Localisation par vision monoculaire pour la navigation autonome : précision et stabilité de la méthode

Eric Royer, Maxime Lhuillier, Michel Dhome, Jean-Marc Lavest

Localisation par vision monoculaire pour la navigation autonome : précision et stabilité de la méthode

Localization with monocular vision for autonomous navigation : accuracy and stability of the method

Eric Royer Maxime Lhuillier Michel Dhome Jean-Marc Lavest

LASMEA, UMR 6602 CNRS et université Blaise Pascal
24 Avenue des Landais, 63177 Aubière, FRANCE
Eric.ROYER@lasmea.univ-bpclermont.fr
http ://wwwlasmea.univ-bpclermont.fr/Personnel/Eric.Royer/

Résumé

Nous présentons une méthode pour déterminer la localisation d'un robot mobile par rapport à une séquence vidéo d'apprentissage. Dans un premier temps, le robot est conduit manuellement sur une trajectoire et une séquence vidéo de référence est enregistrée par une seule caméra. Puis un calcul hors ligne nous donne une reconstruction 3D du chemin suivi et de l'environnement. Cette reconstruction est ensuite utilisée pour calculer la pose du robot en temps réel dans une phase de navigation autonome. Les résultats obtenus sont comparés à la vérité terrain mesurée par un GPS différentiel ou une plate-forme rotative graduée.

Mots Clef

Localisation, temps réel, robot mobile, reconstruction 3D.

Abstract

We present a method for computing the localization of a mobile robot with reference to a learning video sequence. The robot is first guided on a path by a human, while the camera records a monocular learning sequence. Then a 3D reconstruction of the path and the environment is computed off line from the learning sequence. The 3D reconstruction is then used for computing the pose of the robot in real time in autonomous navigation. Results from our method are compared to the ground truth measured with a differential GPS or a rotating platform.

Keywords

Localization, real-time, mobile robot, 3D reconstruction.

1 Introduction

Nous traitons du problème de la localisation en temps réel dans un environnement urbain. Notre but est de développer un système robotique capable de naviguer de façon autonome sur de grandes distances (de plusieurs centaines de mètres à quelques kilomètres). Dans un premier temps le robot est guidé manuellement et il enregistre une séquence de référence avec un système vidéo monoculaire. Ensuite le robot doit être capable de naviguer de façon autonome sur une trajectoire donnée au voisinage de la trajectoire d'apprentissage. Notre méthode consiste à construire un modèle tridimensionnel de l'environnement du robot en utilisant seulement les données enregistrées par la caméra pendant la phase d'apprentissage. Ensuite, lorsque le robot est à proximité de la trajectoire apprise, il est possible d'utiliser l'image fournie par la caméra pour établir des correspondances entre modèle 3D reconstruit et primitives 2D vues dans l'image. Ceci permet par un calcul de pose robuste de localiser le robot en temps réel. Ce processus est illustré sur la figure 1. Aucune de ces opérations ne nécessite une intervention manuelle et le seul capteur utilisé est une caméra vidéo calibrée. Ce système de localisation a été utilisé avec succès pour la navigation autonome d'un véhicule électrique [18].

Dans un grand nombre d'applications en robotique mobile, la localisation du véhicule est fournie par un récepteur GPS (Global Positionning System). Un GPS différentiel a une précision de quelques centimètres s'il y a suffisamment de satellites visibles. Cette précision est suffisante pour piloter un robot. Malheureusement, le GPS fonctionne mal en milieu urbain car les bâtiments peuvent masquer les satellites. De plus les signaux GPS peuvent être réfléchis par les façades et induire une mauvaise position. Le cas le plus défavorable se produit lorsque le robot est dans une rue étroite (canyon urbain). Dans ce cas, tous les satellites visibles sont dans un même plan vertical ce qui est une très mauvaise situation pour calculer une position par triangulation. D'un autre côté, ce type d'environnement fournit beaucoup d'informations utilisables par un système de vision de par la présence d'un grand nombre d'objets à proxi-

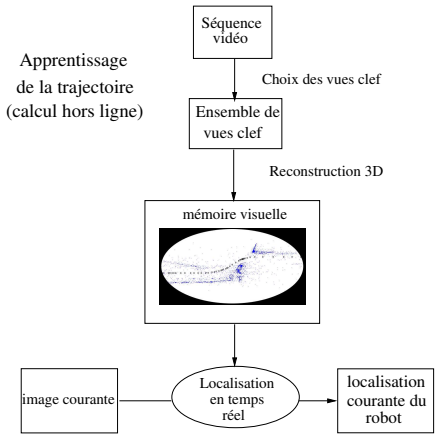

FIG. 1 – Un aperçu de notre système de localisation

mité immédiate. Notre but est de faire un système adapté au milieu urbain qui soit capable de fournir une position à une précision du même ordre qu'un GPS différentiel. A ce titre, à la fin de cet article les résultats obtenus par notre système sont comparés avec ceux obtenus par un GPS différentiel.

Dès qu'on dispose d'une modélisation 3D de l'environnement, il est possible de calculer la position de la caméra par rapport à cette reconstruction. Plusieurs approches de construction de carte sont possibles. Le SLAM (Simultaneous Localization And Mapping) est très séduisant car la localisation est possible dès que le système est mis en marche. Mais la cartographie est quelque chose de complexe, aussi faire du SLAM en temps réel à partir de la vision monoculaire est difficile. Cela a déjà été fait par Davison [5], mais uniquement pour un environnement de taille réduite dans lequel on utilise moins d'une centaine d'amers. Il est également possible de calculer le mouvement de la caméra à partir des données vidéo en faisant de l'odométrie visuelle comme le propose Nistér *et al.* [15]. Cela rend inutile la construction d'une carte, mais l'accumulation des erreurs au cours du temps fait que la précision de la localisation calculée décroît avec la distance parcourue. Une autre approche possible consiste à commencer par construire une carte hors ligne et utiliser ensuite cette carte pour la localisation en ligne. Le principal avantage est que la contrainte de temps réel ne s'applique pas à la construction de la carte et des algorithmes plus précis peuvent être employés. Vacchetti *et al.* [20] utilisent un modèle CAO fourni par l'utilisateur pour obtenir un système de localisation très performant. Cobzas *et al.* [4] utilisent une caméra montée sur une plate-forme rotative pour construire un ensemble d'images panoramiques dans lesquelles on ajoute une information 3D obtenue avec un télémètre laser. Après la construction de ce modèle 3D, une seule image est suffisante pour localiser la caméra. Kidono *et al.* [11] utilisent également une étape de reconstruction 3D avant de pouvoir se localiser au voisinage de cette trajectoire. Dans leurs travaux, ils supposent que le sol est plan et utilisent un système de vision stéréo couplé à un odomètre pour construire la carte. Notre système fonctionne sur le même principe mais nos hypothèses de travail sont moins restrictives car nous ne faisons pas l'hypothèse d'un sol plan et nous travaillons avec une seule caméra calibrée.

Pour la navigation autonome d'un robot le long d'une trajectoire apprise, d'autres solutions n'utilisant pas la construction d'une carte ont été proposées. Elles ne donnent pas la pose du robot à chaque image, mais elles évitent l'étape de construction de carte qui peut être longue. Cela a été présentée par Matsumoto *et al.* [13]. Plus récemment, les travaux de Remazeilles *et al.* [17] et de Blanc *et al.* [3] montrent comment il est possible de faire naviguer un robot par asservissement visuel entre des images clef.

Dans la section 2, nous présentons la méthode utilisée pour construire le modèle de l'environnement à partir des images enregistrées pendant la séquence de référence. Puis dans la section 3, la méthode utilisée pour calculer la localisation en temps réel est détaillée. Finalement, nous donnons les résultats obtenus dans la section 4. Ces résultats sont comparés avec la vérité terrain obtenue grâce à un GPS différentiel pour évaluer la précision de la position calculée par vision en environnement extérieur. Une comparaison entre l'orientation calculée par vision et des mesures effectuées sur une plate-forme rotative viennent compléter ces résultats.

2 Reconstruction 3D de la séquence de référence

2.1 Présentation

Le but de la reconstruction est d'obtenir la pose d'un sous-ensemble des caméras de la séquence de référence ainsi qu'un ensemble d'amers visuels avec leurs positions 3D. Tout ceci est exprimé dans un repère global. Pour la reconstruction, nous utilisons une séquence obtenue avec une seule caméra calibrée. Dans nos expériences, la caméra a été calibrée avec une mire plane selon la méthode proposée par Lavest *et al.* [12]. Calibrer la caméra est important parce que l'objectif grand angle que nous avons utilisé présentait une forte distorsion radiale. Si on connaît les paramètres intrinsèques et les coefficients de distorsion alors le calcul du mouvement de la caméra est à la fois plus robuste et plus précis. De plus les algorithmes utilisant une caméra calibrée fonctionnent même si la scène est plane dans un certain nombre d'images, ce qui n'est pas le cas si les données de calibrage sont inconnues (par exemple [2], [16]).

Chaque étape de la reconstruction ainsi que la localisation reposent sur la mise en correspondance d'images. Ceci est fait en détectant des points d'intérêt de Harris [9] dans chaque image. Pour avoir des points répartis sur toute l'image, on divise l'image en 64 cases et on conserve les 500 meilleurs points sur l'ensemble de l'image ainsi qu'un

minimum de 20 points pour chaque case. Pour chaque point d'intérêt dans l'image 1, on sélectionne des points correspondants possibles dans une région de recherche de l'image 2. Pour chaque appariement possible, on calcule un score de corrélation centré et normé. Puis on retient les couples avec le meilleur score. Faire la mise en correspondance de cette façon peut paraître coûteuse en temps de calcul, mais on peut programmer un détecteur de coins de Harris très efficace en utilisant les extensions SIMD des processeurs récents.
Dans la première étape de la reconstruction, on extrait un ensemble d'images clef à partir de la séquence de référence. Ensuite on calcule la géométrie épipolaire et le mouvement de la caméra entre les images clef. Les points d'intérêt utilisés pour calculer la géométrie épipolaire sont reconstruits en 3D. Ces points serviront d'amers pour la localisation. Ils sont stockés avec une imagette qui représente le voisinage du point dans l'image clef où il a été détecté. Ainsi il est possible de les mettre en correspondance avec les points qui seront détectés dans de nouvelles images.

2.2 Sélection des images clef

Si le mouvement de la caméra entre deux images clef est trop faible, alors le calcul de la géométrie épipolaire est mal conditionné. Il faut donc faire en sorte que le déplacement de la caméra entre deux images clef soit le plus grand possible tout en étant toujours capable de faire une mise en correspondance entre les images. La première image de la séquence est toujours choisie comme image clef, elle est notée I_1. La deuxième image clef I_2 est choisie la plus éloignée possible de I_1 dans le flux vidéo en respectant la contrainte qu'il y ait au moins M points d'intérêt communs entre I_1 et I_2. Une fois que les images clef I_1 à I_n sont choisies ($n > 1$), on choisit I_{n+1} pour qu'il y ait au moins M points d'intérêt communs entre I_{n+1} et I_n et au moins N points d'intérêt communs entre I_{n+1} et I_{n-1}. Cela nous assure qu'il y a suffisamment de points en correspondance entre deux images clef pour calculer le mouvement de la caméra. Dans nos expériences nous détectons environ 1500 points d'intérêt par image et les seuils sont fixés à $M = 400$ et $N = 300$.

2.3 Calcul du déplacement de la caméra

A ce stade nous avons un sous-ensemble des images de la séquence vidéo enregistrée pendant la phase d'apprentissage. L'algorithme utilisé pour calculer la reconstruction 3D de la scène et de la trajectoire de la caméra peut être divisé en trois parties. Avec les trois premières images clef, on calcule le déplacement de la caméra en utilisant un calcul de matrice essentielle. Ensuite pour chaque triplet d'images, le déplacement est obtenu par un calcul de pose. Ces calculs produisent une solution initiale de la reconstruction qui est optimisée dans un ajustement de faisceaux hiérarchique.
Pour le premier triplet d'images, le calcul est réalisé par la méthode proposée par Nistér [14] pour trois images. Pour résumer, on calcule la matrice essentielle qui lie la première et la dernière image du triplet en utilisant un échantillon de 5 points en correspondance. La contrainte épipolaire pour chacun des 5 points s'écrit $q'^{i^T} E q^i = 0, \forall i \in \{1..5\}$. Il existe une sixième relation qui doit être vérifiée par toute matrice essentielle : $EE^T E - \frac{1}{2} trace(EE^T)E = 0$. En combinant ces relations on aboutit à une équation polynômiale de degré 10. Il y a donc au plus 10 solutions pour E et chaque matrice essentielle donne 4 solutions possibles pour (R, T). Les solutions pour lesquelles au moins un des 5 points est reconstruit derrière une des caméras sont éliminées. Ensuite on calcule la pose de la deuxième caméra du triplet en utilisant 3 des 5 points de l'échantillon. Ce calcul est fait de manière robuste en utilisant l'approche RANSAC [7]. Chaque tirage aléatoire de 5 points produit un certain nombre d'hypothèses pour les trois caméras. La meilleure est choisie en calculant l'erreur de reprojection de tous les points dans les trois images. On conserve la solution qui donne le plus grand nombre d'appariements cohérents.

Pour le calcul de la deuxième caméra ainsi que pour les calculs de pose utilisés dans les autres parties de cet article, on a besoin d'un moyen de calculer la pose d'une caméra connaissant un ensemble de points 3D ainsi que leurs positions 2D dans les images et les paramètres intrinsèques de la caméra. Un état de l'art de ces algorithmes figure dans un article d'Haralick *et al.* [8]. L'algorithme de Grunert est utilisé tel qu'il est décrit dans [8]. Cette méthode fait appel à des calculs trigonométriques dans le tétrahèdre formé par 3 points 3D et le centre optique de la caméra. Le calcul aboutit à la résolution d'une équation du quatrième degré. Il y a donc au plus 4 solutions pour chaque échantillon de 3 points. La solution est choisie dans le processus de RANSAC.

Pour les triplets suivants, on utilise une méthode différente pour calculer le déplacement de la caméra. Supposons qu'on connaisse la pose des caméras C_1 à C_N. Il est possible de calculer la pose de C_{N+1} en utilisant les poses de C_{N-1} et C_N et des correspondances de points dans le triplet d'images $(N-1, N, N+1)$. On apparie un ensemble de points P^i dont les projections sont connues dans chaque image du triplet. A partir des projections dans les images $N-1$ et N, on peut retrouver la position 3D de P^i. Puis en utilisant les coordonnées 3D des P^i et leurs projections dans l'image $N+1$, on utilise le calcul de pose de Grunert pour obtenir la pose de la caméra C_{N+1}. Pendant ce temps la position 3D des points est enregistrée car ce sont ces points qui serviront d'amers pour la localisation. L'intérêt du calcul de pose itératif tel qu'il est décrit est qu'on peut traiter des images dans lesquelles tous les points observés sont sur un même plan. Après le calcul de pose une deuxième étape de mise en correspondance est faite en tenant compte de la contrainte épipolaire qui vient d'être calculée. Cette deuxième étape d'appariement permet d'augmenter le nombre de points correctement reconstruits d'environ 20 %. C'est particulièrement important, parce que nous avons besoin de nombreux points pour

calculer la pose de la caméra suivante.

2.4 Ajustement de faisceaux hiérarchique

Le calcul du mouvement de la caméra décrit précédemment ne donne pas une très bonne solution. De plus, le calcul de la pose de la caméra C_N dépend des résultats obtenus pour les caméras précédentes, et les erreurs de calcul peuvent s'accumuler tout au long de la séquence. Pour limiter grandement ce problème, on utilise un ajustement de faisceaux pour affiner la solution obtenue. L'ajustement de faisceaux est un processus de minimisation basé sur l'algorithme de Levenberg-Marquardt. La fonction de coût est $f(C_E^1, \cdots, C_E^N, P^1, \cdots, P^M)$ où C_E^i désigne les paramètres extrinsèques de la caméra i, et P^j désigne les coordonnées 3D du point j. La fonction de coût est la somme des erreurs de reprojection de tous les points considérés comme corrects dans toutes les images :

$$f(C_E^1, \cdots, C_E^N, P^1, \cdots, P^M) = \sum_{1 \leq i \leq N} \sum_{1 \leq j \leq M, j \in J_i} d^2(p_i^j, K_i P^j)$$

où $d^2(p_i^j, K_i P^j)$ est le carré de la distance euclidienne entre $K_i P^j$ projection du point P^j par la caméra i, et p_i^j est le point d'intérêt correspondant ; K_i est la matrice de projection 3×4 construite d'après les valeurs contenues dans C_E^i et les paramètres intrinsèques connus de la caméra ; J_i est l'ensemble des points dont l'erreur de reprojection dans l'image i est inférieure à 2 pixels au début de la minimisation. Après quelques itérations, on recalcule J_i et on refait une série d'itérations. On continue à recalculer J_i tant que le nombre de points correctement appariés augmente.

Ce ne serait pas une bonne idée de commencer par calculer toutes les positions de caméras et utiliser l'ajustement de faisceaux une seule fois sur la totalité de la séquence. Dans ce cas, l'accumulation des erreurs pourrait donner une solution initiale trop éloignée de la vraie solution pour que l'ajustement de faisceaux converge. Il faut utiliser l'ajustement tout au long de la reconstruction. Faire un ajustement après chaque nouvelle image est bien trop coûteux en temps de calcul. Une solution plus rapide décrite dans [10] consiste à faire un ajustement hiérarchique. Une longue séquence est divisée en deux parties avec un recouvrement de deux images pour pouvoir les fusionner ensuite. Chaque sous-séquence est elle-même divisée en deux et ceci de façon récursive jusqu'à ce que chaque sous-séquence contienne seulement trois images. Chaque triplet est traité comme indiqué au paragraphe 2.3. Pour le premier triplet on utilise un calcul de matrice essentielle. Pour chaque triplet suivant, les deux première images sont communes avec le triplet précédent. Donc deux des caméras sont déduites directement du triplet précédent et la dernière est obtenue par un calcul de pose. Une fois qu'une solution initiale a été calculée sur un triplet, cette solution est optimisée dans un ajustement de faisceaux sur trois images.

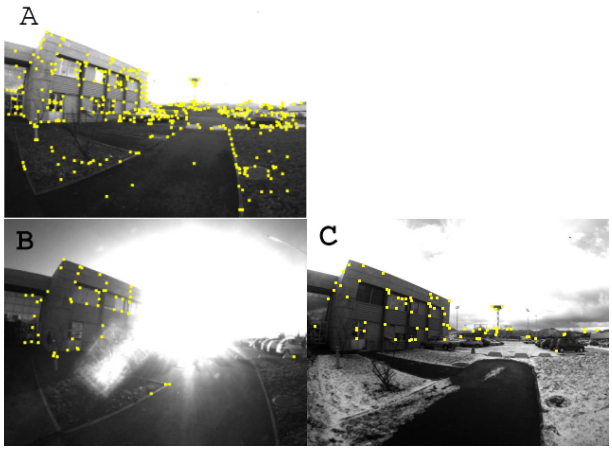

FIG. 2 – Appariement de points entre une image du flux vidéo (B et C) et l'image clef la plus proche (A). En B, soleil dans le champ de la caméra, en C, sol enneigé. Seuls les appariements corrects sont affichés.

Pour fusionner deux séquences S^1 et S^2, on utilise les 2 dernières caméras S_{N-1}^1 et S_N^1 de S^1 et les deux premières caméras S_1^2 et S_2^2 de S^2. Comme les images sont les mêmes, les caméras associées après la fusion doivent être les mêmes. Pour cela, on applique une rotation et une translation à S^2 pour que S_N^1 et S_2^2 aient la même position et orientation. Ensuite un facteur d'échelle est appliqué pour que $d(S_{N-1}^1, S_N^1) = d(S_1^2, S_2^2)$, où $d(S_n^i, S_m^j)$ est la distance euclidienne entre les centres optiques des caméras associées à S_n^i et S_m^j. Cela n'assure pas que S_{N-1}^1 et S_1^2 sont les mêmes, mais un ajustement de faisceaux est utilisé sur le résultat de la fusion pour que tout rentre dans l'ordre. On fusionne ainsi jusqu'à ce que la séquence complète soit reconstruite. La reconstruction se termine avec un ajustement de faisceaux global. Dans cette dernière étape on traite plusieurs dizaines de milliers de points.

3 Localisation en temps réel

Le résultat de la phase d'apprentissage est une reconstruction 3D de la scène : on a la pose de la caméra pour chaque image clef et un ensemble de points 3D associés avec leur position 2D dans les images. Au début du processus de localisation, on ne sait pas où le robot se trouve. Il faut donc comparer l'image courante avec toutes les images clef pour trouver la plus proche. Ceci est fait en mettant en correspondance des points d'intérêt entre image courante et image clef et en calculant une pose par RANSAC pour ne garder que les appariements cohérents. La pose obtenue avec le plus grand nombre d'appariements cohérents est une bonne estimation de la position initiale de la caméra. Cette étape demande quelques secondes mais elle n'est faite qu'une fois au tout début.

Après cela, on dispose en permanence de la pose approximative de la caméra. Il suffit de mettre à jour cette pose, ce

qui est beaucoup plus rapide. Voici la méthode utilisée pour cela. On commence par faire une prédiction de la pose. Dans notre cas, nous reprenons simplement la pose calculée à l'image précédente mais on pourrait intégrer ici une mesure d'odométrie par exemple. Cette prédiction nous permet de choisir l'image clef la plus proche et donc de sélectionner dans la base de données un ensemble de points d'intérêt potentiellement visibles avec leurs coordonnées 3D. Puis on détecte les points d'intérêt dans l'image courante et on fait un appariement avec l'image clef choisie. Cet appariement est rapide car les zones de recherche sont réduites (30x20 pixels) grâce à la prédiction de la pose courante qui nous donne également une prédiction sur la position où les points peuvent être retrouvés dans l'image. A ce stade, on dispose d'un ensemble d'appariements entre points 2D de l'image courante et points 3D de la base d'apprentissage. On fait un calcul de pose robuste avec RANSAC pour calculer la pose de la caméra courante. Ensuite la pose est affinée en utilisant la méthode itérative proposée par Araújo *et al.* [1] avec quelques modifications pour la rendre robuste aux faux appariements. Cet algorithme est une minimisation de l'erreur de reprojection pour tous les points en utilisant la méthode de Newton. A chaque itération, on résout le système linéaire $J\delta = e$ pour calculer un vecteur de corrections δ qui doit être soustrait des paramètres de la pose, où e est le terme d'erreur formé par les erreurs de reprojection de chaque point en x et en y, et où J est la matrice jacobienne de l'erreur. Araújo donne une façon de calculer explicitement J. Dans notre implémentation, les points utilisés dans la minimisation sont recalculés à chaque itération. On ne garde que ceux dont l'erreur de reprojection est inférieure à 2 pixels. En général, moins de 5 itérations sont nécessaires. La figure 2 montre les appariements obtenus entre une image clef et des images prises pendant la localisation. Comme on le voit sur ces images, cette méthode de localisation est robuste à des occultations importantes grâce à l'utilisation de la structure 3D de la scène qui permet d'imposer des contraintes fortes sur la position des points lors de l'appariement.

4 Résultats

4.1 Expériences réalisées

Nous avons mené plusieurs expériences pour évaluer la précision de nos algorithmes de reconstruction et de localisation. En environnement extérieur, nous avons utilisé un récepteur GPS différentiel pour enregistrer la position du véhicule. Ces données représentent la vérité terrain. L'enregistrement de données GPS avec une précision centimétrique n'est pas possible partout. C'est pourquoi nous avons choisi un endroit où il n'y avait pas beaucoup d'immeubles pour ne pas masquer les satellites. C'est un cas défavorable pour nos algorithmes de vision parce qu'ils sont plutôt conçus pour des zones urbaines denses où l'environnement est riche en informations visuelles. Pour compléter ces mesures dans des lieux où le GPS ne pouvait pas être utilisé, nous avons enregistré des trajectoires en boucle, ce qui permet de mesurer la dérive totale au bout de la reconstruction. Pour mesurer la précision de l'orientation donnée par l'algorithme de localisation, nous avons fait des mesures sur une platine orientable en intérieur. Enfin, la robustesse du processus de localisation a été testée dans différentes conditions de luminosité. La figure 2 montre deux cas difficiles où la navigation autonome d'un robot a été possible en utilisant l'algorithme de localisation décrit dans cet article. La robustesse du système repose essentiellement sur deux points : un objectif fish-eye et l'utilisation d'un grand nombre de points répartis sur la totalité de l'image. Le grand angle de vue permet d'être moins sensible aux occultations provoquées par un piéton où un véhicule par exemple. Le grand nombre de points permet d'assurer qu'on pourra en retrouver au moins quelques uns dans la majorité des situations. Mais il faut bien voir que chaque occultation, modification de l'environnement, ou zone de l'image où le capteur est saturé fait perdre un certain nombre de points, ce qui peut conduire à une impossibilité de calculer la pose.

FIG. 3 – Quelques images extraites de $campus_1$

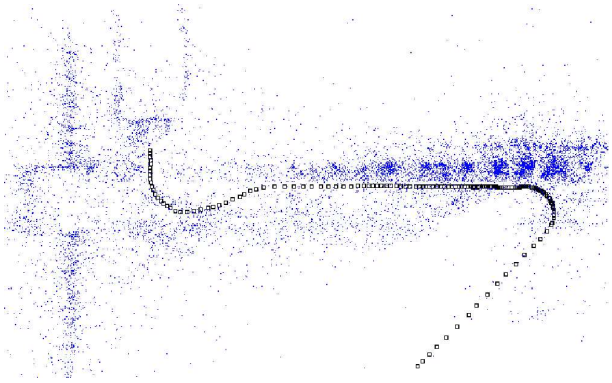

FIG. 4 – Vue de dessus de la reconstruction 3D de $campus_1$

4.2 Précision sur la position

Comparaison avec le GPS. Comparer des positions obtenues avec notre algorithme à celles fournies par le GPS n'est pas trivial. Deux opérations sont nécessaires pour comparer les deux jeux de données. Tout d'abord, l'antenne du récepteur GPS n'occupe pas la même position que la caméra sur le véhicule. L'antenne GPS est placée à l'aplomb du milieu de l'essieu arrière, alors que la caméra est à l'avant (la position de chaque capteur sur le véhicule a été mesurée manuellement). Les deux capteurs n'ont donc pas exactement la même trajectoire. A partir des positions GPS, on calcule les informations que donnerait un GPS virtuel qui serait placé à la même position que la caméra. D'autre part, la reconstruction 3D est faite dans un repère euclidien arbitraire lié aux caméras, alors que les positions GPS sont données dans un autre système de coordonnées liées au sol. La reconstruction 3D obtenue doit subir une transformation rigide (rotation, translation et mise à l'échelle) pour être exprimée dans le même repère que le GPS. L'approche décrite par Faugeras *et al.* [6] est utilisée pour calculer cette transformation.
Une fois ces corrections effectuées, on peut calculer pour chaque caméra une erreur de position en mètres. Pour cela, on suppose que les données GPS sont exactes. Le récepteur GPS utilisé est un Real Time Kinematics Differential GPS (modèle Thalès Sagitta). Selon le constructeur, il a une précision de 1 *cm* dans le plan horizontal. La précision sur un axe vertical n'est que de 20 *cm* sur notre plate-forme. Aussi, le calcul d'erreur de position n'est fait que dans un plan horizontal.
Nous avons enregistré quatre séquences vidéo le même jour, approximativement sur la même trajectoire (avec un écart latéral entre deux trajectoires de 1 *m* au maximum). Nous avons calculé une reconstruction 3D pour chaque séquence. Pour évaluer notre algorithme nous avons utilisé tour à tour chacune de ces séquences comme séquence de référence et nous avons lancé l'algorithme de localisation sur les autres. La longueur de la trajectoire est d'environ 80 *m*. On peut voir quelques images extraites des vidéos sur la figure 3. La figure 4 montre la reconstruction obtenue à partir de la vidéo $campus_1$ en vue de dessus. Les carrés noirs représentent la position des images clef, et les points 3D reconstruits apparaissent sous forme de points. Quelques piétons circulaient pendant l'enregistrement des images, mais cela n'a pas perturbé le fonctionnement des algorithmes.

Précision de la reconstruction. Après avoir calculé la reconstruction à partir des images clef, on utilise l'algorithme de localisation pour obtenir la pose de chaque caméra dans la séquence (pas seulement les images clef). Puis on calcule l'erreur de position par rapport au GPS obtenue pour chaque caméra. La moyenne de ces erreurs sur toute la séquence figure dans le tableau 1. On indique aussi le nombre d'images clef et le nombre de points utilisés dans la séquence. La figure 5 montre les positions calculées pour chaque image clef (petits cercles) comparées à la trajectoire enregistrée par le GPS (ligne continue). L'erreur de reconstruction est en majeure partie due à une lente dérive du processus de reconstruction. Elle augmente avec la longueur et la complexité (plus grand nombre de virages) de la trajectoire.

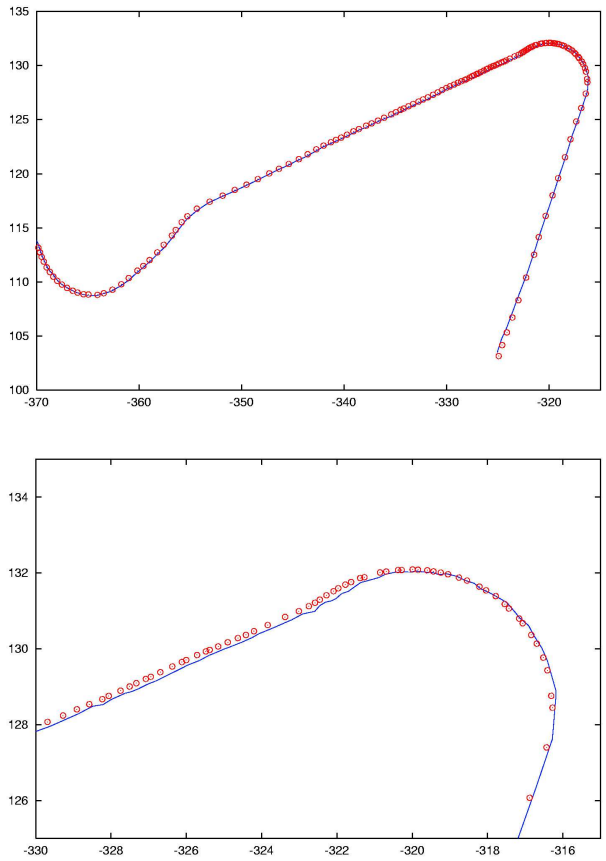

FIG. 5 – Position des images clef (cercles rouges) superposées à la trajectoire enregistrée par le GPS (ligne continue bleue), unités en mètres. La trajectoire complète est en haut et un agrandissement apparaît en bas

Précision de la localisation. Nous avons calculé une localisation pour chaque image de $campus_i$ en utilisant $campus_j$ comme séquence de référence pour tout i et j tels que $i \neq j$ (cela fait 12 expériences de localisation au total). La mesure de l'erreur de localisation est plus complexe que celle de l'erreur de reconstruction. Si on se contente de calculer l'écart entre la position donnée par l'algorithme de vision et celle donnée par le GPS, on calcule une erreur de localisation globale qui intègre à la fois les erreurs commises à la localisation mais aussi à la reconstruction. Et finalement, on retrouve approximativement la même erreur que celle de la reconstruction.
Mais heureusement, dans un grand nombre d'applications, une position globale n'est pas nécessaire. C'est le cas en particulier si on veut faire naviguer un robot. Dans ce cas nous avons juste besoin de calculer la distance entre la position courante du robot et sa position lors de l'apprentissage

Séquence	erreur de position moyenne	nombre d'images clef	nombre de points 3D
$campus_1$	25 cm	113	14847
$campus_2$	40 cm	121	15689
$campus_3$	34 cm	119	15780
$campus_4$	24 cm	114	14802

TAB. 1 – Erreur de reconstruction moyenne pour une trajectoire de 80m

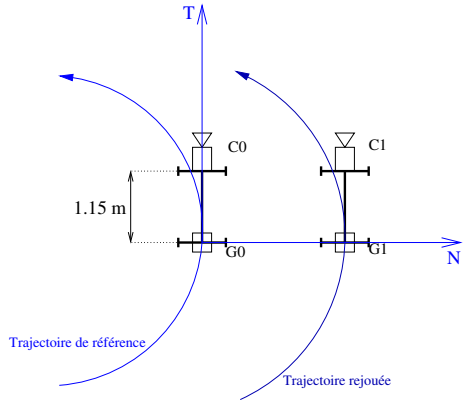

FIG. 6 – Calcul de l'écart latéral par rapport à la trajectoire de référence

de la trajectoire, ainsi que l'écart angulaire entre ces positions. Cela veut dire qu'il suffit d'une position relative par rapport à la trajectoire de référence. On peut définir l'erreur de localisation de manière à mesurer l'erreur commise sur cette position relative. La définition de cette erreur est un peu plus compliquée que pour l'erreur de reconstruction.

On commence par calculer l'écart latéral entre la position courante du robot et la position la plus proche sur la trajectoire de référence. Ceci est illustré par la figure 6. La position du robot est toujours définie par la position du point situé au milieu de l'essieu arrière. Cette position est celle qui est donnée par le récepteur GPS. Quand on travaille avec la vision, il faut calculer la position de ce point du robot à partir de la position obtenue par vision. Ceci est fait après avoir appliqué un changement d'échelle global à la reconstruction 3D, pour que l'échelle soit la même pour les données vision et GPS. A partir de la position C_1 (localisation par vision de la caméra courante), on calcule G_1 (position correspondante du robot). C'est fait simplement en mesurant la position de la caméra sur le toit du véhicule. Ensuite on cherche la position G_0 du robot la plus proche de G_1 sur la trajectoire de référence. En G_0, on calcule la tangente $\overrightarrow{T}$ et la normale unitaire $\overrightarrow{N}$ à la trajectoire. L'écart latéral calculé par vision est $\delta_V = \overrightarrow{G_0G_1} \cdot \overrightarrow{N}$. L'écart latéral est aussi calculé à partir des données GPS et on obtient δ_G (dans ce cas on dispose directement de G_1 et G_0). δ_G et δ_V sont deux mesures de la même distance physique obtenue à partir de deux capteurs différents. Finalement, l'erreur de

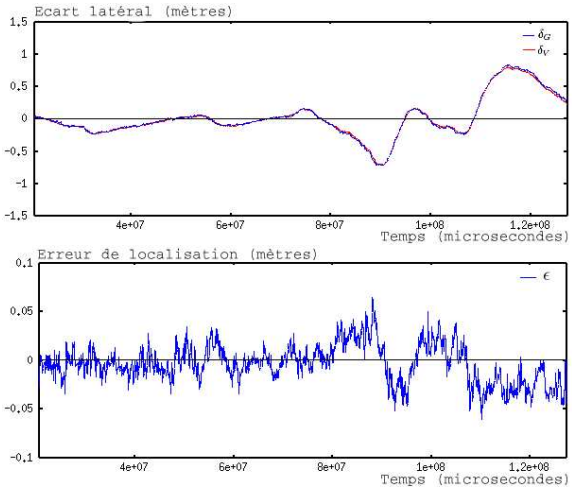

FIG. 7 – Erreur latérale (en haut) mesurée avec le GPS δ_G (bleu) ou par vision δ_V (rouge) et erreur de localisation ϵ (bas)

localisation est définie par $\epsilon = \delta_V - \delta_G$. De là, on peut calculer l'écart type de ϵ pour la totalité de la trajectoire : on appelle ceci l'erreur de localisation moyenne.

Nous avons calculé l'erreur de localisation moyenne pour chacune des 12 expériences : l'erreur varie de 1,4 *cm* à 2,2 *cm*, avec une moyenne de 1,9 *cm*. Il faut noter que l'erreur obtenue ici est du même ordre de grandeur que la précision donnée pour le récepteur GPS par la fabricant (1 *cm*). Cette erreur intègre donc pour partie le bruit de mesure du récepteur GPS. La figure 7 montre l'écart latéral et l'erreur de localisation calculés pour une des expériences dont l'erreur de localisation moyenne était de 1,9 *cm*. Pour s'assurer que cette façon de mesurer l'erreur de localisation est valide, nous avons couplé l'algorithme de localisation avec une loi de commande pour contrôler le déplacement du véhicule. Nous avons utilisé tour à tour le récepteur GPS et l'algorithme de vision pour contrôler le véhicule grâce à la loi de commande détaillée dans [19]. Les deux capteurs ont donné une précision équivalente pour la trajectoire rejouée (4 *cm* en ligne droite et moins de 35 *cm* d'écart latéral dans les courbes). Cela confirme que la localisation fournie par la vision ou par le GPS sont équivalents pour la navigation autonome du robot. L'erreur est due plus à la difficulté de contrôler le véhicule qu'à la partie localisation.

Trajectoires en boucle et cas des longues séquences. L'algorithme de reconstruction nous a permis de traiter de longues séquences. Par exemple la figure 13 montre la reconstruction obtenue sur une trajectoire mesurant approximativement 400 *m*. La séquence comporte 333 images clef et 22468 points ont été reconstruits. Quelques images extraites de cette séquence sont visibles sur la figure 14. Cette trajectoire constitue un aller-retour et le point d'arrivée devrait être confondu avec le point de départ. La figure permet de bien visualiser la dérive du processus de reconstruc-

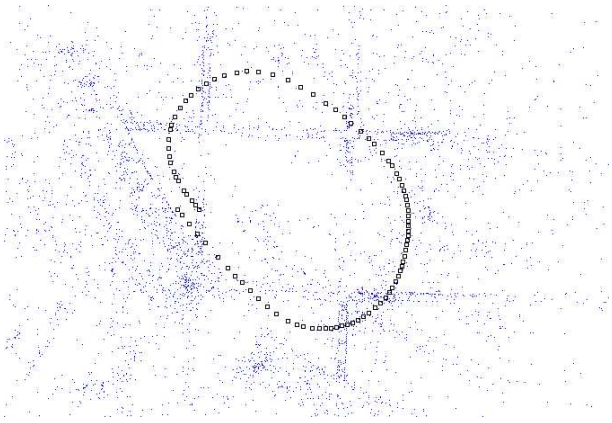

FIG. 8 – Reconstruction 3D d'une séquence en boucle

FIG. 9 – Quelques images extraites de la séquence en boucle

tion. L'erreur commise en fin de séquence peut paraître importante mais cela ne gêne en rien la navigation autonome du robot. Cela peut être mis en évidence en considérant une trajectoire qui boucle sur elle-même : la reconstruction associée à une telle trajectoire apparaît en figure 8 et quelques images extraites de la séquence vidéo sont visibles sur la figure 9. Là encore la position de la dernière caméra devrait coïncider avec la première, mais à cause de l'erreur de reconstruction, la boucle ne se referme pas exactement. Malgré tout, il a été possible de faire naviguer un robot sur cette boucle sans le moindre à-coup visible dans sa trajectoire. En effet, la loi de commande utilisée [19] nécessite en entrée l'écart latéral par rapport à la trajectoire de référence ainsi que l'écart angulaire par rapport à cette trajectoire. Or la localisation est faite par rapport au nuage de points visibles à un moment donné par le robot (les points qui sont cohérents par rapport à une seule image clef). Lorsqu'on passe de la fin d'une boucle au début d'une nouvelle, cela correspond à un changement de repère global qui concerne la trajectoire de référence, la position courante et le nuage de points visibles, et dans ce cas l'écart latéral à la trajectoire de référence ne présente pas de discontinuité. Cette propriété est très intéressante parce que cela permet d'envisager des parcours de plusieurs kilomètres de long sans devoir faire une reconstruction 3D très coûteuse en temps de calcul et en mémoire vive. Il suffit d'enchainer des morceaux de trajectoire d'une centaine de mètres.

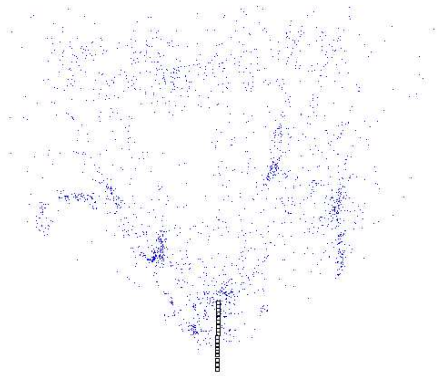

FIG. 10 – Reconstruction 3D utilisée pour évaluer la précision de l'orientation

4.3 Précision sur l'orientation

Pour mesurer la précision de l'orientation fournie par l'algorithme de localisation , nous avons placé la caméra sur une plate-forme rotative qui peut être orientée précisément avec un graduation au degré près (l'erreur sur la lecture de la graduation est de l'ordre de $\pm 0.1°$). La séquence d'apprentissage était une séquence composée de 18 images clef enregistrées sur une trajectoire en ligne droite de 1 *m* de long orientée dans le même sens que l'axe optique de la caméra (ce qui correspond à l'orientation $0°$). La reconstruction de la trajectoire ainsi que le nuage de points associé est visible en vue de dessus sur la figure 10. L'objectif utilisé donne un champ de $130°$ dans la diagonale de l'image.

Dans la phase de localisation, pour chaque orientation de la caméra de $\alpha_0 = -94°$ à $\alpha_0 = +94°$ par incrément de $2°$, nous avons noté l'angle α_0 mesuré sur la graduation de la plate-forme et l'angle α fourni par l'algorithme de localisation. Au delà de $95°$, la zone d'image commune entre l'image courante et l'image de référence devient très petite et on n'est plus certain de trouver un nombre de points en correspondance suffisant pour continuer à se localiser. L'erreur de localisation calculée à partir de ces données apparaît sur la figure 11. Des images capturées pour les orientations $0°$, $44°$ et $90°$ apparaissent en figure 12.

L'algorithme de localisation est capable de fournir une orientation avec une précision de l'ordre du dixième de degré même si on s'écarte de $90°$ d'un côté ou de l'autre par rapport à l'axe d'apprentissage. Pour quantifier plus précisément l'erreur, il faudrait disposer d'un moyen de mesure plus précis de l'orientation de la plate-forme. La lecture de la graduation ne donne pas une précision meilleure que l'algorithme de localisation.

4.4 Temps de calcul et espace mémoire

Les mesures sont faites sur un Pentium 4 à 3.4GHz avec des images de 640x480 pixels et 1500 points d'intérêt détectés par image. Le code utilise le jeu d'instructions

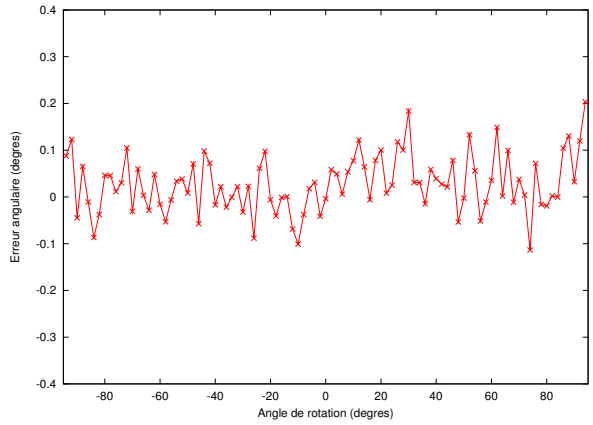

FIG. 11 – Erreur angulaire $|\alpha_0| - |\alpha|$ en fonction de α_0

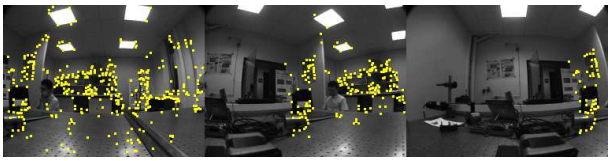

FIG. 12 – De gauche à droite images prises à $0°$, $44°$ et $90°$, avec les points d'intérêt correctement appariés en surimpression

SSE2 pour toutes les opérations de traitement d'image. Le temps de reconstruction pour la séquence $campus_1$ (113 images clef) est d'environ une heure. Le processus de localisation tourne en 60 *ms* La détection des points d'intérêt prend 35 *ms*, l'appariement 15 *ms*, et le calcul de pose avec RANSAC et l'optimisation itérative demandent 10 *ms*. La reconstruction de la séquence $campus_1$ occupe 38 *Mo* sans compression de données.

5 Conclusion

Nous avons présenté une méthode de localisation temps réel pour un robot mobile au voisinage d'une trajectoire apprise. Après un parcours en conduite manuelle, on calcule une carte de l'environnement hors ligne. A l'aide de cette carte, le robot est capable de se localiser. La précision de notre algorithme a été mesurée en extérieur à l'aide d'un récepteur GPS différentiel. L'erreur moyenne de localisation par rapport à la trajectoire de référence est de l'ordre de 2 *cm*, ce qui est quasiment équivalent au GPS différentiel. La vision permet en plus de donner une orientation de la caméra précise au dixième de degré. Ce système a été utilisé avec succès comme seul capteur pour la navigation autonome d'un robot mobile. De plus, notre algorithme est bien adapté aux rues étroites et aux zones urbaines denses où les signaux GPS ne peuvent pas être utilisés. La vision constitue donc un complément intéressant au GPS. Les conditions où chaque capteur présente des défaillances sont différentes. Nous travaillons actuellement sur le calcul en ligne d'un ellipsoïde d'incertitude associé à la localisation obtenue. C'est particulièrement important pour intégrer la localisation par vision dans un système multi-capteurs où une étape de fusion de données doit être mise en œuvre. L'autre axe sur lequel nous souhaitons travailler est la mise à jour de la base d'amers en fonction des changements dans l'environnement.

Références

[1] H. Araújo, R.J. Carceroni, and C.M. Brown. A fully projective formulation to improve the accuracy of Lowe's pose estimation algorithm. *Computer Vision and Image Understanding*, 70(2) :227–238, 1998.

[2] P. Beardsley, P. Torr, and A. Zisserman. 3d model acquisition from extended image sequences. In *European Conference on Computer Vision*, pages 683–695, 1996.

[3] G. Blanc, Y. Mezouar, and P. Martinet. Indoor navigation of a wheeled mobile robot along visual routes. In *IEEE International Conference on Robotics and Automation,ICRA'05*, Barcelonne, Espagne, 18-22 Avril 2005.

[4] D. Cobzas, H. Zhang, and M. Jagersand. Image-based localization with depth-enhanced image map. In *International Conference on Robotics and Automation*, 2003.

[5] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the 9th International Conference on Computer Vision, Nice*, 2003.

[6] O. Faugeras and M. Herbert. The representation, recognition, and locating of 3-d objects. *International Journal of Robotic Research*, 5(3) :27–52, 1986.

[7] O. Fischler and R. Bolles. Random sample consensus : a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, 24 :381–395, 1981.

[8] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3) :331–356, 1994.

[9] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[10] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

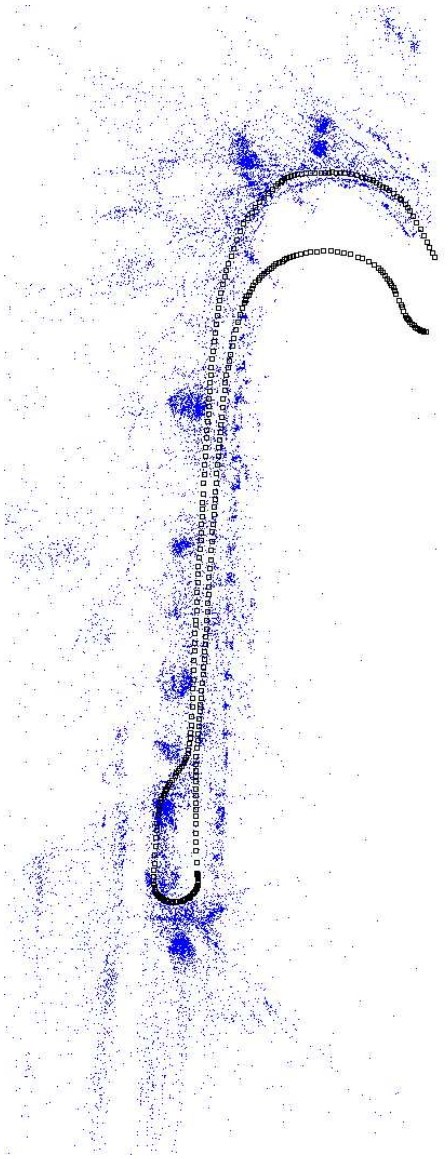

FIG. 13 – Reconstruction 3D d'une séquence de 400 *m* de long en aller-retour

FIG. 14 – Quelques images clefs de la séquence de 400 *m* de long

[11] K. Kidono, J. Miura, and Y. Shirai. Autonomous visual navigation of a mobile robot using a human-guided experience. *Robotics and Autonomous Systems*, 40(2-3) :124–1332, 2002.

[12] J. M. Lavest, M. Viala, and M. Dhome. Do we need an accurate calibration pattern to achieve a reliable camera calibration ? In *European Conference on Computer Vision*, pages 158–174, 1998.

[13] Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using view-sequenced route representation. In *International Conference on Robotics and Automation*, pages 83–88, 1996.

[14] D. Nistér. An efficient solution to the five-point relative pose problem. In *Conference on Computer Vision and Pattern Recognition*, pages 147–151, 2003.

[15] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *Conference on Computer Vision and Pattern Recognition*, pages 652–659, 2004.

[16] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *International Conference on Computer Vision*, pages 90–95, 1998.

[17] A. Remazeilles, F. Chaumette, and P. Gros. Robot motion control from a visual memory. In *International Conference on Robotics and Automation*, volume 4, pages 4695–4700, 2004.

[18] E. Royer, J. Bom, M. Dhome, B. Thuilot, M. Lhuillier, and F. Marmoiton. Outdoor autonomous navigation using monocular vision. In *International Conference on Intelligent Robots and Systems*, 2005.

[19] B. Thuilot, J. Bom, F. Marmoiton, and P. Martinet. Accurate automatic guidance of an urban vehicle relying on a kinematic gps sensor. In *Symposium on Intelligent Autonomous Vehicles IAV04*, 2004.

[20] L. Vacchetti, V. Lepetit, and P. Fua. Stable 3–d tracking in real-time using integrated context information. In *Conference on Computer Vision and Pattern Recognition, Madison, WI*, June 2003.