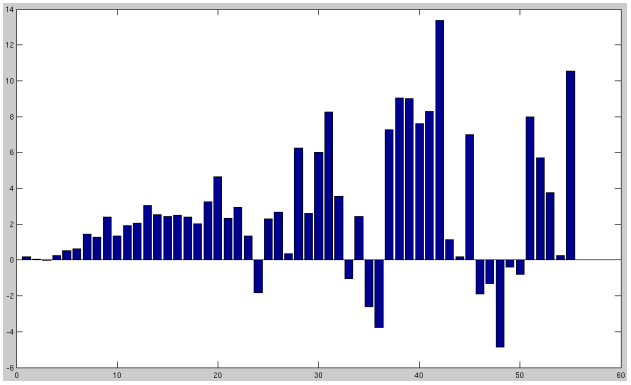

Figure 4 – *Réduction d'erreur, en ordonnée, apportée sur l'estimation des caméras, en abscisse, lorsque la suivante est relocalisée*

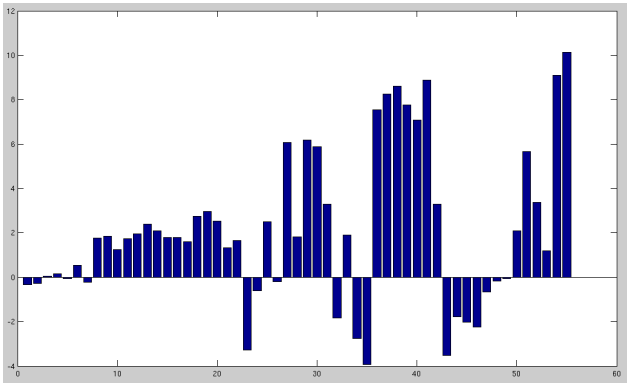

Figure 5 – *Réduction d'erreur, en ordonnée, apportée sur l'estimation des caméras, en abscisse, lorsque le système est relocalisé deux caméras plus tard*

On constate que pour environ la trentaine de premières caméras, la diminution d'erreur est effective et les poses de l'historique sont efficacement corrigées. Au delà, la solution n'est pas toujours efficace. En fait, dans ces cas le système a trop divergé pour être corrigé par notre approche, avec les paramètres définis. Ainsi, lorsque l'erreur que nous mesurons diminue l'aberration de l'état du système est simplement modifiée, mais pas diminuée.

Lors de la construction du modèle nous avons constaté que l'une des caméras clé, pourtant raisonnablement initialisée se retrouvait perdue après l'étape d'ajustement de faisceaux. Nous avons alors annulé l'optimisation de pose de cette caméra lors de l'étape de reconstruction. Cependant, créer un tel cas d'exception dans le traitement des caméras dans l'exécution en ligne de la version dégradée n'est pas envisageable. L'ajustement de faisceaux, et alors toute l'estimation ultérieure du modèle, va ainsi échouer. A la figure 6 nous traçons ε_{deg}, pour lequel nous n'imposons pas de condition sur la prise en compte des paramètres issus de l'ajustement de faisceaux, ε_{rel1} et ε_{rel2}. On constate que notre solution peut atténuer la perte du système si la caméra clé suivante est relocalisée. Cela ne fonctionne pas lorsque la caméra est relocalisée plus tard : cela est dû à ce que sur les trois caméras prises en compte lors de l'ajustement de faisceaux deux sont grandement corrompues, alors une seule juste et fixe ne suffit pas à les contrebalancer.

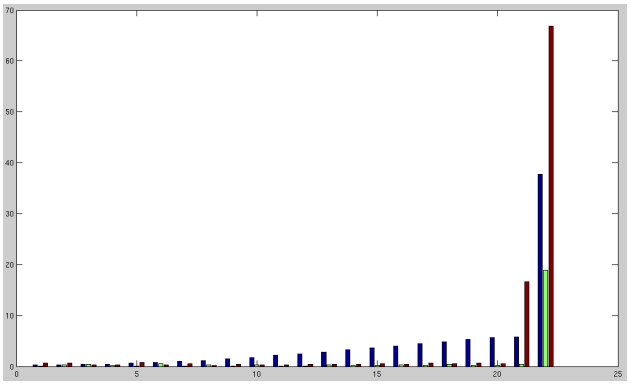

Figure 6 – *Les erreurs d'estimation des poses de caméra. En bleu : ε_{deg}, en vert : ε_{rel1}, en rouge : ε_{rel1}*

6 Limitations et travaux futurs

Notre proposition se confronte à plusieurs limitations. Notre approche nécessite de connaître le changement de base entre le repère du système et des modèles de scène. S'il est possible d'initialiser le système en observant une scène connue, c'est ce que nous avons fait, cela limite la diversité des cas d'utilisation. Une solution plus générale reste à trouver.

L'ajustement de faisceaux repose sur une approximation localement linéaire de l'espace des paramètres optimisés. C'est pour cela que la procédure nécessite une approximation suffisamment peu erronée de la solution pour converger. Alors, comme le montrent l'expérience, il existe un minimum, fonction du contexte, de dérive au delà duquel la correction des paramètres historiques échoue. L'étude de ce minimum est importante afin de pouvoir déterminer quand il est souhaitable d'appliquer notre proposition et quand il ne l'est pas. Lorsque cela ne l'est pas, car la dérive est trop importante, une solution serait plus probablement d'appliquer un schéma de *SLAM-SFM* remontant l'historique des poses de caméras et tirant parti du facteur d'échelle fourni par la scène. Bien sûr, la dérive peut se produire en sens inverse et il reste à investiguer comment tirer parti de la première estimation pour l'éviter.

Nous nous sommes restreints dans nos expérimentations à ne tirer parti que des informations apportées par les poses de référence. L'apport d'information fourni par l'utilisation de points de modèles est à étudier. Il fait peu de doutes que la correction des poses erronées n'en serait qu'améliorée. De même lorsque nous nous sommes attachés à propager

l'information nous n'avons fixé que la dernière caméra clé. Nous avons constaté à la figure 6 que très rapidement cela n'était pas suffisant. Il serait certainement avantageux de garder des poses de l'historique comme observatrices. Cela permettrait de contrebalancer plus durablement l'effet des caméras corrompues.
Par ailleurs, dans le présent article, nous avons utilisé un schéma de fusion de données assez simple. Notre contribution tirerait probablement profit de l'utilisation de schémas plus élaborés et plus efficaces tels que présentés dans [16]. Egalement, dans [14] et [16] les auteurs utilisent une centrale inertielle ou un capteur GPS pour robustifier leur estimations de paramètres, mais l'usage conjoint des deux selon la méthode de [16] reste à être mis en oeuvre.
Enfin, toujours dans une optique de réduction de la dérive et parce qu'elle se manifeste beaucoup sur le facteur d'échelle, l'usage d'un accéléromètre semble pertinent. Supposant la position initiale du système connue, puis celle de la caméra clé précédente et parce que le temps écoulé entre deux caméras clés est mesurable, la vitesse à l'issue de l'étape d'initialisation, puis entre deux caméras clés successives peut être calculée. Alors, à partir des accélérations linéaires, il est possible d'estimer le déplacement relatif entre deux caméras puis d'incorporer la norme dans l'ajustement de faisceaux comme paramètre fixe à la manière de [14] ou [16].

Références

[1] H. Durrant-Whyte et T. Bailey. Simultaneous localization and mapping : Part1. *IEEE Robotics and Automation Magazine*, 2006.

[2] R. Hartley B. Triggs, P. McLauchlan et A. Fitzgibbon. Bundle adjustement - a modern synthesis. Dans *Vision Algorithms*, 1999.

[3] A. J. Davison. Real-time simultaneous localization and mapping with a single camera.

[4] J. Bergen D. Nister, O. Naroditsky. Visual odometry. *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[5] M. Dhome F. Dekeyser P. Sayd E. Mouragnon, M. Lhuillier. Real time localization and 3d reconstruction. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[6] J.M.M. Montiel H. Strasdat et A.J. Davison. Real-time monocular slam : Why filter ? Dans *ICRA*, 2010.

[7] A. Eudes et M.Lhuillier. Error propagations for local bundle adjustment. Dans *CVPR*, pages 2411–2418, 2009.

[8] Pierre Lothe, Steve Bourgeois, Fabien Dekeyser, Eric Royer, et Michel Dhome. Towards geographical referencing of monocular slam reconstruction using 3d city models : Application to real-time accurate vision-based localization. Dans *CVPR*, pages 2882–2889, 2009.

[9] R. I. Hartley et A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN : 0521540518, second édition, 2004.

[10] D. Stricker G. Bleser. Advanced tracking through efficient image processing and visual-inertial sensor fusion. *VR*, 2008.

[11] Alexandre Eudes, Maxime Lhuillier, Sylvie Naudet-Collette, et Michel Dhome. Fast odometry integration in local bundle adjustment-based visual slam. Dans *ICPR*, pages 290–293, 2010.

[12] Pierre Lothe, Steve Bourgeois, Eric Royer, Michel Dhome, et Sylvie Naudet-Collette. Real-time vehicle global localisation with a single camera in dense urban areas : Exploitation of coarse 3d city models. Dans *CVPR*, pages 863–870, 2010.

[13] Michela Farenzena, Adrien Bartoli, et Youcef Mezouar. Efficient camera smoothing in sequential structure-from-motion using approximate cross-validation. Dans *ECCV (3)*, pages 196–209, 2008.

[14] J. Michot. *Recherche linéaire et fusion de données par ajustement de faisceaux*. Thèse de doctorat, Université Blaise Pascal, 2010.

[15] Julien Michot, Adrien Bartoli, et François Gaspard. Bi-objective bundle adjustment with application to slam. Dans *3DPVT*, 2010.

[16] Maxime Lhuillier. Fusion of gps and structure-from-motion using constrained bundle adjustments. Dans *CVPR*, pages 3025–3032, 2011.

[17] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, et Richard Szeliski. Building rome in a day. Dans *ICCV*, pages 72–79, 2009.

[18] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, et Horst Bischof. From structure-from-motion point clouds to fast location recognition. Dans *CVPR*, pages 2599–2606, 2009.

[19] Vincent Lepetit, Francesc Moreno-Noguer, et Pascal Fua. Ep*n*p : An accurate $o(n)$ solution to the p*n*p problem. *International Journal of Computer Vision*, 81(2) :155–166, 2009.

[20] D. Murray G. Klein. Parallel tracking and mapping for small a.r. workspaces. *International Symposium on Mixed and Augmented Reality*, 2007.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[22] Martin A. Fischler et Robert 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.

[23] D. Nister. An efficient solution to the five-point relative pose problem. *PAMI*, 2004.