Localisation temps-réel d'objets complexes

A. Loesch[1] S. Bourgeois[1] V. Gay-Bellile[1] M. Dhome[2]

[1] CEA, LIST, Laboratoire Vision et Ingénierie des Contenus
Point Courrier 94, Gif-sur-Yvette, F-91191 France
[2] Clermont Université, Université Blaise Pascal, LASMEA, BP 10448
Clermont-Ferrand / CNRS, UMR 6602, LASMEA, AUBIERE

angelique.loesch@cea.fr

Résumé

Cet article adresse la problématique de localisation en temps-réel d'une caméra RGB vis à vis d'un objet complexe, c'est à dire peu texturé, présentant des surfaces courbes et peu de contours francs. Plus précisément, ces travaux proposent une solution compatible avec les exigences de qualité et de facilité d'exploitation requises par les applications industrielles. Pour cela, une méthode de SLAM contraint à un modèle CAO est combinée avec une méthode d'identification de points de contours 3D de ce même modèle et basée sur des rendus virtuels. Les résultats obtenus à partir de données de synthèse et réelles, montrent que cette solution est robuste, précise et stable aux mouvements brusques et aux occultations, mais aussi facile à déployer et capable de gérer des objets de toute forme.

Mots Clef

SLAM, contours d'auto-occultation, ajustement de faisceaux, temps-réel.

Abstract

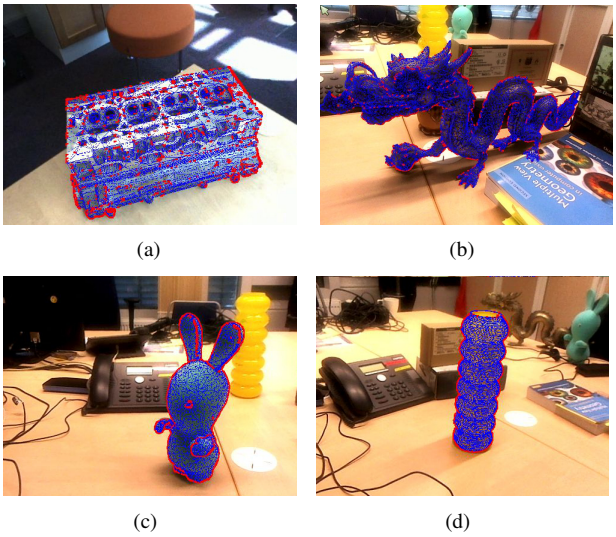
This paper addresses the challenging issue of real-time camera localization relative to a complex object i.e. textureless, with curved surfaces and very few sharp contours. From the CAD model of the observed object, 3D contour points are extracted dynamically by rendering on the graphic hardware. Combined to a keyframe-based SLAM algorithm, the detection of occluding contours of the object allows the improvement of the localization accuracy. The suggested solution is robust, accurate and stable to sudden motions and to occlusions, easy to deploy and real-time. The benefits of this method are demonstrated on synthetic and real data.

Keywords

SLAM, occluding contours, bundle adjustment, real-time.

1 Introduction

Un besoin applicatif existe en terme de localisation 3D d'objets par vision. Cette technologie devient en effet de plus en plus populaire dans le milieu industriel où elle peut être utile lors de contrôle qualité, de robotisation de tâches ou encore d'aide à la maintenance par Réalité Augmentée. Néanmoins, le déploiement de telles applications reste anecdotique en raison de la difficulté à allier qualité de localisation, facilité de mise en œuvre et généricité de la solution. En effet, la majorité des solutions implique : soit des étapes de mise en œuvre complexes, soit une restriction sur la nature des objets pouvant être traités (objets texturés, polyédriques, de petit volume...), soit une restriction sur la qualité du suivi (tremblements, manque de robustesse aux occultations...). Ces restrictions sur la nature de l'objet excluent ainsi des expériences de suivi, les objets dits complexes (Fig. 1), c'est-à dire peu texturés, courbes, avec peu d'arêtes franches et des contours d'auto-occultation (non statiques et apparaissant selon le point de vue), bien que ceux-ci soient nombreux dans le milieu industriel. De plus, certaines applications industrielles requièrent de



(a) (b) (c) (d)

FIGURE 1 – Notre solution localise des objets de différentes natures, comme une pièce industrielle (a), un dragon en métal (b), un lapin non texturé (c), un vase en verre (d).

pouvoir gérer des déplacements rapides, ou encore des occultations partielles ou totales de l'objet, ce qui n'est pas toujours possible avec les solutions existantes.
Les travaux présentés dans cet article visent à fournir une solution répondant simultanément à l'ensemble de ces besoins. Aussi celle-ci se restreint-elle au seul usage d'une caméra couleur, les caméras RGBD impliquant généralement une limite sur le volume, la nature réflective ou absorbante de l'objet, et sur la luminosité de son environnement. Cette étude est en outre restreinte à la seule exploitation de modèles CAO non texturés, les textures pouvant difficilement être considérées comme stables dans le temps (usure, taches...) et pouvant varier pour un même objet manufacturé. De plus, les modèles à base de nuage de descripteurs locaux ou les modèles surfaciques texturés sont actuellement des données peu disponibles dans l'industrie.

2 Etat de l'art et positionnement

Le suivi d'objet consiste généralement à établir des correspondances entre des primitives 2D observées dans une image et les primitives 3D issues d'un modèle de l'objet, susceptibles d'avoir généré de telles observations, afin d'estimer au mieux la pose de la caméra. Lorsque le modèle 3D correspond à un modèle CAO dépourvu de texture, les primitives 2D considérées sont généralement des points de contours de l'image et les primitives 3D correspondent alors à des points de surface orientés du modèle appelés edgelets dans la suite de l'article. L'étape de mise en correspondance s'avère alors particulièrement difficile, ceci pour deux raisons :

– les points de contours 2D étant des primitives visuelles localement peu discriminantes (difficulté de distinguer un point d'un contour d'un autre), la mise en correspondance 2D/3D n'est pas toujours déterminée avec certitude.
– les points de surface susceptibles de générer des contours dans l'image peuvent dépendre du point de vue d'observation (silhouette, contours d'auto-occultation,...).

Deux hypothèses simplificatrices ont été proposées pour résoudre ces problèmes. La première consiste à se limiter aux objets polyédriques afin de réduire les points de surface susceptibles de générer des contours 2D aux seules arêtes franches du modèle CAO. Ces primitives 3D ne dépendent pas du point de vue et peuvent être identifiées préalablement parmi les arêtes du modèle par simple seuillage sur angle diédral. Pour autant, tous les objets n'ont pas naturellement des arêtes franches, certains présentant des arêtes lissées en arrondis. Ce problème peut être négligé, mais le critère d'angle diédral ne permet alors plus d'identifier automatiquement ces contours 3D. Une simplification du modèle peut résoudre ce problème en supprimant les arrondis, mais le contrôle du degré de simplification nécessite généralement un ajustement supervisé par un expert, rendant la solution difficilement accessible. La seconde hypothèse consiste à supposer que, entre deux images du flux vidéo, le déplacement relatif entre l'objet et la caméra est faible. Cette hypothèse de faibles mouvements permet de réduire les points 2D candidats à l'appariement, aux seuls points de contours se situant au voisinage du projeté d'un point 3D selon la pose estimée à l'image précédente [2, 12]. Cependant il en découle un manque de robustesse en présence de mouvements brusques et rapides, réduisant le domaine d'application de cette technologie.
Afin d'obtenir une solution plus robuste et facile à déployer, différents travaux ont essayés d'éviter de faire l'une ou l'autre de ces hypothèses. Ainsi, une première famille de suivis d'objets basés modèle s'est attachée à supprimer l'hypothèse d'objets polyédriques. Deux approches au sein de cette famille se distinguent. La première consiste à approximer localement la surface du modèle CAO à l'aide d'une représentation paramétrique en s'appuyant sur des rayons de courbure [8] ou sur des quadriques [6]. Ces représentations paramétriques ont l'avantage d'être dérivables et donc facilement intégrable dans une fonction de coût à optimiser. Cependant, le nombre de quadriques ou de rayons de courbure peut rapidement augmenter et compliquer la paramétrisation du modèle. Aussi ce nombre doit-il être limité, malgré une approximation des surfaces pour des objets complexes et un suivi moins précis.
La seconde approche consiste à identifier les points de surface de l'objet susceptibles de générer des contours de l'image, à partir de rendus de synthèse de l'objet pour un point de vue donné [9, 13]. Pour que l'approximation des contours francs et d'auto-occultation soit indépendante des conditions d'illuminations [3], ces méthodes s'appuient sur des rendus non photo-réalistes. Ces méthodes ont l'avantage d'être efficaces sur tous types d'objets, aussi bien courbes que polyédraux. Elles prennent de plus en compte la distance d'observation lors de l'extraction des points 3D sur l'image de synthèse et exploite le modèle sans simplification de la part de l'industriel. Cependant, le modèle étant non paramétrique, les contours ne sont valides que pour des poses proches et donc uniquement pour des déplacements de faibles amplitudes. L'extraction des contours est alors réalisée en ligne à chaque image en utilisant la pose de la caméra précédente. Il en résulte un coût assez important au niveau des temps de calcul et de transfert de données entre le GPU et le CPU, en particulier lorsque le modèle géométrique possède beaucoup de facettes ou que l'image est de haute résolution.

Une seconde famille vise à résoudre les problèmes liés à l'hypothèse des petits mouvements. Une première solution consiste à réaliser un suivi à l'aide d'un filtre particulaire [4] afin d'éliminer cette hypothèse en générant un nombre important d'hypothèses sur la pose courante. Les particules échantillonnent l'espace des poses à 6 dimensions, et le score attribué à chaque particule correspond à

une mesure d'écart entre la projection des arêtes franches du modèle CAO selon la pose de la particule, et le contour de l'image le plus proche. Si cette approche offre un suivi robuste aux mouvements brusques et aux occultations, le coût calculatoire d'une telle méthode n'est pas compatible avec une exécution temps-réel pour un modèle complexe.
Une autre stratégie pour éviter l'hypothèse de petits déplacements est de prédire la pose de la caméra sur l'image courante. Une première possibilité est d'utiliser un capteur externe tel qu'une centrale inertielle [1]. D'autres approches reposent sur un module additionnel de prédiction de la pose de la caméra, celle-ci étant utile pour l'initialisation de l'algorithme de recalage sur les contours. Généralement, ce module de prédiction repose sur une estimation du mouvement de la caméra à l'aide de points d'intérêts, ce type de primitives étant plus discriminant et donc plus robuste aux larges déplacements. Ainsi, une première solution consiste à reconstruire dynamiquement des points de texture à la surface de l'objet [9, 11]. Ce type de solution étant peu robuste et peu précise lorsque l'objet est petit dans l'image ou occulté, une seconde solution consiste à reconstruire des points 3D sur l'environnement complet à l'aide d'algorithme de type SLAM. Le *Simultaneous Localisation And Mapping* est une méthode considérant une caméra qui évolue dans un environnement inconnu. Elle réalise une reconstruction en ligne d'une carte 3D de celui-ci à partir de primitives, extraites des images d'une séquence vidéo au cours du temps [5, 7].
Le mouvement estimé à partir de l'environnement permet de prédire la position de l'objet quelque soit les conditions d'observation de ce dernier, ceci sous l'hypothèse que l'objet reste statique vis à vis de celui-ci. Une méthode de SLAM contraint basé images clés [10] propose d'améliorer la reconstruction des points 3D en exploitant l'alignement du modèle sur les contours de l'image comme une contrainte supplémentaire dans l'ajustement de faisceaux de l'algorithme. Cette méthode intègre simultanément les contraintes issues des contours francs du modèle et les contraintes multi-vues apportées par l'ensemble de la scène. Ainsi, lorsque l'objet est bien visible dans l'image, les contraintes issues du modèle garantissent une localisation et une reconstruction précises de l'environnement. Mais il peut également être localisé à partir de ce seul environnement, même en étant petit, occulté ou en dehors du champ de vision. L'hypothèse de suivre un objet statique est une contrainte raisonnable, l'objet étant fixe au cours d'expériences de Réalité Augmentée dans la majorité des applications industrielles. Bien que précis et temps-réel, cette approche atteint ses limites pour des objets courbes sans contour franc puisque seules les arêtes franches du modèle sont exploitées.

Cet article propose deux contributions. La première correspond à une génération d'edgelets par rendu qui résulte en une constellation d'edgelets facilitant l'étape du matching et améliorant l'estimation de la pose. La seconde consiste en l'intégration des edgelets précédemment générés dans un algorithme de SLAM contraint temps-réel.

3 Génération de contours adaptés au calcul de pose

Comme mentionné en Section 2, les solutions de suivi basées sur des techniques d'analyse par synthèse s'appuient sur des rendus de modèles 3D, afin d'identifier les points de surfaces susceptibles de générer un contour. Pour des raisons de performance, ces points de surfaces sont échantillonnés en un ensemble d'edgelets et cela avant l'étape d'appariement. Cependant, dans la plupart des solutions de suivi, l'importance de l'échantillonnage est négligée. Bien que celui-ci ait un impact important sur l'appariement des edgelets avec les contours 2D dans l'image et sur l'estimation de la pose de la caméra à partir de ces correspondances, la plupart des approches s'appuie sur un échantillonnage classique avec un pas constant.
Dans cette section, une nouvelle stratégie d'échantillonnage est introduite. Celle-ci fournit une constellation d'edgelets adaptée à la fois à l'appariement et à l'estimation de la pose.

3.1 Génération des contours

Notre solution cherche à estimer pour chaque pixel de l'image de rendu la probabilité de générer un contour dans l'image réelle et d'être correctement apparié à celui-ci.
Dans ce qui suit, considérons chaque pixel comme une variable aléatoire X_i, avec $X_i = 1$ si le pixel est un contour, 0 sinon. Notons Y_i la variable aléatoire qui représente la direction 2D d'un contour X_i, avec $Y_i \in \{Nord, Nord-Est, Nord-Ouest, Ouest\}$. $L(X_i,Y_i)$ (respectivement $R(X_i,Y_i)$) correspond à la fonction qui retourne le voisin gauche (respectivement droit) d'un pixel X_i en fonction de la direction Y_i, et $Nmap(X_i)$ (respectivement $Dmap(X_i)$) la fonction qui retourne la normale (respectivement la profondeur) d'un pixel X_i.

Probabilité d'apparition d'un contour. La première étape de notre génération d'edgelets consiste à estimer pour chaque pixel la probabilité que celui-ci corresponde à un contour dans l'image. Deux types de contours 2D sont communément distingués : les contours francs correspondant à des discontinuités sur la carte des normales ($Nmap$) et les contours d'occultations incluant la silhouette correspondant à des discontinuités sur la carte de profondeur ($Dmap$). Dans les deux cas, l'orientation d'un point de contour 2D correspond à la direction 2D de la discontinuité dans l'image.
Ainsi la probabilité de X_i d'être un contour franc avec la direction Y_i est définie par :

$$\mathbb{P}_{crease}(X_i|Y_i) = \max(1, crease(X_i,Y_i)), \tag{1}$$

avec la mesure de discontinuité angulaire :

$$crease(X_i,Y_i) = \lambda \times (1 - Nmap(L(X_i,Y_i)) \cdot Nmap(R(X_i,Y_i))), \tag{2}$$

où $\lambda = (1 - cos(angleMax))^{-1}$ est le facteur de normalisation défini afin d'atteindre une intensité de 1 pour une amplitude angulaire de $angleMax$.
De façon similaire, la probabilité de X_i d'appartenir à un contour d'occultation avec une orientation Y_i est donnée par :

$$\mathbb{P}_{silhouette}(X_i|Y_i) = \max(1, silhouette(X_i, Y_i)), \quad (3)$$

où $silhouette$ est la mesure de discontinuité en profondeur définie comme suit :

$$silhouette(X_i, Y_i) = \frac{\text{Laplacian}(Dmap, X_i, Y_i)}{\beta \times Dmap(X_i)}, \quad (4)$$

où Laplacian correspond au laplacien 1D orienté en fonction de la direction Y_i, et β est un facteur de pondération. Puisque la valeur de discontinuité est normalisée en fonction de la distance d'observation, le paramètre β ne dépend pas des dimensions de la scène. Ainsi, β peut-il aisément être interprété comme la discontinuité minimale en profondeur, exprimée comme un ratio de la distance d'observation qui fournit un contour d'occultation avec une probabilité de 1. Par conséquent la probabilité d'être un contour est définie par :

$$\mathbb{P}_{contour}(X_i) = \max_{Y_j}(\max(\mathbb{P}_{crease}(X_i|Y_j), \mathbb{P}_{silhouette}(X_i|Y_j))) \quad (5)$$

et la direction associée à X_i est définie par :

$$Dir(X_i) = \arg\max_{Y_j}(\max(\mathbb{P}_{crease}(X_i|Y_j), \mathbb{P}_{silhouette}(X_i|Y_j))). \quad (6)$$

Probabilité de mise en correspondance. La seconde étape consiste à estimer la probabilité que chaque contour virtuel soit apparié au contour correspondant dans l'image réelle. Généralement, dans le processus d'appariement, les edgelets sont associés au contour le plus proche localement le long de la normal du contour. Par conséquent, la probabilité d'apparier correctement un edgelet peut être évaluée à partir du nombre de contours rencontrés dans ce voisinage 1D le long de la normale du contour. Ainsi, proposons-nous d'évaluer cette probabilité à partir de la carte de probabilité $\mathbb{P}_{contour}$ estimée dans l'étape précédente. Dans ce qui suit, considérons X_n comme étant les pixels du voisinage 1D du pixel X_i, et S la variable aléatoire représentant le nombre de contours dans le voisinage. Nous définissons la probabilité d'un pixel d'être à la fois un contour et correctement apparié dans l'image réelle d'après l'espérance de S par :

$$\mathbb{P}_{match}(X_i) = \frac{\mathbb{P}_{contour}(X_i)}{1 + \mathbb{E}(S)}. \quad (7)$$

3.2 Échantillonnage des contours

Dans l'étape précédente, une carte des edgelets $\mathbb{P}_{match}$ fournissant pour chaque pixel sa probabilité d'être à la fois un contour et correctement apparié, est estimée. Cette carte est alors exploitée dans l'étape d'échantillonnage afin de fournir un ensemble d'edgelets pertinent pour les processus d'appariement et d'estimation de la pose.

Afin d'être approprié au processus de mise en correspondance, cet ensemble d'edgelets doit maximiser la prévision du succès de l'appariement. Cependant, pour être également pertinent au processus d'estimation de la pose, les appariements estimés à partir de la constellation d'edgelets doit également contraindre les 6 degrés de liberté de la pose de la caméra. En particulier, un ensemble de mises en correspondance qui n'est pas uniformément distribué dans l'image en terme de position et d'orientation de contour, ne correspond pas à une configuration pertinente pour l'estimation de la pose.
Notre stratégie d'échantillonnage s'appuie donc sur une division à la fois de l'espace des positions 2D et de l'espace des orientations 2D des edgelets projetés dans l'image. L'espace des positions 2D est divisé par un quadrillage régulier de $N \times N$ baquets, et chaque baquet est lui-même divisé en 4 secteurs angulaires. Un ensemble d'edgelets est ensuite échantillonné pour chaque secteur angulaire de chaque baquet spatial, afin d'obtenir une distribution uniforme dans l'espace 2D. L'échantillonnage est lui-même réalisé en fonction d'une probabilité d'échantillonnage affectée à chaque edgelet. Pour définir cette probabilité, considérons $\{Z_i\}$ comme étant l'ensemble des edgelets d'un secteur angulaire d'un baquet spatial. La probabilité d'échantillonnage $\mathbb{P}_{sampling}$ d'un edgelet $Z \in \{Z_i\}$ est définie comme suit :

$$\mathbb{P}_{sampling}(Z) = \frac{\mathbb{P}_{match}(Z)}{\sum_i \mathbb{P}_{match}(Z_i)}. \quad (8)$$

Avec cette stratégie d'échantillonnage, à l'intérieur d'un secteur angulaire d'un baquet spatial, les edgelets avec une forte probabilité d'appariement $\mathbb{P}_{match}$ ont plus de chance d'être échantillonnés. De plus, la nature aléatoire de l'échantillonnage diminue le risque d'agglomération locale des edglelets.

4 Intégration dans le SLAM

Si notre méthode favorise une bonne distribution des edgelets, elle ne permet pas de garantir que celle-ci contraigne tous les degrés de liberté. Dans le cas d'objets présentant des symétries (boule, vase...), il est en effet impossible de contraindre l'ensemble de ces degrés de liberté. De plus, éviter d'échantillonner les edgelets ambigus car peu saillants, réduit le risque de faux appariements sans supprimer l'hypothèse de faibles mouvements. Le processus d'extraction d'edgelets est donc combiné à un algorithme de SLAM proposé dans [10] qui se base principalement sur [7], afin de résoudre ces deux problèmes. La détection par rendu est intégrée au niveau de l'ajustement de faisceaux dans une implémentation parallèle afin de ne pas dépendre du temps d'exécution de cette extraction et d'avoir des performances temps-réel.

4.1 Vue d'ensemble

L'algorithme du SLAM [7] s'appuie la reconstruction en ligne de la carte 3D de l'environnement pour localiser en

temps-réel la caméra. Cette carte est ensuite enrichie au fur et à mesure à chaque image clé par l'ajout de nouvelles primitives. Un algorithme d'optimisation non linéaire appelé ajustement de faisceaux est par ailleurs régulièrement appliqué afin de réduire localement les erreurs accumulées. Celui-ci minimise les erreurs de reprojection des points 3D observés dans les $n_i = card(A_i)$ images clés. La fonction de coût de l'ajustement de faisceaux est donnée par :

$$E(\{R_j,\mathbf{t}_j\}_{j=1}^{m},\{\mathbf{Q}_i\}_{i=1}^{N}) = \sum_{i=1}^{N}\sum_{j\in A_i} d^2(\mathbf{q}_{i,j}, KP_j\mathbf{Q}_i) \quad (9)$$

où K est la matrice des paramètres intrinsèques de la caméra et P_j est sa $j^{ème}$ pose. $\mathbf{q}_{i,j}$ est l'observation 2D du $i^{ème}$ edgelet Q_i dans la $j^{ème}$ image clé. A_i correspond à l'ensemble des indexes des images clés observant Q_i.
Dans notre cas, nous cherchons à localiser la caméra par rapport à un objet 3D complexe de la scène en nous appuyant sur la connaissance d'un modèle CAO de cet objet. Nous nous inspirons pour ce faire, d'un algorithme de SLAM contraint par le modèle qui permet un meilleur suivi de l'objet mis en scène [10], la caméra étant alors considérée comme se déplaçant dans un environnement partiellement connu. L'algorithme SLAM contraint estime précisément la pose de la caméra par rapport à cet objet en incluant dans le processus d'optimisation du SLAM les contraintes géométriques relatives à son modèle CAO. Contraindre ainsi l'ajustement de faisceaux permet une meilleure stabilité de recalage et une meilleure robustesse aux occultations.

Le SLAM contraint basé images clés de [10] est amélioré en remplaçant les edgelets précalculés par des edgelets générés dynamiquement en fonction de la pose de chaque image clé. Ces derniers sont alors générés à l'initialisation et à chaque image clé, contrairement au suivi basé modèle qui nécessite un rendu à chaque image. Le SLAM estime en effet une pose précise entre deux images clés en se basant sur les points d'intérêt de l'environnement de la scène. Les edgelets générés sont ainsi extraits à partir d'une pose assez précise contrairement aux approches basées modèles qui utilisent généralement la pose estimée précédente.
De plus, afin de masquer les temps de transfert du GPU vers le CPU, notre approche d'analyse par synthèse est réalisée avec l'ajustement de faisceaux dans un thread séparé de celui de la localisation. L'algorithme du SLAM peut continuer à estimer la pose de la caméra sur la suite de la vidéo, sans latence potentielle due à l'extraction des edgelets et leur transfert (celui-ci augmentant en fonction de la complexité du modèle CAO de l'objet).

4.2 Mise en correspondance 2D/3D

Cette étape d'association 2D/3D est réalisée à chaque image clé pour l'ajustement de faisceaux contraint par les edgelets. Après leur extraction du modèle CAO de l'objet, les points 3D orientés doivent en effet être associés aux contours détectés dans l'image courante de la séquence. Contrairement à [10], aucun test de visibilité n'est nécessaire pour déterminer les edgelets visibles à l'image clé puisque tous les points 3D exploités sont extraits d'un rendu du modèle avec la position de la caméra de l'image précédente. Tous les edgelets sont donc projetés dans l'image clé et une recherche le long de la normale au point projeté est réalisée. Cette recherche permet de trouver le maximum de gradient dans le voisinage correspondant à notre edgelet.
La pose sur l'image courante étant estimée assez précisément de part le SLAM, on ne conserve que le contour le plus proche ayant une orientation semblable. De plus, les ambiguïtés d'appariement sont également réduites grâce à la probabilité de mise en correspondance $\mathbb{P}_{match}$ définie pour chaque edgelet : ceux-ci ne sont pas extraits du modèle CAO ou ont peu de chance d'être pris dans l'échantillonnage s'il y a trop de contours ayant la même direction dans leur voisinage proche.

4.3 Fonction de coût

La fonction de coût de l'ajustement de faisceaux correspond à celle proposée dans [10] et est composée des informations issues du modèle géométrique et des relations multi-vues entre les images, toutes exprimées en pixels. La fonction de coût se présente comme suit :

$$E(\{R_j,\mathbf{t}_j\}_{j=1}^{m},\{\mathbf{Q}_i\}_{i=1}^{N}) =$$

$$\sum_{i=1}^{N}\sum_{j\in A_i} d^2(\mathbf{q}_{i,j}, KP_j\mathbf{Q}_i) + \sum_{i=1}^{l}\sum_{j\in S_i} |\mathbf{n}_{i,j}.(\mathbf{m}_{i,j} - KP_j\mathbf{M}_i)| \quad (10)$$

Cette fonction de coût minimise la distance orthogonale entre le projeté de l'edgelet $\mathbf{M}_i$ appartenant à l'ensemble des edgelets L_i et le contour $\mathbf{m}_{i,j}$ associé à $\mathbf{M}_i$, extrait aux images clés $j \in S_i$ et avec lequel il est mis en correspondance dans l'image. $\mathbf{n}_{i,j}$ correspond à la normale de la projection de l'edgelet.

5 Évaluations expérimentales

Dans cette section, nous évaluons à la fois la précision et la robustesse de notre solution de suivi 3D. Afin de mieux apprécier les performances de notre méthode, nous la comparons à un suivi d'objet basé modèle par rendu similaire à [13], et au SLAM contraint segment de [10]. Ces évaluations sont menées sur des objets de différentes natures afin d'évaluer la généricité des méthodes. Cette évaluation a été menée à la fois sur des données de synthèses et des données réelles. Nous utilisons pour cela un processeur Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz et une carte graphique NVIDIA GeForce GT 730M. La contrainte temps-réel est dans ces conditions respectée même sur des modèles complexes de plus de 200K facettes. Les temps de calcul de l'ensemble du rendu avec le transfert sur le CPU, sont en effet de l'ordre de 50ms pour des séquences de haute résolution (1280×720). Ils sont réalisés entre deux images clés de notre algorithme de SLAM (la partie lo-

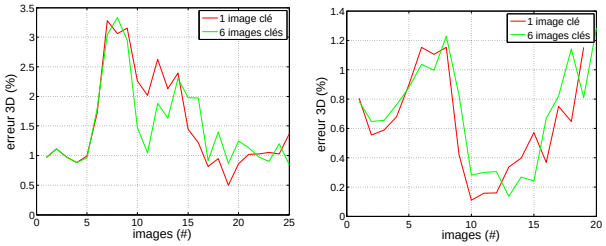
calisation tourne à 60 images par seconde), dont le temps moyen entre deux images clés est d'environ 2s.

5.1 Données de synthèse

Notre méthode a été testée sur différentes séquences de synthèse afin de l'évaluer par rapport à une vérité terrain sur la trajectoire de la caméra. Plusieurs objets avec différents niveaux de complexité ont été utilisés : un nain à 7K facettes ou un dragon à 100K facettes, et une orthèse à 264K facettes pour le modèle original et 93K facettes pour le modèle simplifié. Les deux premiers objets présentent principalement des courbures tandis que le dernier possède de nombreuses arêtes franches comme illustrée sur la Fig. 4. Ces différents objets sont de plus intégrés dans un environnement composé de quatre murs de brique et d'un sol gris et lisse (Fig. 4). La trajectoire est dans chaque séquence identique, seul l'objet suivi est modifié. Celui-ci ne se trouve par ailleurs pas toujours au centre du champ et est parfois occulté. Les résultats quantitatifs sont obtenus en mesurant la différence entre la vérité terrain et les positions estimées de la caméra. Ces erreurs sont par ailleurs exprimées en pourcentage par rapport à la distance caméra/objet.

Evaluation d'un paramètre de la fonction de coût. L'apport du nombre de caméras remettant en cause les edgelets du modèle dans l'ajustement de faisceaux a d'abord été évalué. Avant d'effectuer le processus d'optimisation, le rendu est ainsi réalisé sur la dernière image clé ou sur l'ensemble des six images clés optimisées dans l'ajustement de faisceaux. Lorsque les edgelets sont extraits pour la première fois sur la dernière image clé uniquement, ils ne sont pas remis en cause. En revanche, pour le cas d'une génération sur les six dernières images clés, les edgelets extraits sur la pénultième et les plus anciennes suivantes sont quant à eux remis en cause juste avant l'ajustement de faisceaux. Les résultats sont illustrés sur la Fig. 2. Celle-ci montre l'erreur sur la translation à chaque image clé lorsque les edgelets sont générés sur la dernière ou les six dernières images clés de l'ajustement de faisceaux. Que ce soit pour la séquence du dragon ou de l'orthèse, on constate que l'erreur varie peu. Puisque la précision de notre algorithme n'est pas affectée par la remise en cause des edgelets sur plusieurs images clés, il est suffisant de le faire uniquement pour la dernière image clé afin de réduire la complexité des calculs.

Evaluation de l'échantillonnage. La stratégie d'échantillonnage proposée en Section 3.2 est comparée à un échantillonnage aléatoire. Cette évaluation est réalisée sur la séquence de l'orthèse dans laquelle notre méthode est appliquée une centaine de fois avec les deux types d'échantillonnage (400 edgelets sont utilisés). Les erreurs moyennes, minimales et maximales ainsi que l'écart-type (SD) sont estimés sur ces 100 tirages et sur l'ensemble des images de la vidéo. Les résultats sont décrits dans le



(a) (b)

FIGURE 2 – Erreur de localisation sur la séquence du dragon (a), et de l'orthèse (b), pour des edgelets remis en cause sur les six dernières images clés (vert). Lorsqu'ils sont générés uniquement sur la dernière image clé, ils ne sont pas remis en cause (rouge).

Tableau 1 et montrent que notre échantillonnage contraint mieux l'appariement et l'estimation de la pose, en réduisant de moitié l'erreur de localisation.

	min (%)	max (%)	moyenne (%)	SD (%)
Échantillonnage aléatoire	0.9228	19.0420	8.4383	8.4943
Notre échantillonnage	**0.5788**	**10.3170**	**3.5466**	**3.4458**

TABLE 1 – Erreur de localisation pour deux stratégies d'échantillonnage.

Evaluation de la précision. Notre solution est ensuite comparée à un suivi d'objet basé modèle qui extrait les edgelets à chaque image de la séquence. La Fig. 3 présente l'erreur en translation de la pose de la caméra par rapport à la vérité terrain avec le nain comme objet d'étude. Afin d'évaluer la robustesse aux grands mouvements des deux méthodes, la vitesse de la séquence vidéo a été augmentée au fur et à mesure en ne prenant qu'une image sur huit ou sur dix de la séquence originale. Notre suivi d'objet basé modèle par rendu et sans algorithme de SLAM sur lequel s'appuyer, est moins précis que notre approche de SLAM contraint par nos edgelets. Quelle que soit la vitesse de la séquence, notre approche est stable avec une erreur moyenne comprise entre 0.78% et 0.92%. Le suivi simple en revanche, a une erreur moyenne plus importante et qui augmente avec la vitesse de mouvement de la caméra. Elle est située entre 2.41% et 3.15% pour la séquence originale et la séquence ne prenant qu'une image sur huit. L'erreur moyenne est en revanche de 6.65% sur la séquence ne prenant qu'une image sur 10 avant de dériver complètement à la suite d'un mouvement trop brusque.

Nous avons également comparé notre méthode avec le SLAM basé images clés qui exploite les contours francs de l'objet comme contraintes pour l'ajustement de faisceaux [10]. Nous avons réalisé nos expériences sur les séquences du dragon pour évaluer notre précision pour les objets courbes, et sur l'orthèse pour les objets à contours francs. La Fig. 4 montre l'erreur sur la translation de la

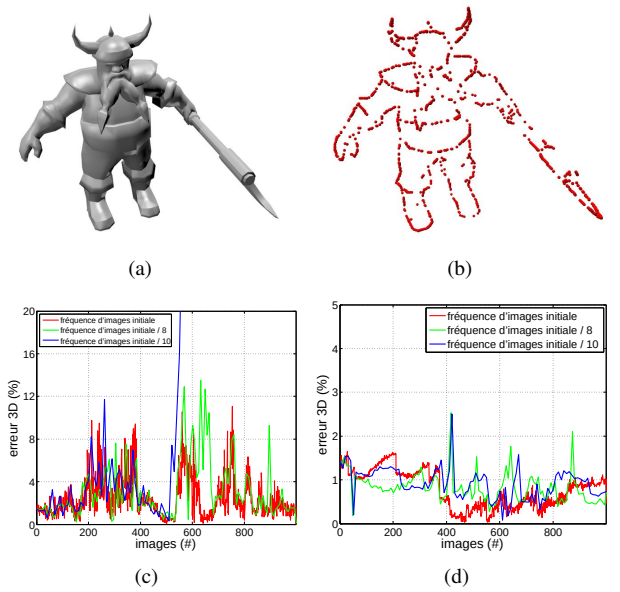


FIGURE 3 – Erreur de localisation sur la séquence du nain pour un suivi d'objet avec approche d'analyse par synthèse (c) et pour notre méthode (d) : erreur sur toutes les images (rouge), une image sur huit (vert) et une image sur 10 (bleu). (a) représente le rendu du nain et (b) les edgelets extraits avec notre approche.

position de la caméra sur le dragon et l'orthèse. Notre solution est cinq fois plus précise en moyenne pour les objets courbes que [10], et équivalente pour les objets à contours francs. De plus, notre courbe d'erreur sur l'orthèse varie peu, que le modèle soit simplifié ou non. Ce n'est pas le cas pour la méthode de SLAM exploitant les contours francs, qui présente une erreur moyenne de 1.18% pour le modèle simplifié et de 2.36% pour le modèle original. Celle-ci présente en effet de meilleurs résultats lorsque le modèle est retravaillé faisant ainsi mieux apparaître les arêtes franches.

Ainsi le suivi basé modèle fonctionne bien pour des petits déplacements mais s'avère peu robuste aux grands mouvements. Le suivi avec SLAM de [10] lui gère mieux ce type de déplacement mais se dégrade sur les objets courbes. Notre solution quant à elle ne rencontre aucune difficulté à bien se localiser quelque soit l'objet ou le mouvement de la caméra.

5.2 Données réelles

Nos résultats ont également été validés sur des séquences réelles, en utilisant notamment un objet courbe comme un bypass à 152K facettes, objet industriel composé essentiellement de tuyaux. Afin de tester la robustesse de notre solution, l'objet suivi dans les séquences fait face à des occultations et la caméra réalise de grands mouvements.

Notre approche a d'abord été comparée au SLAM contraint de [10]. La Fig. 5 montre que le SLAM contraint par les

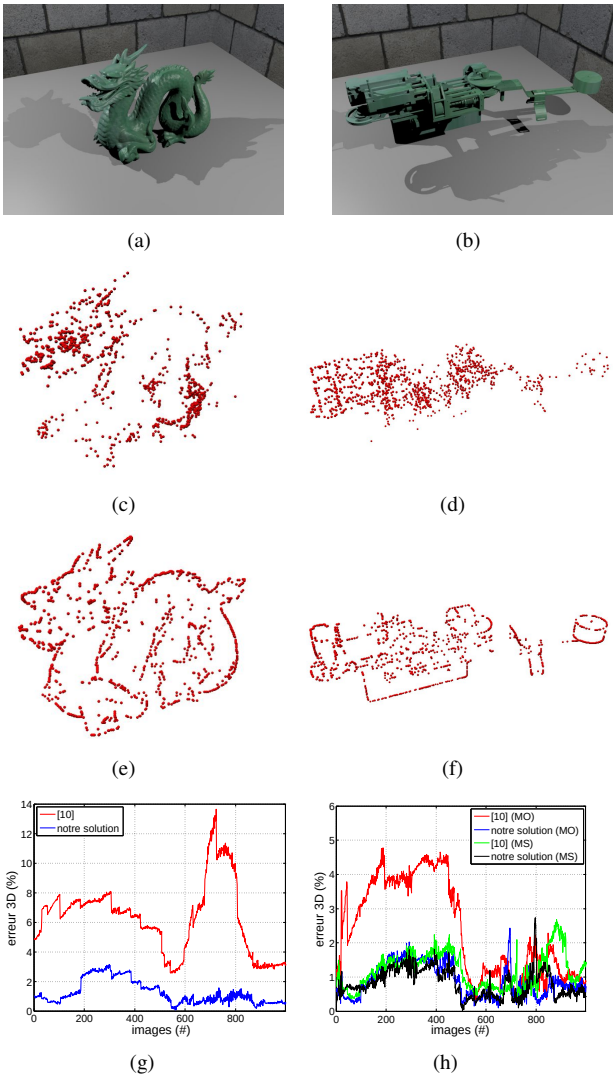
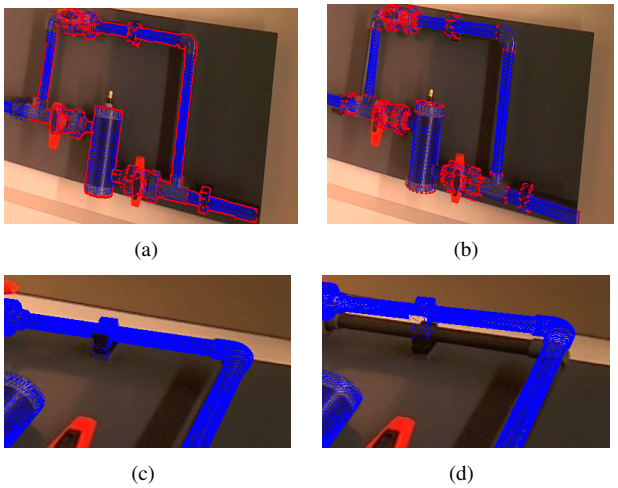


FIGURE 4 – Erreur de localisation sur la séquence du dragon (première colonne), respectivement l'orthèse (deuxième colonne). (a) et (b) représente le rendu des deux objets, (c) et (d) les edgelets extraits sur les arêtes franches, (e) et (f) les edgelets extraits par rendu. (g) compare notre solution (bleu), avec la méthode de [10] (rouge) sur le dragon. (h) compare notre solution (bleu) à [10] (rouge) sur le modèle original (MO) de l'orthèse, ainsi que sur le modèle simplifié (MS) (respectivement noir et vert).

segments ne détecte pas les edgelets de manière uniforme sur le modèle contrairement à notre approche qui les répartit de façon homogène sur l'ensemble du modèle. Les 2000 edgelets extraits sont ainsi concentrés sur les parties présentant des arêtes franches au niveau des robinets pour l'approche de [10]. L'algorithme se concentre alors sur ces zones à forts contours francs et cela ne suffit pas à recaler précisément le modèle 3D sur les zones courbes comme l'illustrent les Fig. 5(c) et 5(d).

Notre approche a par ailleurs été comparée à un suivi d'objet basé modèle par rendu similaire à [13]. Le modèle re-



(a) (b) (c) (d)

FIGURE 5 – Précision de localisation de la caméra par rapport à l'objet sur la séquence du bypass pour notre solution (gauche) et pour le SLAM contraint de [10] (droite). (a) et (b) représentent les edgelets extraits, (c) et (d) la reprojection des modèles CAO au niveau d'une partie du bypass avec peu de contours francs.

calé à tendance à trembler voire à décrocher lors de mouvements brusques, contrairement à notre solution utilisant le SLAM. C'est en effet ce dernier qui permet d'être plus stable dans le temps grâce aux points d'intérêt détectés dans l'environnement.
Enfin la généricité de notre approche a été évaluée avec succès en complément du bypass (Fig. 5) dont le modèle CAO était disponible, sur des objets dont les modèles 3D ont été reconstruits par photogrammétrie. Ces objets sont soit courbes (une statuette de dragon en métal (Fig. 1(b)), un lapin en velours (Fig. 1(c)) et un vase en verre (Fig. 1(d))), soit polyédriques (une culasse de voiture (Fig. 1(a))), et peuvent présenter dans le même temps des contours d'occultation (le bypass). Certains ont de la texture (le dragon), tandis que d'autres en sont dépourvus (le lapin, le vase). De plus le vase a la particularité d'avoir son axe vertical comme axe de symétrie.

6 Conclusion

Dans cet article une solution de localisation d'une caméra par rapport à un objet complexe avec un minimum d'*a priori* sur sa forme et respectant la contrainte temps-réel est proposée. Elle correspond à un algorithme de SLAM basé images clés contraint par des edgelets dynamiques. Ceux-ci sont extraits par rendu à chaque image clé, et échantillonnés de manière probabiliste afin d'obtenir une distribution homogène spatialement et angulairement, et d'empêcher la sélection d'edgelets ambigus dans l'étape de mise en correspondance.
Notre méthode évaluée et comparée sur des séquences de synthèse et des séquences réelles, est robuste et stable, et autorise de larges mouvements avec la caméra. Par ailleurs, notre solution de suivi a été testée sur un large panel d'objets (polyédrique, courbe, symétrique...) attestant sa généricité. Le modèle CAO n'a enfin, pas besoin d'être préparé en amont du suivi pour faire apparaître les arêtes franches. Aussi notre solution est-elle plus facile à déployer pour l'industriel.

Références

[1] G. Bleser, H. Wuest, and D. Strieker. Online camera pose estimation in partially known and dynamic scenes. In *ISMAR*, 2006.

[2] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *PAMI*, 24(7) :932–946, 2002.

[3] A. Hertzmann. Introduction to 3d non-photorealistic rendering : Silhouettes and outlines. In *SIGGRAPH*, 1999.

[4] G. Klein and D. Murray. Full-3d edge tracking with a particle filter. In *BMVC*, 2006.

[5] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR*, 2007.

[6] G. Li, Y. Tsin, and Y. Genc. Exploiting occluding contours for real-time 3d tracking : A unified approach. In *ICCV*, 2007.

[7] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *CVPR*, 2006.

[8] M. A. Oikawa, T. Taketomi, G. Yamamoto, M. Fujisawa, T. Amano, J. Miyazaki, and H. Kato. Local quadrics surface approximation for real-time tracking of textureless 3d rigid curved objects. In *SVR*, 2012.

[9] A. Petit, E. Marchand, and K. Kanani. Combining complementary edge, point and color cues in model-based tracking for highly dynamic scenes. In *ICRA*, 2014.

[10] M. Tamaazousti, V. Gay-Bellile, S. N. Collette, S. Bourgeois, and M. Dhome. Real-time accurate localization in a partially known environment : Application to augmented reality on textureless 3d objects. In *ISMAR Workshop*, 2011.

[11] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *ISMAR*, 2004.

[12] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *PAMI*, 26(10) :1385–1391, 2004.

[13] H. Wuest, F. Wientapper, and D. Stricker. Adaptable model-based tracking using analysis-by-synthesis techniques. In *CAIP*, 2007.