Modélisation de l'environnement par facettes planes pour la cartographie et la localisation simultanées par stéréovision

Cyrille Berger* Simon Lacroix
Université de Toulouse
LAAS/CNRS
7, Ave du Colonel Roche
F-31077 Toulouse Cedex 4
France

FirstName.Name@laas.fr

30 novembre 2007

Résumé

Nous proposons un enrichissement des modèles d'amers utilisés pour la cartographie et localisation simultanées (SLAM) par stéréovision. Dans ce contexte, les amers habituellement utilisés sont des points d'intérêts correspondant à un point en trois dimensions : en associant à ces points des facettes planes orientées dans l'espace (quand elles existent dans l'environnement), nous augmentons la description des amers par un repère orienté. Grâce à cette information supplémentaire, la perception d'un seul amer permet l'observation totale de la position du robot, et surtout la connaissance de l'orientation des facettes rend plus robuste l'association d'amers perçus de deux points de vue différents. L'article décrit le modèle des facettes retenu, les moyens de les détecter et de les mettre en correspondance, et présente des résultats de SLAM obtenus avec de tels modèles d'amers.

Mots Clef

Stéréovision, SLAM, facettes planes.

Abstract

In the context of stereovision SLAM, we propose a way to enrich the landmark models. Vision-based SLAM usually approaches rely on interest points associated to a point in the cartesian space : by adjoining oriented planar patches (if they are present in the environment), we augment the landmark description with an oriented frame. Thanks to this additional information, the robot pose is fully observable with the perception of a single landmark, and the knowledge of the patches orientation helps the matching of landmarks perceived from two different viewpoints. The paper depicts the chosen landmark model, the way to extract and match them, and presents some SLAM results obtained with such landmarks.

Keywords

Stereovision, SLAM, planar patches.

1 Introduction

Toute solution au problème de la cartographie et localisation simultanées (SLAM) nécessite le développement des fonctions suivantes :

- Détection des amers. Cela consiste à identifier et extraire dans les données perçues les éléments sur lesquels le robot se basera pour estimer sa position,
- Estimation de masures relatives. Deux processus sont nécessaires :
  - Estimation de la position des amers détectés relativement à la position courante du capteur : il s'agit de l'étape *d'observation*,
  - Estimation des déplacements du robot entre deux acquisitions de données : c'est l'étape de *prédiction*,
- Association des données. Les observations des amers ne sont utiles que si ils sont perçus de plusieurs positions différentes : ils doivent pour cela être correctement *mis en correspondance* dans les données acquises de différents points de vue.
- Estimation. C'est le processus qui est au cœur du SLAM : en intégrant les différentes prédictions de déplacement et observations d'amers, la technique d'estimation estime la position du robot et des amers dans un même repère.

Dans la communauté des roboticiens, la plupart des efforts ont porté sur le processus d'estimation. Différents formalismes ont été appliqués avec succès, et des contributions importantes portent sur la définition des structures des cartes d'amers, de manière à réduire la complexité algorithmique du processus d'estimation et les difficultés dûes au fait que le problème n'est pas décrit par des équations linéaires (voir un état de l'art complet et récent dans [4, 1]).

*Cyrille Berger bénéficie d'une convention Cifre établie avec Thales Optronics

Mais la plupart des fonctions impliquées dans le SLAM sont des *processus de perception*. Cela est évident pour la détection des amers et la mesure de leur position relative, qui résultent de traitements des données acquises. Le problème de la mise en correspondance des amers peut être résolu par la seule connaissance de leurs positions estimées et observées, mais il l'est de manière plus robuste quand les amers sont *identifiés et reconnus*, car il ne dépend pas des estimées de positions courantes – qui peuvent parfois être inconsistantes.

Pour ces processus de perception, le choix du modèle des amers joue un rôle crucial. Un bon amer doit être remarquable dans les données, aisément détecté, sa position par rapport au capteur doit pouvoir être mesurée, et en particulier il doit être aisé à mettre en correspondance lorsqu'il est perçu de positions différentes. On peut souvent distinguer deux parties dans le modèle d'un amer : la partie dédie à l'estimation (variables géométriques qui définissent sa position), et la partie dédiée à la mise en correspondance, qui contient les informations nécessaires à ce processus. Par exemple, la plupart des approches du problème SLAM basées sur la vision exploitent des points d'intérêts (points de Harris ou points "SIFT", que ce soit par stéréovision [8, 14] ou vision monoculaire [3]). Ces points d'intérêt ont toutes les propriétés requises pour constituer de bons amers : ils correspondent à des points 3D dans l'environnement, auxquels sont associés des informations visuelles utiles pour leur mise en correspondance.

Mais le modèle de l'environnement constitué par ces points est pauvre, et n'est exploitable que pour la résolution du problème SLAM. Il y a un intérêt à définir des modèles d'amer plus riches, d'une part pour faciliter le processus de mise en correspondance, et d'autre part pour construire des modèles d'environnement plus représentatifs de la structure de l'environnement, qui pourront alors être exploitables par d'autres processus (détermination d'espace libre, calculs de visibilité...). Les très récentes contributions au SLAM par vision monoculaire qui exploitent des segments vont dans cette direction [5, 16, 9].

Dans cet article, nous proposons un modèle d'amers constitué de facettes planes détectées par stéréovision. Basé sur des points d'intérêt, ce modèle enrichit leur description par 6 paramètres de positionnement et des informations de texture stockées dans une imagette : cette description donne une meilleure observabilité de la position du robot par la perception d'un faible nombre d'amers, et facilite la mise en correspondance des amers détectés de points de vue différents. Dans l'environnement, une facette correspond à une zone plane de petite taille : d'une part de tels éléments sont nombreux, même dans un environnement naturel, et d'autre part, il est possible de prédire la forme et l'apparence d'une facette d'un point de vue donné, ce qui facilite leur mise en correspondance.

L'article est organisé comme suit : la section 2 présente ce modèle et le processus de détection dans une paire d'images stéréoscopiques. Ensuite, la section 3 présente les algorithmes de suivi et de mise en correspondance des facettes. Enfin, des résultats de SLAM basés sur ces amers sont présentés en section 4.

2 Modélisation et détection des facettes

2.1 Définition d'un facette

Repère associé Ce que nous appellons une "facettes" est en fait un ensemble de propriétés géométriques qui représentent le repère associé à une facette, complétées par des informations provenant du signal (comme la texture). La figure 1 présente un exemple de facettes extraites à partir d'une paire d'images stéréoscopiques.
Les paramètres géométriques d'une facette comporte le vecteur d'origine ainsi que les trois angles d'Euler.

Dimension Dans le but de simplifier les calculs, nous nous limitons à des facettes de dimension physique fixes, de l'ordre de dix centimètre sur dix.

Texture l'information géométrique est une donnée insuffisante pour effectuer l'appariement des facettes, elle est donc complétée par une information sur le signal, à savoir la texture, codé dans une imagette, que nous avons choisie de taille fixe afin de faciliter les comparaisons.

2.2 Extraction des facettes

Détection de points d'intérêt Les points d'intérêts sont des pixels d'une image qui doivent être reconnaissables et dont la détection est reproductible. Une facette peut-être associée à un point de Harris [6], ou a des points invariants par changement d'échelle [13, 11, 7]. Ces derniers offrent une meilleure répétabilité, au prix d'un temps d'extraction plus long qu'un détecteur de Harris.

Estimation d'homographie On peut exploiter un algorithme de stéréovision dense pour estimer la normale associée à un point d'intérêt et pour déterminer si la surface qui le supporte est plane : il suffit de déterminer par une méthode des moindre carrés si les points 3D au voisinage du point d'intérêt considéré correspondent à un plan. Mais les algorithmes de stéréovision dense temps réel donnent des résultats assez bruités, qui rendent l'estimation de la normale très instable.
Une méthode basée sur l'estimation d'une homographie est plus fiable. Les projections d'un plan sur des images prises de deux point de vue différents sont liés par une homographie $s * H$: soit un plan P et deux images I_1 et I_2, pour tout point de P, les coordonnés des pixels correspondants dans I_1 et I_2 sont liées par l'homographie $s * H$, où H est une matrice de dimension 3x3, et s est un coefficient de valeur quelconque, généralement choisi de tel sorte que $(s * H)(3, 3) = 1.0$. Donc, deux images I_1^p et I_2^p extraites respectivement de I_1 et I_2 correspondent à une zone plane si il existe une matrice H telle que

$$H * I_2^p = I_1^p \tag{1}$$

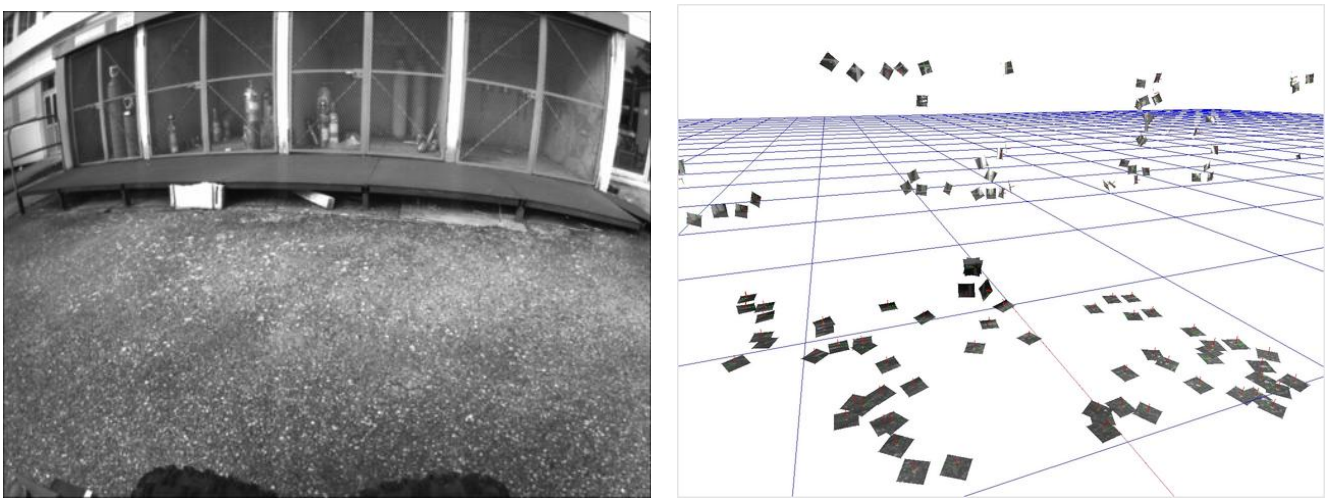

FIG. 1 – La figure de droite montre les facettes extraites à partir d'une paire stéréoscopique pour la scène représentée sur la figure de gauche.

Les algorithmes d'alignements qui permettent d'obtenir la valeur de H sont des procédures d'optimisation dont le but est de minimiser :

$$E = H * I_2^p - I_1^p - (\mu(H * I_2^p) - \mu(I_1^p)) \qquad (2)$$

Où $\mu(H * I_2^p)$ et $\mu(I_1^p)$ sont la moyenne des pixels de $\mu(H * I_2^p)$ and $\mu(I_1^p)$, et permettent de réduire l'influence du changement d'illumination entre les deux images.

Une comparaison des différents algorithmes d'alignements peut être trouvée dans [2], où est aussi proposée une nouvelle méthode pour l'estimation de l'homographie, "Inverse Compositional Estimation" (ICE). [12] introduit la méthode "Efficient Second-order Minimization" (ESM) permet pour le suivi de plan de grandes dimensions à l'aide d'une estimation d'homographie.

Les deux méthodes sont capables, sur de petites surfaces, de trouver une estimation de l'homographie qui est soit très bonne, soit complètement fausse. Dans les évaluations que nous avons menées, il s'est révélé que lorsque l'estimation était fausse, la normale obtenue avait un caractère aléatoire, ce qui permet d'éliminer les mauvaises facettes à posteriori lors de leur suivi dans les séquences (voir 3.4). Nos évaluations ont montré qu'un plus faible nombre de facettes était éliminé avec ICE, ce qui semble indiquer que pour des plans de petites dimensions (de l'ordre de 20 pixels par 20 pixels), ICE a un meilleur comportement que ESM – contrairement à ce qui est observé pour des plans de grandes dimensions dans [2] et [12].

Estimation de la normale Une fois que l'homographie est estimée, la normale peut être calculée en utilisant les paramètres intrinsèques de la caméra, par exemple en générant trois points du plan, et en estimant les paramètres de ce plan. Les trois points du plans sont calculés en utilisant l'estimation de l'homographie.

Complétion de la base Une base nécessite la connaissance de trois vecteurs, mais en pratique, il est nécessaire d'en extraire uniquement deux, le troisième étant déterminé à l'aide du produit vectoriel. Ainsi le premier vecteur est obtenu par le calcul de la normale. Un second vecteur est déduit du signal, en calculant l'orientation de l'image autour du point d'intérêt.

Pour estimer l'orientation, le gradient est calculé pour chaque pixel P d'une fenêtre W autour du point d'intérêt IP, en utilisant deux masques de Sobel, un vertical et un horizontal. Ainsi, un angle est connu en chaque point de la fenêtre, et l'orientation finale est estimée par la somme pondérée :

$$Orientation = \frac{\sum_{P \in W} w\left(d(P, IP) * atan\left(\frac{Gy(P)}{Gx(P)}\right)\right)}{\sum_{P \in W} w\left(d(P, IP)\right)} \qquad (3)$$

Où $d(P, IP)$ est la distance entre le pixel P et le point d'intérêt IP et où $w(x)$ est une gaussienne.

Cependant, malgré la diminution de la sensibilité au bruit et au changement de point de vue obtenue par l'utilisation de la somme pondérée, l'orientation est soit très stable (dans la plus part des cas) soit aléatoire. De la même façon qu'à la suite du calcul de l'homographie, les facettes dont l'orientation n'est pas stable peuvent être éliminées à posteriori lors de leur suivi dans les séquences (voir 3.4). Notons que cette orientation correspond en fait au troisième angle d'Euler qui décrit l'orientation de la facette.

2.3 Texture

La texture d'une facette est interpolée depuis l'image, en utilisant les informations géométriques. L'interpolation de la texture de la facette F est effectuée en calculant à quel point 3D $P \in F$ dans l'environnement correspond un pixel p_t de la facette, et en projetant ce point P sur la caméra en un pixel p_c.

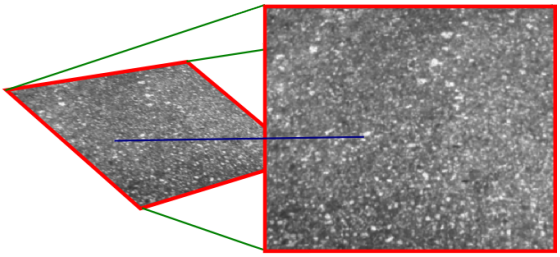

FIG. 2 – Interpolation de la texture d'une facette. La ligne bleue montre comment un pixel de l'image réel est associé à un pixel de la texture.

Soit $\mathcal{P}_{Camera}$, la projection d'un point de l'environnement sur la caméra. Soit OF le vecteur de l'origine du monde vers l'origine de la facette F et soit v et w, les vecteurs de la base de la facette, respectivement calculés à partir de l'orientation et par complétion de la base. Si on considère que les pixels de la texture sont indexés à partir du centre par i et j, l'équation suivante donne la valeur des pixels de la texture et est schématisé par la figure 2 :

$$p_t(i,j) = p_c(\mathcal{P}_{Camera}(OF + i * v + j * w)) \quad (4)$$

Cette procédure a l'avantage de faciliter l'appariement ultérieur des facettes. En effet, après interpolation, la texture de la facette est stockée en mémoire avec l'apparence qu'elle aurait si la caméra avait observé la facette dans les conditions idéales, c'est à dire avec sa normale confondue avec l'axe optique de la caméra, et le deuxième vecteur de la base parallèle avec l'horizon de la caméra. Ce qui signifie que lors de la phase d'appariement, une comparaison pixel à pixel permet d'obtenir un score de corrélation entre la texture observée et la texture mémorisée.

$$Zncc = \frac{\sum p_{mem}(i,j) * p_{obsv}(i,j) - \mu(mem) * \mu(obsv)}{\sigma(mem) * \sigma(obsv)} \quad (5)$$

Où p_{mem}, $\mu(mem)$ et $\sigma(mem)$ désignent un pixel, la moyenne et l'écart type de la texture mémorisée, et p_{obsv}, $\mu(obsv)$ et $\sigma(obsv)$ désignent un pixel, la moyenne et l'écart type de la texture observée.
En théorie, lorsque l'appariement est parfait, on a $Zncc = 1.0$. Dans la pratique le score est utilisée pour trouver la facette mémorisée la plus proche de la facette observée.

2.4 Modèle d'erreur

Le modèle d'erreur pour la représentation réduite des facettes est composé des covariances sur l'estimation de la position de l'origine de la facette et des covariances sur son orientation. L'origine de la facette et son orientation étant déterminées par des processus indépendants (triangulation et estimation d'homographie), les covariances entre ces éléments sont nulles. De même, la connaissance du troisième angle d'Euler est obtenue indépendemment des deux premiers angles. Ce qui donne une matrice de covariances de la forme suivante :

$$\begin{bmatrix} M_{stereo} & 0 & 0 & 0 \\ 0 & \sigma_{yaw} & \sigma_{yaw/pitch} & 0 \\ 0 & \sigma_{yaw/pitch} & \sigma_{pitch} & 0 \\ 0 & 0 & 0 & \sigma_{roll} \end{bmatrix} \quad (6)$$

Où M_{stereo} est le modèle d'erreur classique de la stéréovision [17]. Les valeurs des variances et covariances des angles ont été déterminés empiriquement : $\sigma_{yaw} = \sigma_{pitch} = \sigma_{roll} = 0.1$ et $\sigma_{yaw/pitch} = 0.01$.

3 Mise en correspondance de facettes

3.1 Algorithme général

La méthode utilisée pour l'appariement de facettes est une extension à la troisième dimension d'un algorithme d'appariement de points d'intérêt décrit dans [8]. Un appariement qui repose uniquement sur les informations du signal n'est pas sûr, l'idée est donc d'utiliser les informations géométriques qui lient les facettes entre elles lors de l'appariement.
Soit un ensemble $\mathcal{F}_1$ de facettes que l'on cherche à associer à l'ensemble $\mathcal{F}_2$ de facettes. L'algorithme consiste à chercher un premier appariement entre une facette de $\mathcal{F}_1$ et une de $\mathcal{F}_2$, qui va permet d'obtenir une transformation géométrique $\mathcal{T}_{1\to2}(f)$ qui servira à focaliser la recherche d'autres appariements.

1. Soit $f_1 \in \mathcal{F}_1$, $f_2 \in \mathcal{F}_2$ est la facette dont la texture est la plus proche de f_1, c'est à dire qui maximise $CompareTexture(f_1, f)\ \forall f \in \mathcal{F}_2$ où $CompareTexture$ est une fonction de comparaison de textures (par exemple le score ZNCC)
2. Ce premier appariement permet de calculer la transformation géométrique $\mathcal{T}_{1\to2}(f)$ telle que :
$$\mathcal{T}_{1\to2}(f_1) = f_2 \quad (7)$$
3. $\forall f'_1 \in \mathcal{F}_1$, si il existe $f'_2 \in \mathcal{F}_2$ qui satisfait les deux conditions :
$$\mathcal{T}_{1\to2}(f'_1) = f'_2 \quad (8)$$
$$CompareTexture(f'_1, f'_2) > \mathcal{T}_{texture} \quad (9)$$
Alors le couple (f'_1, f'_2) est un appariement.

La figure 3 montre deux exemples de résultats d'appariement de facettes.

3.2 Suivi de facettes

Un des avantages des facettes est la possibilité de les reprojeter et de simuler comment elles seraient observées par une caméra depuis n'importe quel point de vue. En particulier, si la transformation est connue de manière certaine, il devient très facile de comparer l'image observée à une

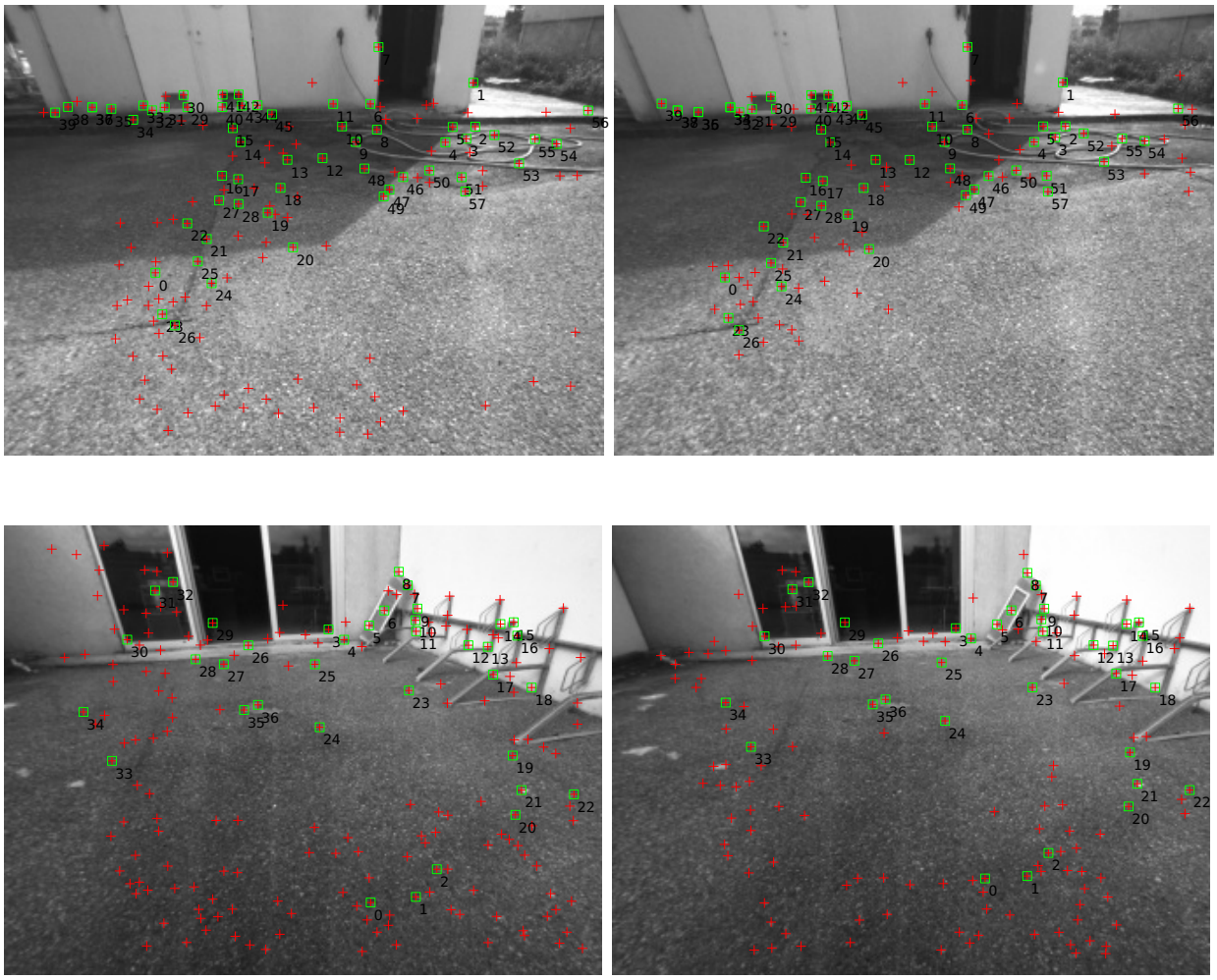

FIG. 3 – Deux résultats d'appariements de facettes.

imagette mémorisée. Mais ceci est d'un faible intérêt dans le cas du SLAM, où le changement de point de vue n'est pas bien connu – notamment lorsque le robot re-visite une scène perçue après avoir effectué un grand déplacement. Mais entre les instants t et $t+1$, le changement de point de vue est suffisamment faible pour que l'on puisse utiliser la transformation géométrique $T_{t\rightarrow(t+1)}$ donnée par l'odométrie, afin de prédire la position des facettes et de tenter de les suivre.
Soient $\mathcal{I}p(I^l_{t+1})$ et $\mathcal{I}p(I^r_{t+1})$ la liste de points d'intérêt détectés à l'instant $t+1$, respectivement dans les images gauche I^l_{t+1} et droite I^r_{t+1}, et $\mathcal{F}(t)$ l'ensemble des facettes détectées à l'instant t.

1. $\forall f \in \mathcal{F}(t)$, on calcule la projection P^l_f de f sur l'image I^l_{t+1}
2. soit $Candidats$ la liste des points d'intérêts qui sont situés à proximité de la position estimée de la facette dans l'image :

$$Candidats = Ip^l \in \mathcal{I}p(I^l_2)/|Ip - P_F| < \epsilon \quad (10)$$

   En utilisant, l'estimation du mouvement $T_{t\rightarrow(t+1)}(base)$, il est possible d'estimer les paramètres de la facette, et en particulier d'utiliser l'estimation de la normale afin de calculer une estimation de la texture pour chaque point de $Candidats$ de la même manière que dans la section 2.3. Soit $I^l_p(F) \in Candidats$ le point dont la texture estimée est la plus proche de celle de la facette.
3. la même méthode est utilisée pour trouver $Ip^r(F)$ dans l'image de droite en rajoutant la contrainte que les deux points d'intérêt doivent se trouver sur la même épipolaire, contrainte provenant de la phase de rectification des images en stéréovision
4. en utilisant le couple (Ip^l, Ip^r), les paramètres de la facette f_{suivi} sont estimés comme présenté dans la section 2, ceci afin de vérifier que $f_{suivi} = T_{t\rightarrow(t+1)}(f)$

D'autres méthodes de suivi (tel que [15] ou [12]) auraient pu être utilisées, mais cette méthode offre l'avantage de permettre un contrôle direct sur les paramètres des facettes, et ainsi d'offrir des possibilités de mise à jour (voir 3.3) ou d'élimination (voir 3.4).
Le principal intérêt de l'algorithme de suivi de facettes est sa vitesse, comparée à l'algorithme général d'appariement de facette : en effet le suivi de facette prend de l'ordre de 300ms, alors que la génération de l'ensemble des facettes d'une image nécessite 500ms et que l'appariement sur une base similaire à celle utilisée pour le suivi nécessite environ une seconde[1].

3.3 Mise à jour des facettes

Une fois qu'une facette a été suivie ou appariée, il est intéressant d'utiliser certaines des nouvelles informations pour les mettre à jour. En particulier, la texture peut-être mise à jour si la facette a été vue d'une "meilleure position", par exemple, si la nouvelle observation est plus proche de la facette que la précédente, et selon un angle entre la normale de la facette et l'axe de la caméra plus faible.

3.4 Élimination des facettes non stables

Après l'utilisation des algorithmes d'appariement ou de suivi, certaines facettes ne seront pas appariées, ou leur observation apparaîtra inconsistante par rapport à l'ensemble des observations de facettes. Ces facettes ont soit une répétabilité trop faible (voir 2.2), ou une normale ou une orientation (voir 2.2) non fiables.
Ces erreurs sont provoquées par différents phénomènes : ainsi, un voisinage du point d'intérêt faiblement texturé peut mener à une estimation de l'homographie erronée (cas d'un point noir sur un mur blanc par exemple, ce qui donne un bon point d'intérêt).

4 Application au SLAM

4.1 Groupement de facettes

Afin de réduire la taille du filtre de Kalman exploité dans le SLAM, les facettes sont regroupées de manière à ne constituer qu'un seul amer. Elles sont regroupées en fonction de leur proximité géométrique, et de telle sorte que la densité du groupe est plus importante au centre de l'amer. La raison est que les facettes les plus proches du centre de l'amer permettent d'estimer plus précisément la position de l'amer. En effet, une erreur sur l'estimation des angles du repère de la facette introduit une erreur d'autant plus importante sur la localisation de l'amer que la facette est éloignée de son centre.
Après la phase de détection, nous disposons d'un ensemble $\mathcal{F}$ de facettes.

1. Soit $f^i \in \mathcal{F}$, soit G^i l'ensemble de facettes qui sont situées autour de f^i :

$$G^i = f \in \mathcal{F}/d(f, f^i) < r \quad (11)$$

   où $d(f_1, f_2)$ est la distance entre deux facettes f_1 et f_2 et r est le rayon d'un amer
2. A partir de ce premier groupe de facettes, le centre de l'amer C est estimé en calculant le barycentre des facettes :

$$OC = \frac{\sum_{f\in=G^i} w_f * f}{\sum_{f\in=G^i} w_f} \quad (12)$$

   La pondération w_f est utilisée pour favoriser les facettes qui sont supposées de meilleure qualité, en l'occurence nous avons choisi de favoriser les facettes dont la normale est parallèle à la caméra, c'est à dire :

$$w_i = < axe_{camera} | n_f > \quad (13)$$

3. Le groupe de facettes formant l'amer est l'ensemble :

$$f \in \mathcal{F}/d(f/OC) < r \quad (14)$$

[1] Temps mesurés sur un processeur Intel core Duo à 1.8 GHz.

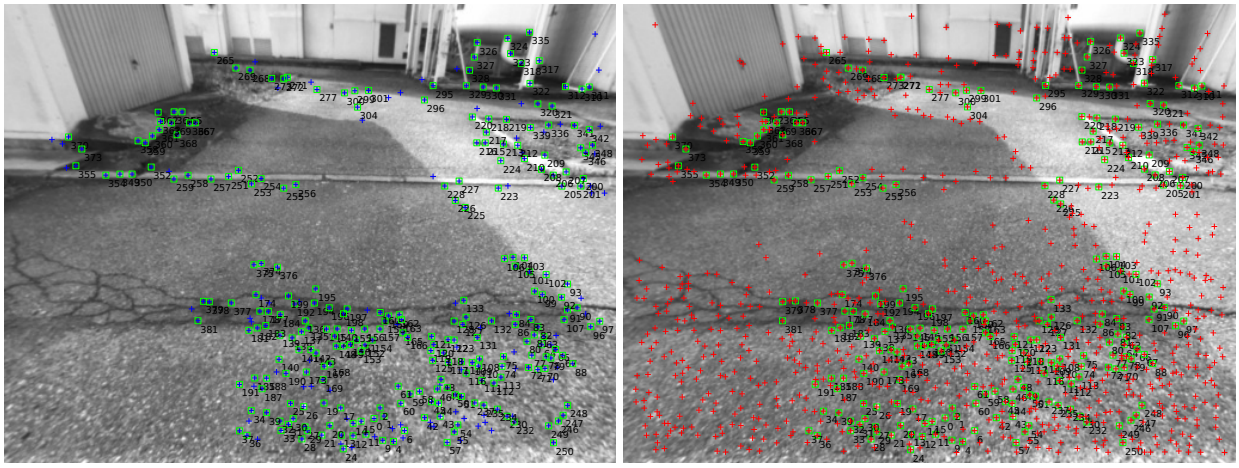

FIG. 4 – Facettes suivies dans deux paires d'images consécutives. Les "+" bleus indiquent les facettes détectées, les points rouges sont les points des Harris, et les carrés verts représentent les facettes suivies.

Les étapes 2 et 3 pourraient être répétées plusieurs fois jusqu'à ce que le groupe de facettes obtenues deviennent stables, mais lors de nos expérimentations, nous avons constaté que le groupe changeait peu lors des itérations suivantes. La figure 5 montre le résultat du regroupage de facettes.

4.2 Exploitation pour le SLAM

Soit $\mathcal{A}$ l'ensemble des amers dans l'environnement, $\mathcal{F}_t$ l'ensemble des facettes se trouvant dans le champ de vue de la caméra à l'instant t, c'est-à-dire $\mathcal{F}_t$ contient les facettes qui ont été suivi entre l'instant $t-1$ et t, les facettes détectées à l'instant t et les facettes qui n'ont pas été observées à l'instant t, qui se trouve dans le champs de vision de la caméra mais qui ont vraissemblablement été occulté à l'instant t.
Soit M_{robot} l'estimation du mouvement du robot, obtenue par exemple à l'aide de l'odométrie.

1. L'ensemble des facettes suivies $\mathcal{F}_{tr}$ est déterminé en suivant l'algorithme décrit à la section 3.2 et de l'estimation M_{robot} du mouvement et de $\mathcal{F}_{t-1}$, ce qui permet d'en déduire un ensemble d'observations $obsv$ d'amers
2. si le ratio de facettes qui sont suivies descend en dessous d'un certain seuil :

$$\frac{|\mathcal{F}_{tr}|}{|\mathcal{F}_{t-1}|} < th_{facettestrackees} \tag{15}$$

   alors il est nécessaire de lancer une nouvelle phase de détection :

   (a) l'algorithme de détection de facettes décrit en section 2.2 permet d'obtenir un ensemble $\mathcal{F}_{detect}$ de facettes

   (b) l'algorithme de mise en correspondance de la section 3.1 est utilisé afin de déterminer si l'un des amers de $\mathcal{A}$ vient de réapparaître dans l'environnement, si c'est le cas l'ensemble d'observations $obsv$ est complété avec la nouvelle observation de l'amer, et les facettes qui font parties de cet amer sont retirées de $\mathcal{F}_{detect}$

   (c) l'algorithme de regroupement de facettes de la section 4.1 permet la création de nouveaux amers $newamers$
3. les ensembles $obsv$ et $newamers$ sont utilisés pour la mise à jour du filtre de Kalman
4. l'ensemble $\mathcal{F}_t$ est déterminé en retirant les facettes que l'on ne peut plus suivre et en ajoutant les facettes nouvellement détectées :

$$\mathcal{F}_t = (\mathcal{F}_t \cup \mathcal{F}_{detect}) \setminus \mathcal{F}_{untrackable} \tag{16}$$

   où $\mathcal{F}_{untrackable}$ est le sous-ensemble de facettes de $\mathcal{F}_{t-1}$ qui ne sont plus dans le champ de vue de la caméra

La figure 6 montre deux trajectoires, une avec la détection du bouclage et l'une où l'algorithme de mise en correspondance a été désactivée.

5 Perspectives

Certaines limitations réduisent l'intérêt de la méthode :
– bien que les facettes soient repérables depuis différents points de vue, le fait qu'elles soient centrées sur des points d'intérêt les rend sensibles à leur répétabilité par changement d'échelle et d'orientation
– en l'absence d'heuristique qui permette de réduire l'espace de recherche, l'appariement des facettes est un processus coûteux en temps de calcul. L'heuristique utilisée

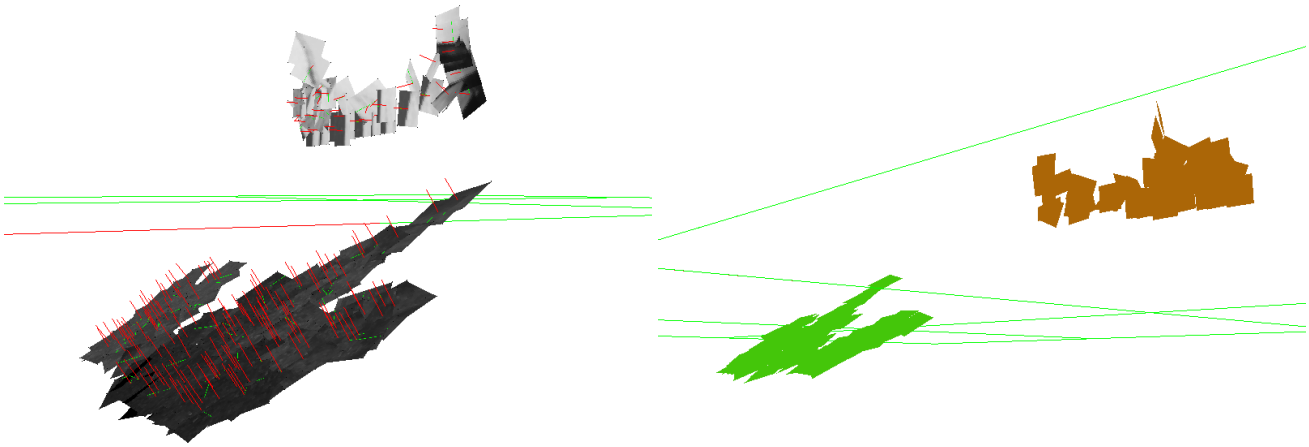

FIG. 5 – L'image de gauche montre les facettes qui ont été extraites dans l'environnement, l'image de droite montre les deux groupes de facettes qui ont été générés pour une insertion dans le filtre de Kalman.

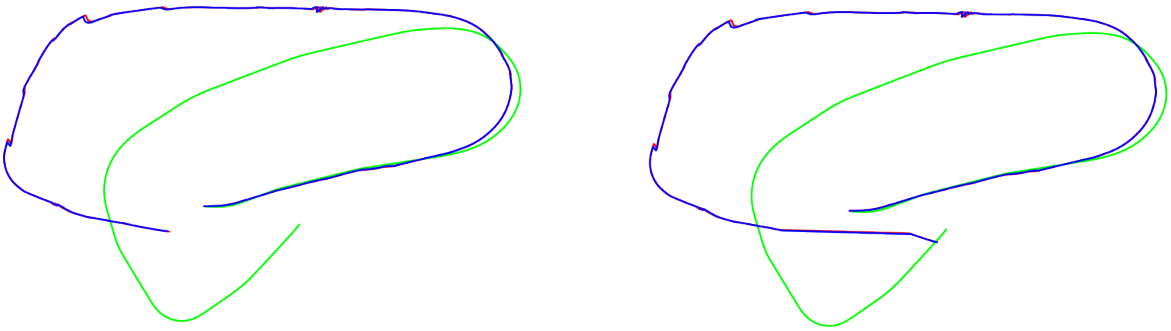

FIG. 6 – Sur les deux figures, le robot a parcouru 59m, la trajectoire verte est le résultat de l'odométrie, la trajectoire bleue la position du robot après la phase de prédiction, et la trajectoire rouge, qui est quasiment superposée à la trajectoire bleue, est la trajectoire après utilisation des observations. La figure de gauche montre la trajectoire en l'absence de l'algorithme d'appariement, c'est à dire qu'il n'y a pas de détection de fermeture de boucle. La figure de droite montre la trajectoire avec utilisation de l'algorithme de mise en correspondance.

ici est basée sur l'estimation de la position : il sera nécessaire d'envisager d'autres méthodes, notamment lorsque la position est inconnue ou trop imprécise.

De plus, cette représentation de l'environnement, bien que plus riche que les modèles utilisés jusqu'à présent en SLAM par vision, est loin d'exploiter l'ensemble des informations d'une paire d'images stéréoscopiques. Dans le but de limiter cette perte d'information, nous avons pris le parti de supposer que la transformation entre deux facettes observées à un instant donné était rigide (section 4.2), et que les deux facettes pouvaient donc être insérées dans un même amer sans que cela pose problème. Il serait intéressant de pouvoir aussi re-estimer les positions relatives des facettes au sein d'un amer en associant par exemple un filtre de Kalman par amer, en utilisant des méthodes dites "Divide and Conquer" [10].

Cependant, nous avons pu montré ce que pouvait apporté une modélisation enrichie de l'environnement, par exemple, à l'aide de facettes. En effet, de tels modèles permettent d'obtenir une carte d'amers plus compact, avec laquelle il est possible de détecter une fermetture de boucle, ce qui ouvre la voie à du SLAM sur des distances plus importantes avec des cartes de taille plus grande.

Références

[1] T. Bailey and H. Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM) : Part II - State of the Art. *Robotics and Automation Magazine*, September 2006.

[2] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition*, December 2001.

[3] A. Davison. Simultaneous localisation and mapping with a single camera. In *9th International Conference on Computer Vision, Nice (France)*, October 2003.

[4] H. Durrant-Whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM) : Part I - The Essential Algorithms. *Robotics and Automation Magazine*, June 2006.

[5] E. Eade and T. Drummond. Edge landmarks in monocular slam. In *British Machine Vision Conference, Edinburgh (UK)*, Sep. 2006.

[6] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.

[7] Luc Van Gool Herbert Bay, Tinne Tuytelaars. Surf : Speeded up robust features. In *9th European Conference on Computer Vision*, 2006.

[8] I-K. Jung and S. Lacroix. A robust interest point matching algorithm. In *8th International Conference on Computer Vision, Vancouver (Canada)*, July 2001.

[9] T. Lemaire and S. Lacroix. Monocular-vision based SLAM using line segments. In *IEEE International Conference on Robotics and Automation, Roma (Italy)*, April 2007.

[10] J.D. TardÃşs L.M. Paz, P. Jensfelt and J. Neira. Ekf slam updates in o(n) with divide and conquer slam. In *IEEE Int. Conf. Robotics and Automation*, Rome, Italy, april 2007.

[11] D. Lowe. Distinctive features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2) :91–110, 2004.

[12] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, April 2004.

[13] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1) :63–86, 2004.

[14] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, 21(8) :735–758, 2002.

[15] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle (USA), June 1994.

[16] P. Smith, I. Reid, and A. Davison. Real-time monocular slam with straight lines. In *British Machine Vision Conference, Edinburgh (UK)*, Sep. 2006.

[17] Y. Xiong and L. Matthies. Error analysis of a real time stereo system. In *IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico*, pages 1087–1093, June 1997.

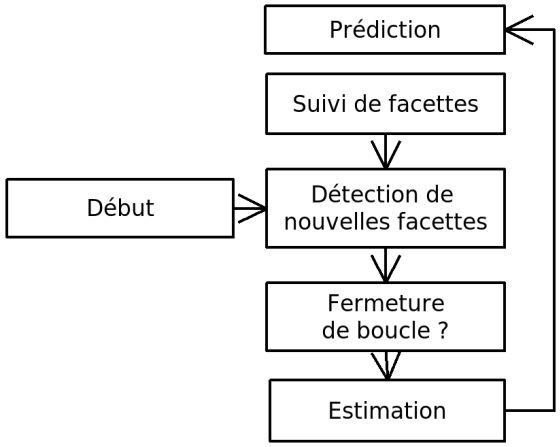

FIG. 7 – Les différentes étapes de l'algorithme de SLAM