# Suivi de caméra stéréo pour des applications de réalité augmentée sur téléphone mobile

Simone Gasparini, Pascal Bertolino

# Suivi de caméra stéréo pour des applications de réalité augmenté sur téléphone mobile

S. Gasparini[1] P. Bertolino[1]

[1] GIPSA-Lab, Université de Grenoble, CNRS

11 rue des Mathématiques,
Grenoble Campus BP46,
F-38402 St Martin d'Hères cedex

simone.gasparini@gmail.com, pascal.bertolino@gipsa-lab.fr

## Résumé

*Nous présentons notre recherche en cours sur un algorithme de suivi dédié à des équipements mobiles pourvus d'une caméra stéréo. L'algorithme fonctionne en temps réel sur un prototype de plate-forme et il peut être utilisé comme moteur d'applications de réalité augmentée. Afin de s'adapter aux ressources limitées disponibles, nous avons conçu un algorithme qui utilise la caméra stéréo uniquement pour la reconstruction 3D des points, alors que le suivi des points n'est réalisé que sur une des deux images, permettant ainsi de réduire les temps de traitement. Nous montrons quelques résultats préliminaires dans lesquels le suivi de caméra a été validé avec un scénario réaliste et qui démontrent une robustesse bien adaptée à une application de réalité augmentée.*

## Mots Clef

Application de vision par ordinateur, Caméra stéréo, Suivi de caméra, Dispositifs mobiles, Réalité Augmentée.

## Abstract

*We present our current work on a camera tracking algorithm designed for a mobile device equipped with a stereo camera. The tracker runs in real-time on a prototype mobile platform and it can be used as the core engine of augmented reality applications. In order to cope with the limited resources available, we design an algorithm that relies on the stereo camera only for the 3D reconstruction of points, while the point tracking is performed only on one of the two images, thus reducing the computational effort. We show some preliminary results in which the camera tracker has been validated in a realistic scenario and it is proved to have an adequate robustness for an augmented reality application.*

## Keywords

Computer vision application, Stereo camera, Camera tracking, Mobile devices, Augmented reality.

## 1 Introduction

Ces dix dernières années, la diffusion à grande échelle d'équipements mobiles pourvus de caméras, tels que les *smart phones* et les tablettes a contribué à l'augmentation des applications de vision par ordinateur destinées à ce nouveau type de matériel. L'application la plus remarquable est la Réalité Augmentée [21], qui permet d'améliorer l'expérience utilisateur en incrustant des objets virtuels dans les images. Les domaines et contextes d'applications sont nombreux, parmi lesquels l'affichage d'information sur les produits dans les magasins, le jeu vidéo et la navigation urbaine pour les piétons [23].

Pour que le rendu des objets virtuels soit réaliste, c'est-à-dire qu'ils soient perçus par l'utilisateur comme faisant partie de la scène [7], la position, l'orientation et le mouvement de l'appareil doivent être connus à chaque instant.

Dans le cas de téléphones mobiles munis d'une caméra, le système doit être capable de détecter et suivre les mouvements de l'utilisateur.

Selon le niveau de précision désiré pour le suivi de mouvement de l'appareil, différentes solutions ont été proposées. Les matériels récents sont souvent équipés de capteurs de mouvements tels que des accéléromètres et un magnétomètre faisant office de boussole. Le couplage de ces capteurs fournit en continu à la fois le mouvement et l'orientation. Malheureusement les capteurs inertiels seuls ne sont pas très précis car ils fournissent des mesures très bruitées lorsque le mouvement est lent, et les champs magnétiques peuvent interférer avec le magnétomètre [8].

Une approche classique pour calculer le mouvement et l'orientation des équipements mobiles consiste à utiliser l'image fournie par la caméra embarquée pour suivre dans la scène des points fixes et exploiter leur mouvement apparent pour estimer le mouvement de la caméra [12].

Cette technique est connue sous le nom de *camera tracking* ou suivi de caméra et se compose de quatre étapes principales : (*i*) détection de points d'intérêt dans différentes images, (*ii*) mise en correspondance de ces points entre ces

images et reconstruction 3D, (*iii*) suivi des points dans les nouvelles images, et (*iv*) estimation du mouvement et de l'orientation de la caméra. Éventuellement, un traitement d'optimisation (Ajustement de faisceaux ou *Bundle Adjustment* [35]) est rajouté pour maintenir une consistance globale entre les points 3D et les positions de la caméra.

La technique de suivi présentée ci-dessus permet de fournir une estimation plus sure et plus stable du mouvement de la caméra, notamment pour des mouvements sans à-coups. Néanmoins, d'autres facteurs tels que le changement des conditions d'éclairement, l'occultation d'objets, le flou dû au mouvement ou des environnements peu texturés peuvent affecter la robustesse du suivi.

Souvent, le suivi de caméra et les capteurs de mouvement sont couplés [18, 38] pour combiner les avantages des deux approches et limiter leurs défauts. Dans ce cas, la caméra et les capteurs doivent être mutuellement calibrés [17].

Dans une approche basée uniquement sur la vision, la détection des points caractéristiques dans la scène est souvent facilitée par des marqueurs spéciaux qui sont facilement identifiables dans l'image [6, 10]. Le suivi de caméra à base de marqueurs est très robuste et est une approche classique utilisée dans les applications de réalité augmentée. En outre, des méthodes optimisées [37] ont été proposées pour réduire les calculs relatifs à leur détection, permettant aux appareils mobiles de détecter ces marqueurs en temps réel.

L'inconvénient majeur de cette solution est que la scène doit être équipée de marqueurs, ce qui n'est pas toujours possible, par exemple lorsque le suivi doit être réalisé dans des grands espaces ou pour des applications de navigation en extérieur.

Dans la littérature, on trouve de nombreux travaux traitant de solutions de suivi de caméra sans marqueurs, la plupart développées pour des PC de bureau [9, 19, 26, 28]. Ces méthodes qui sont rassemblées sous l'appellation VisualSLAM reposent soit sur le filtre de Kalman étendu (FKE), soit sur l'ajustement de faisceaux (AF) comme traitements d'optimisation. Alors que l'ajustement de faisceaux est capable de gérer plus de points que le FKE, étant par conséquent plus précis [33], les deux méthodes requièrent une puissance de calcul importante et ne sont pas destinées aux appareils mobiles dont les ressources sont limitées.

La méthode proposée par Klein et Murray dans [19] a été adaptée et portée sur iPhone [20]. Les auteurs limitent le problème posé par la consommation de ressources trop élevée de l'ajustement de faisceaux en dédiant un *thread* au traitement de suivi et en réduisant la dimensionnalité du problème d'optimisation.

Dans ce papier, nous présentons les premiers résultats de travaux en cours sur un suivi de caméra pour équipement mobile doté d'une caméra stéréo. Ces dernières années, la vision 3D s'est popularisée et plusieurs dispositifs mobiles équipés d'une caméra stéréo et d'un écran autostéréoscopique ont fait leur apparition sur le marché, notamment les smartphones LG Optimus 3D P920 et HTC EVO 3D, ou la nouvelle console Nintendo 3DS ; ces systèmes, par contre, sont plutôt destinés à la prise et à la visualisation de photos et vidéos en 3D et pas encore à de vraies applications de vision par ordinateur stéréoscopique.

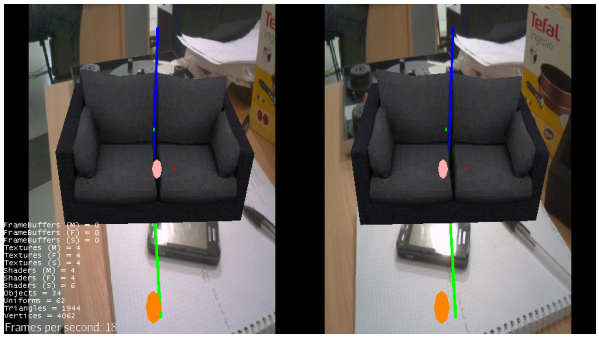


FIGURE 1 – Exemple d'application finale avec un objet virtuel incrusté dans une paire d'images stéréo affichées côte à côte sur l'écran d'un téléphone mobile.

Cela ouvre la voie à toute une panoplie d'applications intéressantes et stimulantes qui pourraient améliorer l'expérience utilisateur et l'interaction de celui-ci avec le monde qui l'entoure. En particulier, nous travaillons sur un prototype de plate-forme mobile développé par ST-Ericsson. Nous développons une méthode de suivi de caméra qui exploite la configuration stéréo et qui peut être utilisée pour des applications de réalité augmentée stéréo développées par nos partenaires industriels, notamment la société Pointcube (Figure 1).

Dans le contexte du suivi de caméra, les systèmes stéréo sont intéressants car ils peuvent fournir en continu la profondeur 3D de la scène. En revanche, ils demandent plus de puissance de calcul puisqu'ils traitent deux images au lieu d'une. On trouve dans la littérature quelques travaux qui ont adapté les techniques du VisualSLAM aux systèmes stéréo [29, 25]. Ces recherches s'intéressent principalement aux applications en robotiques et donc au suivi de caméra dans des environnements de grande taille. La méthode proposée par Nistér et Stewenius [29] utilise un suivi basé sur de la mise en correspondance : les points caractéristiques extraits dans la paire d'images stéréo sont utilisés pour réaliser une première reconstruction des points. Ensuite, pour chaque nouvelle paire d'images, de nouveaux points caractéristiques sont extraits puis mis en correspondance avec les points de la paire précédente, dans le but d'estimer le mouvement de la caméra.

Les auteurs fusionne à cette estimation visuelle les données provenant des capteurs inertiels et du GPS dans un traitement d'ajustement de faisceaux. Globalement, le système fonctionne à 13 images par seconde sur un PC standard et n'est pas envisageable sur du matériel mobile. La méthode proposée par Mei *et al.* [25], quant à elle, utilise la reprojection des points 3D reconstruits à l'étape précédente pour estimer la position de la caméra stéréo et pour chaque nouvelle paire d'images, elle reconstruit également de nou-

veaux points 3D qui sont ajoutés à l'ensemble de points déjà suivis. Les points d'intérêt sont représentés avec des descripteurs SIFT de façon à garantir la relocalisation du robot quand celui-ci est perdu. Cette méthode nécessite un PC Quad Core pour fonctionner en temps réel.

Développer un suivi de caméra pour une machine mobile nécessite de concevoir un algorithme qui doit répondre aux contraintes liées aux ressources de calcul et mémoire limitées, tout en garantissant un niveau de robustesse adapté à l'application. Aussi, la principale contribution de ce travail est un algorithme original capable d'estimer le mouvement de la caméra stéréo en temps réel et de façon robuste. À notre connaissance, il s'agit de la première fois qu'un algorithme de suivi de caméra est conçu pour une plate-forme mobile équipée d'une caméra stéréoscopique.

Le papier est organisé de la façon suivante : la Section 2 présente les principaux aspects de notre approche de suivi de caméra ainsi que certains détails de l'algorithme proposé. La Section 3 présente quelques expérimentations préliminaires ainsi que la validation de notre méthode sur un prototype. La Section 4 conclut en discutant sur certains points et sur la suite à donner à notre travail.

# 2 Suivi de Caméra Stéréo

L'algorithme de suivi se compose de deux étapes principales, l'initialisation puis le suivi à proprement parler. L'étape d'initialisation n'est exécutée qu'une seule fois, pour la première image. Elle consiste essentiellement en une première reconstruction 3D stéréo des points qui seront suivis dans les images suivantes. Il faut souligner que, afin de limiter les calculs, nous traitons les deux images de la paire stéréo uniquement lorsque c'est nécessaire, c'est-à-dire lorsqu'il y a trop de points caractéristiques perdus pendant le suivi et que de nouveaux points 3D doivent être reconstruits. En revanche, durant le suivi des points, seule une des deux images de la paire stéréo est utilisée (nous avons arbitrairement choisi l'image gauche).

Dans notre approche, la caméra stéréo est calibrée lors d'une étape préliminaire. Nous avons développé une application Android indépendante pour calibrer la caméra stéréo à l'aide d'image de plans (un damier par exemple) en suivant la méthode proposée par [34], disponible dans la bibliothèque OpenCV . La méthode estime à la fois les paramètres intrinsèques et extrinsèques des deux caméras, ainsi que la distorsion de chaque optique. Comme généralement la longueur focale des caméras employées sur les *smart phones* ne varie pas (le zoom est réalisé numériquement) et les deux caméras étant fixes, la calibration ne doit être réalisée qu'une seule fois.

Le modèle de projection est alors représenté par les deux matrices de caméra $\mathtt{K}^l$ et $\mathtt{K}^r$, ainsi que la rotation $\mathtt{R}$ et le vecteur de translation $\mathbf{t}$ reliant les deux caméras, de telle façon à ce qu'un point générique $\mathbf{Q}$ dans l'espace est projeté dans deux points images dépourvus de distorsion $\mathbf{q}^l$ et $\mathbf{q}^r$ :

$$\begin{aligned}\mathbf{q}^l &\sim \mathtt{K}^l \begin{bmatrix}\mathtt{I} & \mathbf{0}\end{bmatrix} \mathbf{Q} \\ \mathbf{q}^r &\sim \mathtt{K}^r \begin{bmatrix}\mathtt{R} & \mathbf{t}\end{bmatrix} \mathbf{Q},\end{aligned}$$

où $\mathtt{I}$ est la matrice identité $3 \times 3$ et $\mathbf{0}$ le vecteur nul $3 \times 1$. Comme indiqué plus haut, nous considérons la caméra gauche comme étant le système de référence, et tous les points reconstruits seront représentés dans ce référentiel. Les sections suivantes décrivent en détails chaque étape de notre algorithme.

## 2.1 Initialisation

L'étape d'initialisation n'est exécutée qu'au début du suivi pour détecter les points caractéristiques avec la caméra stéréo et les reconstruire en 3D, pour qu'ils soient ensuite suivis dans les images suivantes et utilisés pour estimer le mouvement et la pose de la caméra.

Dans le cas d'une caméra monoculaire, lors de l'initialisation, l'utilisateur doit bouger la caméra pour avoir au moins une paire d'images, séparées d'un déplacement significatif, permettant ainsi une reconstruction 3D meilleure et plus fiable [26, 19, 20]. En revanche, l'utilisation d'une caméra stéréo évite à l'utilisateur toute interaction et rend cette étape complètement transparente. Les points sont extraits dans les deux images de la paire stéréo et sont mis en correspondance entre images gauche et droite. Une triangulation permet d'obtenir un ensemble d'association points 2D-3D.

**Extraction de points d'intérêt.** A l'heure actuelle, il existe un large choix à la fois d'extracteurs de points caractéristiques et de leur représentation (descripteurs) [36]. Les plus fiables sont SIFT [24] et SURF [2], qui sont robustes aux rotations et aux déformations perspectives. En revanche, ces outils sont coûteux en temps de traitement et ne peuvent être utilisés dans un contexte temps réel, notamment sur des systèmes mobiles. Récemment, une nouvelle gamme de descripteurs et d'extracteurs de points caractéristiques ont été proposés pour palier cette limitation, tout en maintenant des performances acceptables, souvent comparables à celles de SIFT et SURF . Les plus notables sont BRIEF [5], ORB [32], FREAK [1] et BRISK [22]. Une revue détaillée et une comparaison de ces descripteurs pour les applications de suivi de caméra sont proposées dans [13].

Dans notre algorithme, nous avons choisi une combinaison des points caractéristiques FAST [31] et des descripteurs BRIEF , puisque lors de nos expérimentations, c'était la combinaison qui fournissait le meilleur compromis entre vitesse et taux de reconnaissance. FAST est réputé pour être un extracteur de coins très rapide ayant une bonne répétabilité et une bonne robustesse face aux changements d'éclairage [13]. Le descripteur BRIEF a été introduit récemment. Il décrit l'emplacement du point caractéristique à l'aide de descripteurs binaires résultant d'une série de tests (binaires) aléatoires sur les pixels de cet emplacement. Le descripteur obtenu pour un point forme une série compacte de bits (128) qui peut être mise en correspondance avec

d'autres séries, en utilisant la distance de Hamming, très rapide à calculer.
Finalement, pour garantir une distribution uniforme des points dans toute l'image, celle-ci est divisée en cellules (par une grille $4 \times 4$ dans notre cas) et dans chaque cellule sont pris en compte au plus les `MaxFeat` points détectés.

**Mise en correspondance des points.** Une fois les points extraits et leur descripteur calculé, dans les deux images, les descripteurs sont mis en relation pour établir la correspondance entre les points des deux images de la paire stéréo. Pour chaque point d'une image, son descripteur BRIEF est comparé avec les descripteurs de l'autre image et celui qui fournit la distance minimale est considéré comme étant un correspondant candidat.
Cependant, cette mise en correspondance exhaustive basée sur la plus courte distance fournit souvent beaucoup de correspondances erronées. Pour améliorer la robustesse de cette partie du traitement, nous relâchons la contrainte de distance minimale pour considérer uniquement les paires de candidats qui se voient mutuellement comme une bonne correspondance : pour chaque descripteur de point dans les deux images, ses `k` plus proches voisins (selon la distance de Hamming) dans l'autre image sont calculés avec l'algorithme FLANN [27]. Puis, pour chaque descripteur $\mathbf{d}_i^l$ de l'image gauche, on considère ses `k` voisins $\mathbf{d}_j^r$ dans l'image de droite, et on sélectionne comme correspondant candidat le descripteur $\mathbf{d}_j^r$ qui a $\mathbf{d}_i^l$ parmi ses `k` voisins. Dans nos expérimentations, nous utilisons $\texttt{k} = 4$.
Notons que dans les applications de stéréovision, la mise en correspondance de points peut être facilitée en rectifiant les deux images, de telle façon à ce que la recherche du point correspondant soit limitée aux points appartenant à la même ligne. Dans notre cas, nous avons observé que la rectification d'image réalisée au niveau logiciel introduisait un surcoût important de calculs. D'autre part, sur la plate-forme mobile que nous utilisons, une rectification réalisée au niveau matériel est en cours de développement. Ce nouveau module fournira très bientôt de façon native des images déjà rectifiées sans perte de performances.

**Reconstruction 3D.** Les points correspondants trouvés à l'étape précédente sont triangulés pour reconstruire géométriquement les points 3D de la scène. Tout d'abord, tous les points des deux images sont corrigés : la distorsion due à l'optique est enlevée selon les coefficients de distorsion estimés de chaque caméra. Les points 3D sont ensuite calculés avec la méthode classique de triangulation [16].
Comme l'ensemble des couples de points peut encore contenir des correspondances incorrectes ou des aberrations nous filtrons ces couples à l'aide de deux critères : le test de chiralité et l'erreur de reprojection. Le test de chiralité assure que le point reconstruit est une solution physiquement possible de la triangulation, autrement dit que le point 3D reconstruit est bien devant chacune des deux caméras. Si un couple de points correspondant est incorrecte, il peut générer un point 3D qui se situe derrière une des deux caméras ou derrière les deux. Tous les couples ne respectant pas ce test ne sont pas pris en compte dans la suite des traitements.
Pour chaque point 3D, nous calculons également son erreur de reprojection et nous écartons tous ceux dont l'erreur dans les deux images est supérieure à un seuil donné $\texttt{repThreshold} = 2\text{pix}$.

## 2.2 Suivi

Les points détectés lors de l'étape d'initialisation et associés à des points 3D valides sont suivis dans les images suivantes à l'aide de l'algorithme de Lucas-Kanade (LK) basé sur le flux optique de l'image gauche seulement, pour réduire les coûts de calcul. La nouvelle position des points dans l'image est utilisée pour estimer la nouvelle pose et orientation de la caméra. .
Durant le suivi, des points peuvent être perdus suite à une occultation, des mouvements amples par rapport à la caméra ou simplement par échec de l'algorithme LK. Afin de maintenir un nombre suffisant de points suivis et ainsi préserver la fiabilité de l'estimation de mouvement, lorsque le nombre de points suivis devient inférieur à un seuil $\texttt{minFeat} = 20$, nous extrayons un nouvel ensemble de points. Comme dans l'étape d'initialisation, la paire stéréo fournit de nouveaux points caractéristiques qui sont mis en correspondance et donnent de nouveaux points 3D. Contrairement à l'étape d'initialisation, ici, nous avons aussi implémenté un algorithme qui permet de réassocier les nouveaux points avec des points 3D déjà connus mais qui pour diverses raisons (occultations, sortie du champ de la caméra, ...) n'étaient plus suivis.

**Suivi de points.** Les points caractéristiques de l'image gauche sont suivis à l'aide d'une version parcimonieuse de l'algorithme de suivi de Lucas-Kanade utilisant une représentation pyramidale des images [3]. La méthode du flot optique est généralement un algorithme lourd en calculs. La version parcimonieuse calcule uniquement le flot optique sur une petite zone autour du point à suivre. Cela permet d'accélérer le traitement. De plus, en utilisant une représentation pyramidale, plusieurs facteurs d'échelle permettent de prendre en compte des changements d'échelle dus à une variation de distance entre la caméra et la scène. Durant nos expérimentations, nous avons trouvé qu'un bon compromis entre vitesse et robustesse était obtenu avec un flot calculé sur une zone de $11 \times 11$ pixels autour du point et en utilisant 2 niveaux de la pyramide.

**Estimation de la pose.** Une fois que la nouvelle position des points est calculée, l'estimation de la pose de la caméra, connaissant la projection des points 3D sur le plan image, est un problème bien connu en vision par ordinateur sous le nom de PnP (*Perspective-N-Points*) [15]. Dans notre implémentation, nous utilisons la méthode proposée par Grunert [14] pour calculer une estimée initiale de la pose ave un ensemble de 3 points minimum. L'algorithme RANSAC [11] est utilisé pour filtrer les éventuels points aberrants et affiner la solution initiale.

**Réassociation de points.** Si le nombre de points suivis est inférieur au seuil `minFeat` la paire stéréo courante est utilisée pour détecter de nouveaux points caractéristiques et reconstruire de nouveaux points 3D. Nous suivons une procédure similaire à l'étape d'initialisation (*c.f.* Section 2.1) : extraction des points avec FAST et calcul de leur descripteur BRIEF . Nous gardons seulement les points qui sont à une distance minimale de `minDist` $= 15$pix des points suivis actuels. Cela évite la redondance de points et garantit une distribution plus uniforme des points dans l'image.

Ensuite, nous considérons l'ensemble de tous les points 3D qui ont été reconstruits jusque là et qui sont visibles de la caméra mais non associés à un des points suivis. La visibilité des points 3D est calculée en considérant la position et l'orientation réelle de la caméra et son champ de vision, selon les données de calibration. Ces points 3D sont projetés dans l'image de gauche : si la projection de ce point dans l'image est à une distance d'un point nouvellement détecté inférieure à un seuil `repThreshold`, alors ce point est assigné au point 3D et il est ajouté à l'ensemble des points suivis. De façon à accélérer cette recherche, les nouveaux points détectés dans l'image de gauche sont rangés dans un arbre KD et l'algorithme FLANN permet pour un point projeté, de rechercher dans l'arbre le point le plus proche.

Il faut noter que cet algorithme de réassociation peut générer des fausses associations, puisque par exemple des occultations entre points 3D ne sont pas prises en compte. Cependant, le système est globalement robuste aux fausses associations : lorsque des points faussement associés sont suivis dans l'image suivante, ils sont rejetés par le traitement RANSAC de l'estimation de la pose, qui est robuste aux points aberrants.

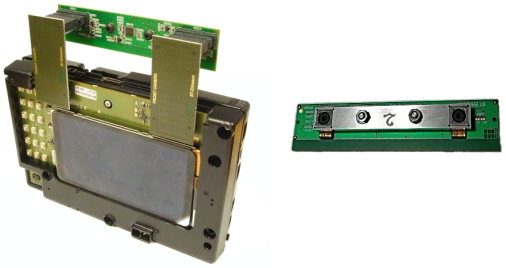
Les nouveaux points restants sont alors mis en correspondance avec ceux de l'image droite, triangulés et ajoutés à l'ensemble des points suivis, comme décrit en Section 2.1.

# 3 Résultats expérimentaux

Le suivi de caméra présenté a été implémenté sur un prototype de plate-forme mobile développé par ST-Ericsson (Figure 2). La plate-forme est équipée d'un processeur d'application U8500 contenant un CPU dual-core ARM® CORTEXTM-A9 avec NEON, cadencé à 800MHz et un GPU ARM® MALI-400 cadencé à 400MHz.

Le système d'exploitation est Android OS 2.4.3 et la plate-forme est équipée d'une platine supportant deux caméras de 5.3Mpix espacées de 6.5cm. La plate-forme fournit un flux vidéo stéréo composé de deux images de $640 \times 480$ pixels placées l'une dessus l'autre, à un débit de 30fps (*c.f.* Figure 3).

Notre algorithme de suivi de caméra a été implémenté en C++ en utilisant plusieurs fonctions des bibliothèques OpenCV [4] pour la détection de points, le calcul des descripteurs et l'algorithme LK. Il faut noter que pour l'instant, les optimisations possibles avec le GPU et le *multi-threading* n'ont pas été utilisées. Nous avons uniquement tiré avantage de la parallélisation TBB [30] utilisée dans la fonction OpenCV calculant le flot optique parcimonieux.



(a) La plate-forme mobile (b) La caméra stéréo

FIGURE 2 – Le prototype mobile (a) utilisé pour notre suivi de caméra est équipé d'une platine stéréo (b) ayant une distance inter-oculaire de 6.5cm.

L'application principale est développée en JAVA et elle traite l'image stéréo en appelant une fonction JNI native de la bibliothèque de suivi de caméra écrite en C++.

Afin d'estimer visuellement la qualité et la stabilité du suivi de caméra, nous avons réalisé des tests préliminaires en dessinant un cube dans l'image stéréo comme le montre la Figure 3. Durant l'étape d'initialisation, on considère les points 3D reconstruits et on détermine le plan dominant de la scène : RANSAC est à nouveau utilisé pour plaquer un plan sur les points (la grille de la Figure 3), puis nous dessinons un cube en fil de fer sur ce plan. La position initiale du cube est choisie arbitrairement en intersectant le plan avec l'axe des $z$ de la caméra gauche (*i.e.* l'axe optique de la caméra). Le point d'intersection est pris comme origine du cube alors que son orientation est choisie de façon à ce que son axe des $x$ soit parallèle à l'axe des $x$ de la caméra gauche. A chaque nouvelle image, le cube est dessiné en tenant compte du mouvement de la caméra. Les vidéos avec plusieurs résultats préliminaires sont disponibles à l'adresse suivante : `http://goo.gl/11UdM`.

Lors de nos résultats expérimentaux préliminaires, le suivi de caméra fonctionne en temps réel, à une cadence de 15 images par seconde. La Table 1 montre un découpage de l'algorithme suivant ses fonctions les plus pertinentes et leur temps d'exécution moyen. Ce temps a été calculé en relevant l'horloge processeur avant et après l'appel de chaque fonction. Notons que durant une exécution type de l'algorithme, le suivi par l'algorithme de LK est le plus consommateur avec un temps d'exécution moyen de $\sim$ 60ms. Cela n'est pas surprenant car la détermination du flot optique nécessite des calculs intensifs. Le temps d'exécution dépend également du nombre de points suivis, comme on peut le voir dans la Figure 4 : le temps du suivi LK décroît au fur et à mesure que des points sont perdus et il atteint son maximum après incorporation de nouveaux points lors de la procédure de réassociation (ce qui correspond aux pics du graphe). Dans notre configuration, nous avons jusqu'à 300 points. Une alternative au suivi de

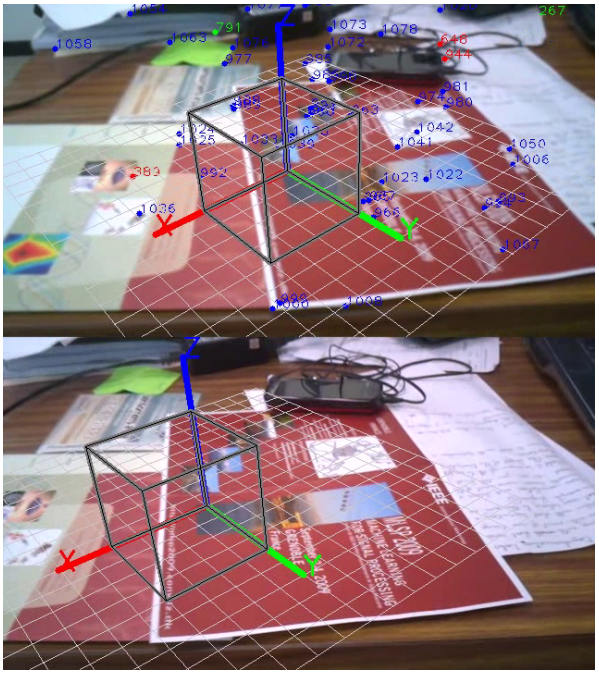


FIGURE 3 – Pour évaluer visuellement la qualité du suivi, un cube fil de fer est incrusté sur le plan dominant de la scène, déterminé lors de l'étape d'initialisation.

points consiste à extraire les points dans chaque nouvelle image, de calculer leur descripteur et de les mettre en correspondance avec ceux de l'image précédente. Comme le montre la Table 1, ces trois opérations peuvent être aussi coûteuses que l'approche avec le flot optique. D'autre part, nous avons remarqué que l'approche du suivi par mise en correspondance était moins robuste que celle utilisant le flot optique, car elle génère plus de points aberrants et de fausses correspondances stéréo.

Le traitement de réassociation est le plus coûteux en calculs car il doit traiter les deux images de la paire stéréo pour extraire les points et calculer leur descripteur : comme le montre la Table 1, ces deux opérations demandent $\sim 50$ms pour chaque image et jusqu'à $\sim$ 20ms pour leur mise en correspondance. Cependant, la réassociation n'est pas effectuée à chaque image mais seulement lorsque trop de points sont perdus lors du suivi. Ceci est lié au mouvement de la caméra, et donc pour des mouvements fluides, de meilleures cadences sont obtenues. Dans nos expérimentations, nous avons remarqué qu'une réassociation est généralement réalisée toutes les $50-60$ images, ce qui correspond aux pics de la Figure 4.

Pour évaluer numériquement la qualité du suivi de caméra, nous avons évalué l'erreur entre une position de départ estimée et une position d'arrivée estimée de la caméra lorsque celle-ci effectue une boucle dans l'espace. Comme nous n'avions à notre disposition aucun système permettant un mouvement connu et contrôlé (comme un bras de robot) nous avons déplacé manuellement la plate-forme pour lui faire faire une trajectoire en boucle qui la ramène dans sa

| Fonction | Moyenne | Max | Min |
|---|---|---|---|
| Extraction de points | 30.19 | 77.53 | 13.11 |
| Calcul des descripteurs | 24.96 | 33.32 | 21.19 |
| Mise en correspondance | 17.49 | 21.8 | 10 |
| Suivi LK | 58.36 | 126.89 | 37.01 |
| Estimation de pose | 16.03 | 83.87 | 0.43 |
| Réassociation | 147.43 | 188.81 | 115.06 |

TABLE 1 – Statistiques sur les temps d'exécution des fonctions principales de notre algorithme. Le temps est exprimé en [ms].

| Séquence | Nombre d'images | Distance parcourue | Erreur |
|---|---|---|---|
| Séquence 1 | 412 | 4.62m | 50.74mm |
| Séquence 2 | 454 | 5.37m | 73.09mm |
| Séquence 3 | 328 | 3.54m | 29.11mm |
| Séquence 4 | 409 | 3.61m | 18.56mm |

TABLE 2 – Erreurs réalisées en fermant une boucle pour différentes séquences vidéo.

position initiale. Lors de nos essais l'erreur entre les positions de départ et d'arrivée étaient de l'ordre de 20mm à 80mm sur une trajectoire moyenne de 4m (voir Table 2). La Figure 5 montre quelques exemples de ces trajectoires reconstruites.

# 4 Conclusion et perspectives

Dans ce papier, nous avons présenté une recherche en cours visant à développer un suivi de caméra pour un système mobile équipé d'une caméra stéréo. L'algorithme proposé a été conçu pour les capacités limitées d'un environnement mobile. Après une étape d'initialisation où les images stéréo sont utilisées pour reconstruire un premier ensemble de points 3D, le suivi est limité à l'image gauche afin de limiter les temps de traitement. Les paires stéréo sont utilisées pour détecter et reconstruire de nouveaux points lorsque leur nombre descend en dessous d'un certain seuil. Nous avons montré que le suivi de caméra proposé peut s'exécuter en temps réel sur un prototype d'équipement mobile doté d'une caméra stéréo, et qu'il fournit un bon niveau de stabilité et de robustesse, nécessaire pour une application de réalité augmentée.

Il y a plusieurs directions vers lesquelles nous nous dirigeons pour améliorer notre algorithme. Nous envisageons d'y ajouter un module de re-localisation pour permettre à l'algorithme de reconnaître un endroit qui a déjà été visité durant une session précédente. La re-localisation est également importante pour retrouver la position de la machine dans le cas où le suivi est perdu à cause d'occultations ou des mouvements amples et soudains de la caméra. Une autre direction de recherche en relation avec la première concerne un module d'optimisation, basé sur l'ajustement de faisceaux qui contribuera à améliorer le problème de fermeture de boucle en optimisant la structure 3D et en

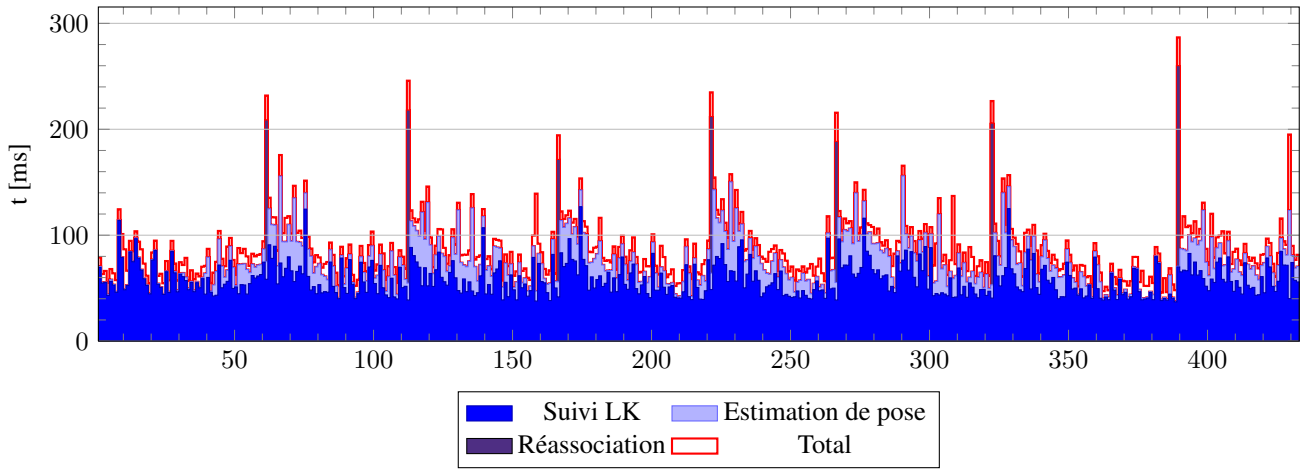
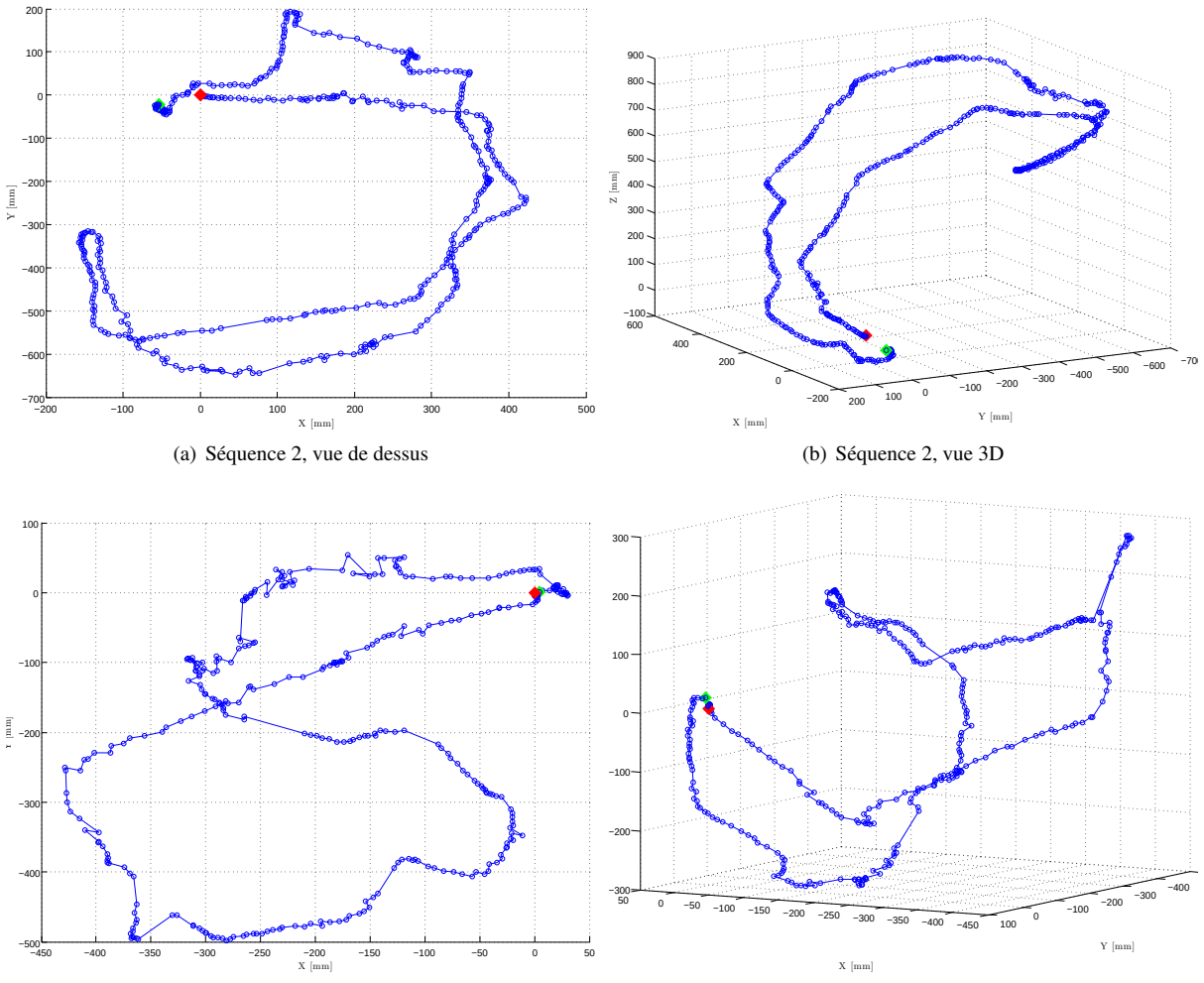



FIGURE 4 – Un exemple du temps d'exécution de notre algorithme, avec le temps d'exécution partiel des principales fonctions.




(a) Séquence 2, vue de dessus

(b) Séquence 2, vue 3D

(c) Séquence 4, vue de dessus

(d) Séquence 4, vue 3D

FIGURE 5 – Vue de dessus du plan $XY$ ((a) et (c)) et vue 3D ((b) et (d)) de la trajectoire des Séquences 2 et 4, avec le point de départ (en rouge) et le point d'arrivée (en vert).

générant une carte plus précise de la scène. Pour cela, nous investiguons une solution inspirée de [20], avec en tâche de fond un *thread* en charge de l'optimisation.
L'intégration et la fusion des données provenant des capteurs inertiels est une autre direction intéressante à investiguer, car elles peuvent améliorer la qualité du suivi, notamment lors de mouvements irréguliers et importants qui eux ne sont pas correctement pris en compte par les capteurs d'images. Dans un avenir proche, les codes de notre algorithmes seront disponibles pour la communauté.

# 5 Remerciements

Ce travail a été réalisé dans le cadre du projet Moov3D, soutenu par les pôles de compétitivité MINALOGIC et IMAGINOVE et financé par OSEO et la région Rhône-Alpes.

# Références

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak : Fast retina keypoint. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2012)*, pages 510–517, 2012. 3

[2] H. Bay, T. Tuytelaars, L. Van Gool, and L. Van Gool. Surf : Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proceedings of the European Conference Computer Vision (ECCV 2006)*, volume 3951, pages 404–417. Springer, 2006. 3

[3] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000. 4

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 5

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF : Binary Robust Independent Elementary Features. In *Proceedings of the European Conference on Computer Vision (ECCV 2010)*, September 2010. 3

[6] L. Calvet, P. Gurdjos, V. Charvillat, S. Gasparini, and P. Sturm. Suivi de caméra à partir de marqueurs plans composés de cercles concentriques : paradigme et algorithmes. In *ORASIS - Congrès des jeunes chercheurs en vision par ordinateur*, Praz-sur-Arly, France, 2011. INRIA Grenoble Rhône-Alpes. 2

[7] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic. Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, 51(1) :341–377, 2010. 1

[8] P. Corke, J. Lobo, and J. Dias. An introduction to inertial and visual sensing. *The International Journal of Robotics Research*, 26(6) :519–535, 2007. 1

[9] A. J. Davison, I. D. Reid, N. Molton, and O. Stasse. Monoslam : Real-time single camera slam. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(6) :1052–1067, 2007. 2

[10] M. Fiala. Artag fiducial marker system applied to vision based spacecraft docking. *Robot Vision for Space Applications*, page 35, 2005. 2

[11] M. A. Fischler and R. C. Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communication of ACM*, 24(6) :381–395, 1981. 4

[12] A. W. Fitzgibbon and A. Zisserman. Automatic camera tracking. In M. Shah and R. Kumar, editors, *Video Registration*, chapter 2, pages 18–35. Kluwer, 2003. 1

[13] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94(3) :335–360, 2011. 3

[14] J. Grunert. Das pothenotische problem in erweiterter gestalt nebst bber seine anwendungen in der geodäsie. *Grunerts Archiv für Mathematik und Physik*, pages 238–248, 1841. 4

[15] R. Haralick, C.-N. Lee, K. Ottenberg, and M. Noelle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal on Computer Vision*, 13(3) :331—-356, 1994. 4

[16] R. Hartley and A. Zisserman. *Multiple View Geometry*, volume 3. Cambridge University Press, 2003. 4

[17] J. Hol. Modeling and calibration of inertial and vision sensors. *International Journal of Robotics Research*, 29(2-3) :1–24, 2010. 2

[18] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion : Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1) :56, 2011. 2

[19] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007. 2, 3

[20] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proceedings of the Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*, Orlando, October 2009. 2, 3, 8

[21] D. W. F. V. Krevelen and R. Poelman. A survey of augmented reality technologies , applications and limitations. *International Journal of Virtual Reality*, 9(2) :1–20, 2010. 1

[22] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk : Binary robust invariant scalable keypoints. In A. Abe and R. Oehlmann, editors, *Proceedings of the International Conference on Computer Vision (ICCV 2011)*, pages 2548–2555. IEEE, 2011. 3

[23] F. Liarokapis, V. Brujic-Okretic, and S. Papakonstantinou. Exploring urban environments using virtual and augmented reality. *Journal of Virtual Reality and Broadcasting*, 3(5) :1–13, 2007. 1

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2) :91–110, 2004. 3

[25] C. Mei, G. Sibley, M. Cummins, P. M. Newman, and I. D. Reid. A constant-time efficient stereo slam system. In *Proceedings of the British Machine Vision Conference (BMVC 2009)*, 2009. 2

[26] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8) :1178–1193, July 2009. 2, 3

[27] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Application (VISSAPP'09)*, pages 331–340. INSTICC Press, 2009. 4

[28] R. A. Newcombe, S. Lovegrove, and A. J. Davison. Dtam : Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV 2011)*, pages 2320–2327, 2011. 2

[29] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23 :2006, 2006. 2

[30] C. Pheatt. Intel® threading building blocks. *J. Comput. Sci. Coll.*, 23(4) :298–298, Apr. 2008. 5

[31] E. Rosten, R. Porter, and T. Drummond. Faster and better : a machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1) :105–119, 2010. 3

[32] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski. Orb : An efficient alternative to sift or surf. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV 2011)*, pages 2564–2571, 2011. 3

[33] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual slam : Why filter ? *Image and Vision Computing*, 30(2) :65–77, 2012. 2

[34] P. F. Sturm and S. J. Maybank. On plane-based camera calibration : A general algorithm, singularities, applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99)*, pages 1432–1437, 1999. 3

[35] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms : Theory and Practice*, ICCV '99, pages 298–372, 2000. 2

[36] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors : A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3) :177–280, 2007. 3

[37] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, ISWC '03, pages 127–, Washington, DC, USA, 2003. IEEE Computer Society. 2

[38] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Virtual Reality, 1999.*, pages 260 – 267, 1999. 2