

Rapport final

Sujet TP3

Cartographie et localisation simultanées avec la vision robotique

**Encadrement :** Alain Boucher (IFI), Serge Stinckwich (MSI)

**Etudiant :** Vu Hong Thuan

Hanoï – juin 2010

Table de matière

[1 Introduction 6](#_Toc265329043)

[1.1 Le SLAM problème 6](#_Toc265329044)

[1.2 Taxonomie du SLAM problème 6](#_Toc265329045)

[1.3 Le SLAM visuel 7](#_Toc265329046)

[1.4 Le travail de TPE 7](#_Toc265329047)

[2 L'état de l'art 8](#_Toc265329048)

[2.1 Des approches pour le problème SLAM 8](#_Toc265329049)

[2.2 Les méthodes probabilistes 8](#_Toc265329050)

[2.2.1 EKF - Extended Kalma Filters 8](#_Toc265329051)

[2.2.2 Inférence de Bayes 8](#_Toc265329052)

[2.2.3 EM (Estimation et maximisation) 8](#_Toc265329053)

[2.2.4 Relaxation et « Multilevel Relaxation » 8](#_Toc265329054)

[2.2.5 Filtreur de particule 9](#_Toc265329055)

[2.3 SLAM visuel 10](#_Toc265329056)

[2.3.1 Point d’intérêt 10](#_Toc265329057)

[2.3.1.1 Coin de Harris 10](#_Toc265329058)

[2.3.1.2 SIFT - Scale-invariant feature transform 10](#_Toc265329059)

[2.3.1.3 Coin de Kanade – Tomasi 11](#_Toc265329060)

[2.3.2 Odométrie visuelle 11](#_Toc265329061)

[2.3.3 Gamma-SLAM 12](#_Toc265329062)

[2.3.4 Mini SLAM 12](#_Toc265329063)

[2.3.5 σSLAM 13](#_Toc265329064)

[2.3.6 MonoSLAM 13](#_Toc265329065)

[2.4 SLAM 3D 14](#_Toc265329066)

[3 Le travail pratique 14](#_Toc265329067)

[3.1 Chercher les logiciels 14](#_Toc265329068)

[3.1.1 Openslam 14](#_Toc265329069)

[3.1.2 Mrpt 14](#_Toc265329070)

[3.1.3 ROS 15](#_Toc265329071)

[3.1.4 Scene 15](#_Toc265329072)

[3.2 Choix de logiciel 15](#_Toc265329073)

[3.3 Choix de caméra 15](#_Toc265329074)

[3.4 Camera calibration 15](#_Toc265329075)

[3.5 Sélectionner les données 17](#_Toc265329076)

[4 Programme 18](#_Toc265329077)

[4.1 La configuration 18](#_Toc265329078)

[4.2 L’interface graphique 19](#_Toc265329079)

[4.3 Structure 22](#_Toc265329080)

[4.3.1 La structure de Scene 22](#_Toc265329081)

[4.3.2 La structure de monoslam 23](#_Toc265329082)

[4.4 La base de calcule 24](#_Toc265329083)

[4.5 Simulation 24](#_Toc265329084)

[4.6 Test et vérification 24](#_Toc265329085)

[4.6.1 Simple scénario 24](#_Toc265329086)

[4.6.2 La cartographie 26](#_Toc265329087)

[4.6.3 La localisation 27](#_Toc265329088)

[4.6.4 La fermeture de trajectoire 28](#_Toc265329089)

[4.6.5 Le problème de robot kidnappé (kidnapped robot problem) 28](#_Toc265329090)

[5 Résultat 28](#_Toc265329091)

[5.1 Avantages et inconvénients 28](#_Toc265329092)

[5.1.1 Avantages 28](#_Toc265329093)

[5.1.2 Inconvénient 29](#_Toc265329094)

[6 Expérience 29](#_Toc265329095)

[6.1 Difficultés 29](#_Toc265329096)

[7 Conclusion et Perspectives 29](#_Toc265329097)

[7.1 Contribution 29](#_Toc265329098)

[7.2 Perspectives 30](#_Toc265329099)

[7.2.1 Reconstruction la structure 3D de l’environnement 30](#_Toc265329100)

[7.2.2 Reconnaissance des lignes dans l’environnement 30](#_Toc265329101)

[7.2.3 Comparer avec l’autre méthode de SLAM visuel 30](#_Toc265329102)

[7.2.4 La fermeture de boucle 31](#_Toc265329103)

[7.2.5 Comparer l’odométrie visuelle et odométrie mécanique 31](#_Toc265329104)

[7.3 Conclusion 31](#_Toc265329105)

[8 Référence 31](#_Toc265329106)

Table de figure

Remerciement

Je souhaite tout d’abord remercier profondément en particulier Messieurs Alain Boucher, Serge Stinckwich qui m’a encadré pendant ce TPE, pour m’avoir toujours fait confiance, me laissant conduire mes travaux à mon aise, pour leur encadrement et leur soutien sans failles, ainsi que leurs encouragements, pour avoir su prodiguer les conseils pertinents afin d’orienter mes recherches, et pour avoir toujours pu me permettre de travailler dans les meilleurs conditions.

Enfin, il m’est impossible de remercier assez ma famille, qui m’a toujours encouragé et soutenu dans mes choix, me permettant de réaliser tout ce en quoi j’ai toujours cru, et pour avoir toujours été à mes côtés.

Abstrait

Titre : **Cartographie et localisation simultanées avec la vision robotique**

SLAM (Simutanous Localisation And Mapping) ou CML (Concurrent Mapping and Localisation). Le SLAM problème est un des problèmes les plus recherchés dans la robotique. Mais il y a beaucoup de défis dans ce problème. Il n’y a pas d’algorithme les meilleurs pour tous les SLAM problèmes. Le choix le meilleur d’algorithme pour un spécifié SLAM problème dépend de facteurs en particuliers de chaque problème.

SLA M visuel est un aspect de recherche dans SLAM. Dans ce cas, on utilise caméra comme percepteurs de l’environnement.

Dans le cadre de mon TPE, je présente le problème SLAM, en particulière SLAM visuel. Ensuite, je résume les méthodes SLAM en général, et SLAM visuel en particulière.

En fin, je présente mes travaux pratiques dans ce TPE en utilisant la méthode monoslam, les résultats obtenus et mes expériences autour ces travaux.

Mots clés : SLAM visuel, monoslam

# Introduction

## Le SLAM problème

Un robot se situe dans un environnement. Le robot ne connait pas l’environnement. Son travail est découvrir l’environnement. Pour finir mon travail, il doit faire une carte d’environnement autour lui-même et se localiser dans cet environnement en même temps.

Cette discipline de recherche s’appelle **SLAM (Simutanous Localisation And Mapping)** ou **CML (Concurrent Mapping and Localisation)**. Le SLAM problème est un des problèmes les plus recherchés dans la robotique.

Un robot contient des composants pour découvrir l’environnement. Ce sont le moteur de motion pour se déplacer dans l’environnement, et des capteurs pour détecter des fonctions de l’environnement, et une antenne de WiFi pour recevoir et transmettre de signal de communication, et un système d’exploitation pour procéder des calculs nécessaires. La restreinte de mémoire de robot, les erreurs dans les mesures, la restreinte de la puissante de robot... On doit faire face à les restreintes dans des recherches sur SLAM.

Un robot est déplacé dans un environnement qu’il ne connaît pas. Il commence à se déplacer dans l’environnement en utilisant ses capteurs pour détecter des fonctions d’environnement. Ces activités se répètent. Les données qu’il a perçues de l’environnement sont manipulées pour faire une carte de l’environnement et se localiser dans cet environnement. Et comment est-ce que le robot manipule les donnés perçues pour faire une localisation et cartographie ? C’est SLAM problème.

Le SLAM problème est un des problèmes les plus recherchés dans la robotique. Mais il y a beaucoup de défis dans ce problème.

Il n’y a pas d’algorithme les meilleurs pour tous les SLAM problèmes. Le choix le meilleur d’algorithme pour un spécifié SLAM problème dépend de facteurs en particuliers de ce problème.

## Taxonomie du SLAM problème

Il y a beaucoup de méthode de classifier le SLAM problème

1. Volumétrique vs base-de-fonction. Un SLAM problème est volumétrique si sa carte a une résolution haute, c’est-à-dire trop détaillé en pixel. Par contre, un SLAM problème base-de-fonction si sa carte est un semble de fonctions (objets, choses) clairsemées.
2. Topologique et métrique. Un SLAM problème est topologique si l’environnement est représenté par une liste de distincts objets et des qualitatives relations entre eux. Un SLAM problème est métrique si des locations des objets dans l’environnement sont représentées par des valeurs métriques.
3. Connue ou inconnue correspondance. L’identification la correspondance des objets perçus dans l’environnement est un problème difficile. Un SLAM problème a la connue correspondance si on assume que le robot a les identifications des objets dans l’environnement. Un SLAM problème a l'inconnue correspondance si le robot ne connaît pas les identifications des objets dans cet environnement. Dans ce cas-là, le robot doit identifier une correspondance des objets perçus dans cet environnement.
4. Statique et dynamique. Un SLAM problème a l’environnement statique si l’environnement ne se change pas. Un SLAM problème a l’environnement dynamique si l’environnement se change beaucoup.
5. Petite et large incertitude. Un SLAM problème a une petite incertitude s’il y a moins d’erreurs d’estimation de sa location. Un SLAM problème a une large incertitude s’il y a plus d’erreurs d’estimation de sa location.
6. Seul vs multiple robot(s). Un SLAM problème a un seul robot. C’est-à-dire, le robot doit découvrir l’environnement indépendamment. Un SLAM problème a des multiples robots. C’est-à-dire, les robots doivent collaborer pour découvrir l’environnement.

Ce sont des façons de classifier les SLAM problèmes. Dans un problème spécifié, on doit fait attention de facteurs avant pour choisir l’algorithme rationnel.

Les algorithmes de SLAM problème sont triés dans au moins de trois types d’algorithmes.

1. Les algorithmes d’EKF (Extended Kalma Filters).
2. Les algorithmes utilisant les techniques d’optimisation à bas de graphe.
3. Les algorithmes utilisant les méthodes de particule.

Chaque type a ses typiques caractéristiques, ses points forts, ses limites. Dans chaque type, il y a des algorithmes, des approches différentes pour résoudre un SLAM problème. Donc, il faut comparer les types d’algorithmes, les algorithmes et identifier lequel qui convient notre SLAM problème.

## Le SLAM visuel

Un robot a beaucoup de façons de percevoir des informations de l’environnement. Il peut utilise des infrarouges, des radars, des lasers, des caméras comme sa capteur. Si le robot n’utilise que des caméras pour prendre des images d’environnement, il utilisera des images prises pour découvrir l’environnement. Les images prises sont utilisées dans des manipulations de localisation et cartographie. C’est le SLAM visuel.

Il y a beaucoup de manipulations que le robot doit faire pour la localisation et cartographie.

1. Détecter les points de repère (landmark) dans l’environnement. Le robot doit estimer les poses de l’environnement. Le robot peut calculer la position de pose dans l’environnement par deux images de cette pose. Il utilise **l’épipolaire contrainte** et la **triangulation** pour détecter la correspondance de deux points dans deux images différente.
2. Le robot peut-être doit détecter les obstacles par utiliser le technique **stereopsis**.
3. Il peut-être doit mésuser la distance des objets loins par des objets près par utiliser le technique **parallax.**
4. Le robot doit détecter si un objet dans son trajet est connu ou déjà-vu. Cela se passe dans le cas où le robot doit traquer, naviguer dans l’environnement. L’autre cas, le robot peut-être détecter le type des objets dans son trajet. C’est la **catégorisation** d’objets. On doit utiliser des technique de **Recognition Visuelle** (**Visual Recognition** – un sous-domaine de vision par ordinateur).

Le nombre d’œil de robot est un des facteurs de SLAM problème. Dans le problème de ce TPE sujet. On utilise le robot qui a une caméra, c’est-à dire, a un œil. Donc, il faut attention à de technique de vision par ordinateur si le robot a un-œil, tel que **monocular cue.**

## Le travail de TPE

En théorie, je dois chercher des algorithmes, des méthodes de SLAM, en particulier, de SLAM Visuel. Je dois comparer les algorithmes, déterminer leurs points forts et leurs limites, identifier quel type de SLAM problème rationnel avec chaque algorithme.

Un problème de ma recherche théorique est que les nouveaux algorithmes, les méthodes sont souvent offlines. C’est-à dire on ne publique pas sa recherche, ou il y a moins documents sur le sujet.

En pratique, je dois implémenter quelques algorithmes pour identifier les théories sur les algorithmes.

On n’utilise pas le robot de l’IFI pour implémenter les algorithmes car:

* Il y a beaucoup d’erreurs de mesure du robot.
* La qualité de caméra de robot (webcam).
* La puissante de procéder les calculs de robot.

Mais, c’est un avantage de mon practice parce que je faire attention aux algorithmes. Et c’est aussi un inconvénient parce que je n’aurai pas l’occasion pour implémenter les algorithmes sur le robot.

Pour restreindre les inconvénients, je dois faire de connaissance sur le robot. Les connaissances sont utilisées pour collecter des données entrées en pratique. Par exemple, je vais utiliser un appareil de photo pour prendre des photos comme un robot. L’hauteur de l’appareil de photo au bas est égale auquel de caméra du robot.

# L'état de l'art

Dans les années récentes, le problème SLAM (Simultaneous Localization and Mapping) est considéré comme un problème important de la robotique.

Dans le problème de SLAM, on doit résoudre le problème de l’incertitude : l'incertitude de la position de robot, l'incertitude de la position de l'objet dans l'environnement, l'incertitude d'identifier et reconnaitre des points d'intérêt dans l'environnement.  
Le problème de SLAM est pour le robot, donc ses solutions doivent être réalisées en temps réel. Cela requiert que le temps de calculatoire, l'espace de stockage et l'incertitude soient petits pour construire une bonne carte acceptable et la trajectoire de robot.

## Des approches pour le problème SLAM

Pour résoudre le problème SLAM, plusieurs méthodes sont présentées. La méthode la plus connue est les méthodes probabilistes.

## Les méthodes probabilistes

### EKF - Extended Kalma Filters

La méthode la plus importante est EKF ([1], [2]). EKF est basé sur l'estimation gaussienne à chaque pose de robot et chaque point de repère. L'algorithme d'EKF contient trois étapes. Premièrement, dans l'étape de prédiction, on estime l'état de système (la position de robot, sa covariance, l'observation) au temps t en utilisant les données au temps t-1. Deuxièmement, dans l'étape d'observation, d'après l'observation de robot, on calcule la covariance de la nouvelle observation avec laquelle vieille. En fin, dans l'étape de mise à jour, on met à jour l'état de système courant avec la nouvelle observation.

L’incertitude du robot augmente de temps en temps par l'accumulation de bruit de mesure. Cette incertitude augmente jusqu’à le robot sent un point de repère vieille. À ce moment-là, les erreurs de ses poses se baissent. C’est à dire, si on augmente l’information sur les poses senties de robot, cette information se propage à tous les vieilles poses. C’est une caractéristique la plus importante de “SLAM posterior”.

Plusieurs méthodes à améliorer l'EKF sont présentées telles que CEKF (Compressed EKF) ([3], [4]), Square Root SAM (Smooth and mapping) [5], SEIF (Sparse Extended Information Filter) [6], TJTF (Thin Junction Tree Filter)[7].

### Inférence de Bayes

L'autre approche est de considérer le problème SLAM comme un modèle Markov et utiliser l'inférence de Bayes pour résoudre le problème [8]. Mais cette méthode n'est pas convenable à une carte avec beaucoup de point de repère. Il existe toujours en pratique que le nombre de point de repère est environ mille, millions points de repères.

### EM (Estimation et maximisation)

En suite, on utilise l'EM ([9], [10]) pour estimer le maximum de vraisemblance de carte. Cette méthode se compose deux étapes. Dans l'étape de l'estimation, une estimation de toutes les poses de robot est réalisée à base de la carte courante et les données en tout le temps. Dans l'étape de maximisation, une carte plus meilleure est calculée à partir de l'estimation de la première étape. La carte meilleure est le maximum de l'espoir de la logarithme de vraisemblance jointe des poses de robot et les données en tout le temps.

L'EM a quelques avantages en comparant avec EKF. Premièrement, l'EM fournit une méthode sans demander de l'identification unique de point d'intérêt comme EKF. L'association de données est réalisée itérativement sur toutes les données d'observation. Donc, l'estimation dans le passé est revue. Mais, l'EM ne requiert pas une assume de bruit Gaussien comme l'EKF.

Mais l'EM présente quelques désavantages. Premièrement, l'EM procède toute la carte dans chaque étape. Donc, l'EM ne fournit pas une solution augmentant que l'estimation est meilleure si on ajoute une nouvelle observation comme l'EKF.

### Relaxation et « Multilevel Relaxation »

Une l'autre méthode de SLAM relative à l'estimation de maximum de vraisemblance est la relaxation ([11], [12]). On utilise « Gausse-Seidel Relaxation » pour résoudre un système d’équation de problème SLAM. L’idée de cette méthode est « déplacer un nœud à sa position où ses voisinages pense qu’il est ». Cette méthode est appliquée beaucoup dans des algorithmes récents. Le « Multilevel Relaxation » [13] est en base d’une méthode multigrille (multigrid method) de résoudre un système d’équation différentielle partielle. Cet algorithme augmente la converge de Relaxation par optimiser la carte de chaque couche.

* 1. Relaxation « un couche » pour SLAM.

Relaxation « un couche » est une étape dans « multilevel relaxation ».  
**Entrée**: un ensemble de relations entre des cadres. Chaque cadre est équivalente un pose de robot.  
**Sortie**: un vecteur de maximum de vraisemblance de chaque cadre.  
**Étapes**:  
-Linéariser la fonction de mesure  
-Calculer la fonction d’erreur quadrique  
-Trouver le minimum de vraisemblance d’erreur quadrique par résoudre itérativement un système d’équation linéaire en utilisant « Gauss-Seidel Relaxation ».

* 1. « Multigrid »  
     C’est un processus de discrétiser et interpoler une équation différentielle partiale. à chaque couche, on restreint un problème et gérer un problème grossier par un opérateur de restriction. Un problème le plus grossier est résout. En fin, le problème est continué par un opérateur d’interpolation.
  2. Relaxation Multi-couche (Multilevel Relaxation)

On choisit chaque deux cadres de trajectoire de robot pour générer chaque niveau de grossièreté.

À chaque niveau, on implémente un opérateur Galerkin basé V-cycle de « multilgrid » pour mettre à jour l’estimation de maximum de la vraisemblance à chaque nouvelle cadre.

### Filtreur de particule

Une autre approche est le filtreur de particule. On attache chaque pose de robot avec des filtreurs de particule ([14],[15]). Chaque particule a un poids de l'importance. Ce poids implique la relation de cette particule avec l'autre particule. Les filtreurs de particules sont échantillonné et re-échantillonné dans chaque fois ou le robot perçoit une nouvelle observation.

**Les étapes de FastSLAM:**

1. Initialiser SLAM.  
   - Mettre la particule de la location de robot à la valeur de coordonnée de démarrer.  
   - Mettre la carte à zéro.
2. Mettre à jour des particules

Si on a déjà reçu une odométrie. On générer par hasard une particule pour la nouvelle location de robot. La distribution de générer des particules base sur un échantillon de motion. L’échantillon de motion est un probabilité conditionnelle de nouvelle location du robot au temps t, sachant la location du robot au temps t-1 et l’odométrie au temps t. Cette étape s’appelle l’étape d’échantillonnage probabiliste (probabilistic sampling step).

Si l'observation est reçue. À partir de chaque particule vieille, on génère une nouvelle particule selon l'état actuel. Les nouvelles particules s'appellent « proposal distribution » En suite, on re-échantillonne les particules par remplace de tous les vieilles particules avec les nouvelles particules à base de leur poids d’importance. Cela s’appelle l’étape de re-échantillonnage.

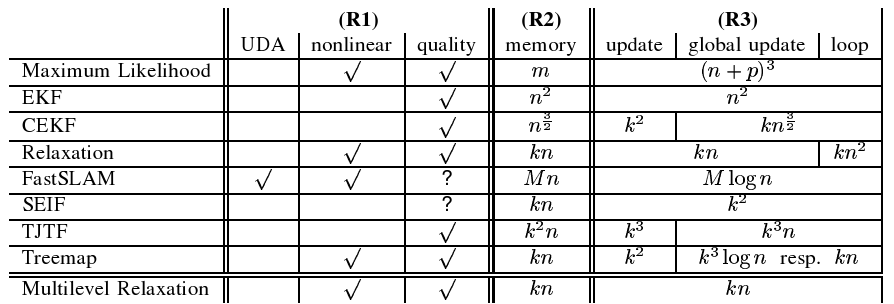
En fin, on recalcule les valeurs des moyennes et covariance avec chaque nouvelle particule à base de la mesure d'observation.

Un problème de méthode de particule est que le filtreur de particule dessine exponentiellement à l’échelle de dimension d’espace d’état. La dimension d’espace est acceptable environ 3 ou 4.

FastSLAM est une des algorithmes les plus faciles à implémenter.  
**Des propriétés de FastSLAM.**  
- Le FastSLAM résout un problème plein et en-ligne de SLAM. Chaque particule a un échantillonnage de trajectoire entière de robot. Les équations de mise à jour sont actuellement pour les poses les plus récentes. Donc, le FastSLAM ressemble un filtreur, similaire EKF.  
- Le FastSLAM est facile à travailler avec plusieurs hypothèses d’association de données. On prend une décision d’association de donnée à base de chaque particule. On n’utilise pas une seule association de données pour un filtreur entière. FastSLAM peut échantillonner les corrects postérieurs que deux types d’algorithmes avant ne peuvent pas demander.  
- Le FastSLAM est efficacement à implémenter. On utilise une méthode d’arbre avancé pour représenter l’estimation de carte. Le temps de mettre à jours est un logarithme de la taille de carte et un linéaire de nombre de particule.

Raoblackwellized (Rao-Blackwellized Particle Filter – RBPF) ([17], [18]) est un variant de filtreur de particule. Le RBPF peut conduire un résultat plus correct que « Kalma Filter » et « Hidden Markov Modele Filter ».

**La table de comparaison de ces méthodes**



R1, R2, R3 sont trois exigences du problème SLAM. R1: l'incertitude est bordée par un seuil. R2: l'espace de stockage est linéaire avec la taille de carte. R3: la mise à jour est linéaire avec la taille de carte. C'est la table de performance de l'algorithme de SLAM avec n points de repères, m mesures, p poses de robot, k points de repère sur une image (point d'intérêt). UDA signifie « uncertain data association ». Cette table est extraite de [13].

## SLAM visuel

Le robot utilise beaucoup de moyens de percevoir l'environnement tels que le laser, l'infrarouge, la caméra. La caméra n'est pas considérée comme un centre des recherches de SLAM. Malgré que la caméra est compacte, bien-compris, bon marché, et ubiquitaire. Mais, il est difficile à construire une carte dans un long terme en temps réel à partir de point d'intérêts extraits de l'image car la rate de données d'une caméra est beaucoup plus haute que celle de l'autre senseur. Dans les années récentes, des recherches sur SLAM visuel donnent beaucoup d'approches. Les approches données sont souvent basées sur le modèle de calcul de la probabilité mentionnée avant en combinaison avec des approches de l'observation par caméra.

### Point d’intérêt

Les idées principales de slam visuel sont de faire la correspondance entre les points d’intérêt dans des images différentes séquentielles. Un point d’intérêt est un point facile à détecter, stable, intéressant à la mise en correspondance des images.

#### Coin de Harris

Un coin de Harris [21] est un type de point d’intérêts communs à utiliser car ses avantages de la rotation, l’échelle, la variation de lumière et des bruits d’image. Coin de Harris se base sur une fonction de détecter le changement d’intensité des régions par des petits mouvements locaux vers différents direction. Un point est détecté comme un coin si l’intensité change par toutes les directions de mouvements de région sur ce point.

#### SIFT - Scale-invariant feature transform

Un point dans une image extraite une description SIFT. La description SIFT de chaque point contient le vecteur gaussien. La description SIFT d’un point est invariante avec l’échelle, le changement d’orientation, la distorsion et le changement de lumière.

#### Coin de Kanade – Tomasi

Le coin de Kanade – Tomasi [22] se base sur le détecteur de coin Harris. L’auteur exprime ce que les régions d’image affine la transformation.

|  |  |
| --- | --- |
| Le coin de Harris | Le point SIFT |
| Le coin de Kanade Tomasi – la région de panneau 25 | |
| Source : Dmitrij Csetverikov, Finding Patterns in Images (Hungary). David Lowe. Distinctive Image Features from Scale  Invariants Keypoints », IJCV, 60(2), 2004.: J. Shi and C. Tomasi, Good Features to Track 1994 | |

Figure Les points d'intérêt

### Odométrie visuelle

C'est une méthode de localisation avec l'entrée étant une seule série des images ([20], [32], [33]). La base de cette méthode est de traquer des points d'intérêts. Des points d'intérêts sont faits la correspondance entre deux images et liés dans tout le processus pour construire une trajectoire de robot. Ensuite, l'estimation de motion de robot est réalisée par l'utilisation d'une architecture géométrique hypothèse-et-test.

Dans chaque image, on détecte des Harry coins [21] comme point d'intérêt. Les coins sont considéré stable avec le problème de déformation base. En contraste à traqueur de KLT [22], on détecte des points d'intérêts dans tous les frames, et fait seulement la correspondance entre des points d'intérêts. Dans le processus d'estimation de motion, on peut réaliser une dans deux implémentations sur le monocular ou le stéréo.

**Implémentation monoculaire**

1. Traquer des points d'intérêts sur certaines cadres. Estimer des poses de robot entre 3 cadres en utilisant l'algorithme « 5-points » [23] et « preemtive RANSAC » [24].
2. Trianguler les points d'intérêts à points 3D en utilisant la première et dernière observation de chaque traque.
3. Traquer certaines images additionnelles. Calculer la pose de camera basée sur des points 3D connus en utilisant l'algorithme « 3-points » [25] et « preemptive RANSAC ».
4. Re-trianguler des points 3D en utilisant les observations première et dernière dans la traque du point. Répéter l'étape 3 sous certaines fois.
5. Répéter de l'étape 1 à l'étape 4 sous certaines fois.
6. Insérer la valeur de « pare-feu » et répéter de l'étape 1 sous certaines fois.

**Implémentation stéréoscopique**

1. Faire la correspondance entre les images gauche et droites de la paire de stéréo. Trianguler des correspondances à des points 3D.
2. Traquer les points sur certaines cadres. Calculer la pose en utilisant « preemptive RANSAC ». L'algorithme « 3-point » (considéré dans les images gauches) est utilisé comme un générateur des hypothèses.
3. Répéter l'étape 2 en certaines fois.
4. Trianguler tous les correspondances des nouveaux points d'intérêts en utilisant des observations des images gauche et droite. Répéter l'étape 2.
5. Re-trianguler tous les points 3D pour construire un pare-feu. Répéter de l'étape 2.

On utilise le pare-feu pour restreindre l'accumulation et la propagation de l'erreur en impliquant simplement que la triangulation n'est pas réalisée si l'observation utilisée dans cette triangulation est plus loin que la valeur de pare-feu récent. On a indiqué que l'odométrie visuelle peut être comparable avec « Differential Global Positioning System (DGPS) » et « Inertial Navigation System (INS) ».

### Gamma-SLAM

On utilise l'odométrie visuelle et un filtreur de particule Raoblackwellized. On utilise une distribution gamma pour représente les hauteurs des objets de l’environnement. Donc, cet algorithme s’appelle Gamma-SLAM [26]. On maintient une distribution postérieur sur la variance de hauteur dans chaque cellule. Certains problèmes SLAM visuel utilisent la technique de stéréo [27]. La plupart des algorithmes de SLAM visuel en base de stéréo aujourd’hui sont basés des repères.

On peut représenter la carte par une grille. Chaque cellule représente une pose de robot. Dans cet algorithme, on peut considérer une carte comme une grille de la probabilité conditionnelle qui confirme la correction de la carte.

Cet algorithme utilise gamma distribution pour calculer la vraisemblance valeur de l’hauteur des objets dans l’environnement. En comparant avec l’algorithme qui utilise seulement Odométrie Visuelle, le résultat obtenu est moins bon que par utiliser gammaSLAM. Une construction d’une carte avec l’hauteur des objets est bonne pour la planification et la navigation de robot dans l’environnement non-structuré.

### Mini SLAM

On a introduit une approche pour un problème de SLAM dans un environnement large avec des exigences de perception et computation minimale [28]. Cette approche basée sur une représentation d’un problème de SLAM par un graphe des poses de robot et des relations entre eux. Pour estimer la covariance pour des liens dans le graphe, miniSLAM est introduit.

Une carte de problème SLAM est présentée par un graphe. Chaque nœud présente une cadre correspondance une pose de robot et l'estimation de cette pose. L’arc représente une relation entre deux cadres. Cette relation se compose de 2 relations: la similarité visuelle et la similarité odométrique. La sortie est un vecteur d’estimation du maximum de vraisemblance pour toutes les poses de cadres.  
Différence avec l’autre SLAM:   
N’utilise pas d’estimation de position des points de repères relatifs. Donc, il est facile à l’appliquer à un environnement grande. Il utilise une estimation des différences entre des points de repères entre des images séquentielles et l’incertitude de cette estimation. Des différences considères sont des relations visuelles et des relations odométriques. Il utilise l’algorithme « Multilevel Relaxation » pour l’estimation du maximum de vraisemblance.

Les auteurs ont réussi avec miniSLAM dans un pratique avec des images omni-dimentionnelles.  
C’est un point attentif. Parce qu’il y a une différence entre le changement de le nombre de similaires d’image omni-dimentionnelle et celui d’image non omni-dimentionnelle. Donc, je pense que l’on ne peut pas appliquer ce calcul pour le robot ayant un caméra non omni-dimmentionnelle avec un champ de vue petit.

### σSLAM

σSLAM, σ signifie stéréo, utilise une estimation stéréoscopique pour estimer la position de repère [29]. Et une estimation de motion est conduite par un flux optique clairsemé. On distingue des repères par « Scale Invariant Feature Transform » (SIFT). Cela est différent avec des approches courantes qui utilisent une odométrie mécanique pour estimer la position de robot et un scan par laser pour distinguer des repères.

Cet algorithme a proposé une « proposal distribution » dans l’approche de « RaoBlackwellized Particle Filter » (RBPF). C'est une mixture entre le modèle de motion et une distribution  
On a conduit une carte dense des 3D landmarks identifié par SIFT. L’invariance et la nouvelle « proposal distribution » améliorent le problème de fermer un cycle large (« closing large loop »).   
On a indique que cet algorithme est pour un environnement avec la lumière haute (beaucoup de fenêtre en glass). On a indique que l’on peut produire une grille 2D d’occupation. Cette grille est bien pour la planification et la navigation de robot. Une observation est un ensemble des correspondances entre les points de repères de la vue courante et les points de repères stockés dans la carte de robot.

### MonoSLAM

C'est une méthode pour une seule camera incontrôlée. Créer une carte en ligne clairsemée et persistent. C'est à dire d'analyser une séquence des images pour produire une reconstruction de trajectoire de robot et structure de la scène observée. Mais cette méthode est bien pour une séquence courte de l'image. L'auteur concentre à faire la localisation comme un travail principal. La carte est construite mais c'est clairsemé et optimisée dans le processus de localisation.

Cette méthode représente le monde sur une carte 3D probabiliste [34]. Carte représente l'estimation courante de l'état de caméra et les points d'intérêts et encore l'incertitude de cette estimation. La carte est initialisée au début et mise à jour par la méthode EKF à chaque fois de la motion de robot et son observation.

On a choisi des larges zones de l'image comme les points de repères naturels ([35], [36]). On détecte les patchs par utilisation une technique mentionné dans [22]. Dans l'initialisation de SLAM, le robot ne connait rien dans l'environnement. Mais, dans ce cas, on aide le robot par ajouter une petite information sur l'environnement. C'est souvent 4 points de repères avec leurs positions.

On considère que la caméra se déplace avec une vitesse de mouvement et de mouvement angulaire constant. On cherche des zones template pour l'estimation de position de l'image [37] en utilisant les faits de correspondance. Pour augmenter la vitesse de cherche, on calcule une prédiction la position de point d'intérêt et l'incertitude de cette prédiction.

D'après l'identification et la mesure première de point d'intérêt, on crée une ligne 3D sur la carte vers le point d'intérêt. C'est une ligne à partir de la position estimée vers le point d'intérêts sur la direction d'observation. Un ensemble d'hypothèses de point d'intérêts est distribué sur cette ligne. En suite, on calcule une approximation de la profondeur de ce point d'intérêt. En fin, on construire ce point d'intérêt sous une forme pleine avec une représentation Gaussienne.

Gestion de carte est de gérer le nombre de points d'intérêts et déterminer quand on peut ajouter un nouveau point d'intérêt ou supprimer un point d'intérêt de la carte. On essaie de maintenir le nombre de point d'intérêts ayant la visibilité proche une valeur prédéterminée. La visibilité de point d'intérêt est estimée par la distance relative de caméra et ce point, et la position de caméra au temps de l'initialisation de ce point. Le point d'intérêt doit être dans l'image et la caméra doit se placer pas trop loin.

Un point d'intérêt est ajouté si le nombre de point d'intérêts visibles est plus petit qu'un seuil. Un point d'intérêt est supprimé si le nombre d'essai échoué de le détecter et le faire correspondance est plus petit qu'un nombre prédéterminé. L'orientation de chaque point d'intérêt est estimée comme la mention dans [38].

Table de comparaison entre l'odométrie visuelle et monoSLAM

Table La comparaison entre odométrie visuelle vs Monoslam

|  |  |  |
| --- | --- | --- |
|  | **Odométrie Visuelle** | **MonoSLAM** |
| **Modèle de vision** | Stéréoscopique, Monoculaire | Monoculaire |
| **Faire la correspondance** | Tous les points d'intérêt | Gestion de point d'intérêt sous une gestion de carte |
| **Application** | σSLAM, Gamma SLAM | Seulement dans monoSLAM, open source |
| **Point d'intérêt** | Coin Harry | Large région de l'image (11x11) |

Table La comparaison entre 5 approche de SLAM visuel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MiniSLAM** | **σSLAM** | **GammaSLAM** | **MonoSLAM** |
| **Modèle de calcul** | Multilevel Relaxation | RaoBlackwellized Particle Filter | RaoBlackwellized Particle Filter | EKF |
| **Odométrie** | Odométrie mécanique de robot | Odométrie visuelle | Odométrie visuelle | Approche de monoSLAM comme une odométrie visuelle |
| **Représentation de carte** | Graphe de cadre | Une grille 2D d'occupation | Une grille de la probabilité | Carte 3D probabiliste |
| **Modèle de vision** | Stéréoscopique | Stéréoscopique | Stéréoscopique | Monoculaire |
| **Point d'intérêt à traquer** | SIFT | SIFT | Coin Harris | Large région de l'image (11x11) - coin de Kanade-Tomasi |
| **Caméra** | omni-dimensionnelle |  |  | Une seule camera incontrôlée. |

## SLAM 3D

Beaucoup de recherche pour construire une carte volumétrique 6DoF (6 degree of freedom) [39]. Mais tous les algorithmes utilisent des scanners laser pour fournir des données 3D. C'est un matériau principal pour construire une carte volumétrique. Une caméra ne suffit pas de la capacité de fournir ce type de données. Donc, on n'utilise pas une caméra comme un senseur pour le problème de SLAM 3D.

# Le travail pratique

Je dois réaliser de faire les étapes comme :

## Chercher les logiciels

### Openslam

Openslam est un site web se trouvant à l’adresse <http://openslam.org>. Dans ce site, il y a quelques algorithmes sur slam tels que CEKF Slam, TJTF Slam, Slam 6D. Mais, tous les algorithmes sur ce site maintenant sont basés sur le laser de robot. Donc, ils ne conviennent pas avec le but de ce TPE.

### Mrpt

Mrpt est une librairie de développement de robot. On peut installer mrpt sur ubuntu directement par sa distribution. Dans le site officiel de mrpt, <http://www.mrpt.org> , on peut trouver beaucoup d’algorithmes Slam, d’outils et de documents. La plupart d’algorithmes sont basés sur le laser. Il y a maintenant seulement l’algorithme Monoslam appartenant à type de slam visuel. Mais, l’implémentation de monoslam dans ce site n’est pas maintenue maintenant. J’ai éssayé de compiler le code de monoslam. Mais après avoir réussi de compiler, le programme de monoslam ne fonctionne pas car il y a eu de changement de source code après sa première version. Donc, c’est difficile à utiliser cette implémentation.

Dans Mrpt, je trouve des outils très utils pour le développement de robots tels que Rawlog Viewer, pour lire le fichier représentant l’état de robot (odométrie, observation, etc.), Camera Calibration, pour réaliser la calibration avec une caméra, 3D Scene Viewer, pour observer des scenes en 3D. Pour mon tpe, j’utilise 3D Scene Viewer pour observer la scene de l’algorithme de slam.

### ROS

Ros est une librairie de développement de robot se trouvant à l’adresse <http://www.ros.org> . Ros a beaucoup de paquets pour développement de robot. Mais il y a moins d’algorithmes slam dans ROS que dans MRPT. Il y a seulement d’algorithme vslam étant de type de Slam visuel. Mais, le module de vslam n’est pas maintenu. J’ai essayé de compiler le vslam dans la poubelle de ROS. Mais, après avoir réussi de compiler, cette version de vslam ne supporte pas le stream d’images monoculaires. Et le team de développement de vslam a annoncé qu’il le finira à la fin de cet été.

### Scene

La librairie Scene est pour l’algorithme de monoslam d’Andrew J. Davison, l’auteur de monoslam. On peut trouver cette librairie à l’adresse <http://www.doc.ic.ac.uk/~ajd/Scene/> . Après fixer les erreurs simples de la compilation, le programme de monoslam a marché. Et j’ai décidé d’utiliser cette librairie.

## Choix de logiciel

Le processus de choix de logiciel se passe d

## Choix de caméra

Dans le travail pratique, pour collectionner des données, on prend des vidéos. Donc le choix de caméra est très important. J’ai essayé d’utiliser 3 caméras : la caméra d’appareil photo de Canon, la webcam de mon portable, la webcam Logitech Express V-UH9, V-UAP9 de l’IFI.

Avec la caméra d’appareil photo de Canon, la qualité de l’image est très bonne. Mais, un problème avec l’appareil photo est le mode automatique. Dans le processus de prendre une vidéo, le focal de l’appareil photo se change. Donc, si on applique cette vidéo avec le programme monoslam, le résultat obtenu est très mauvais.

Avec la webcam Logitech, le focal ne change pas dans le processus de prendre une vidéo. On peut changer le focal avant de prendre une vidéo. Donc, chaque fois on change le focal de webcam, on doit refaire la calibration avec le nouveau focal pour obtenir les valeurs intrinsèques convenables. C’est un inconvénient de cette webcam. L’autre inconvénient est que la qualité des images n’est pas bonne. Il y a beaucoup de bruits dans les images. L’avantage de cette webcam est que l’on la prend à la main facilement, et contrôle la direction de caméra facilement.

Avec la webcam mon portable, le focal est toujours fixé. L’autre part, la qualité des images est acceptable. Il y a moins de bruits dans les images. L’inconvénient de cette webcam est fixé à mon portable. Donc, c’est difficile à la prendre à la main, la contrôler la direction de caméra.

A partir des avantages et inconvénients de chaque caméra, je décide de choisir la webcam de mon portable. Parce que la qualité de vidéo obtenue est meilleure.

## Camera calibration

La calibration de camera est un mesure les paramètres d’intrinsèques de caméra. Ce sont des paramètres internes de chaque caméra. La calibration est obligatoire pour les traitements d’image relative à caméra. Les paramètres d’intrinsèques se composent des valeurs de focaux représentés dans la matrice d’intrinsèque, et des valeurs de distorsion. Chaque caméra a la valeur de focal différente. Lorsque la lentille de caméra n’est pas toujours parfaite, l’image obtenu est distorse, pas rectifie. Donc, il faut trouver le coefficient de distorsion pour la rectification des images traitées.

Pour obtenir les paramètres d’intrinsèques de caméra, j’utilise l’outil « Camera Calibration » de MRPT. Je prépare un tableau d’échec comme l’instruction dans le wiki du site MRPT (<http://babel.isa.uma.es/mrpt/index.php/Application:camera-calib-gui>).

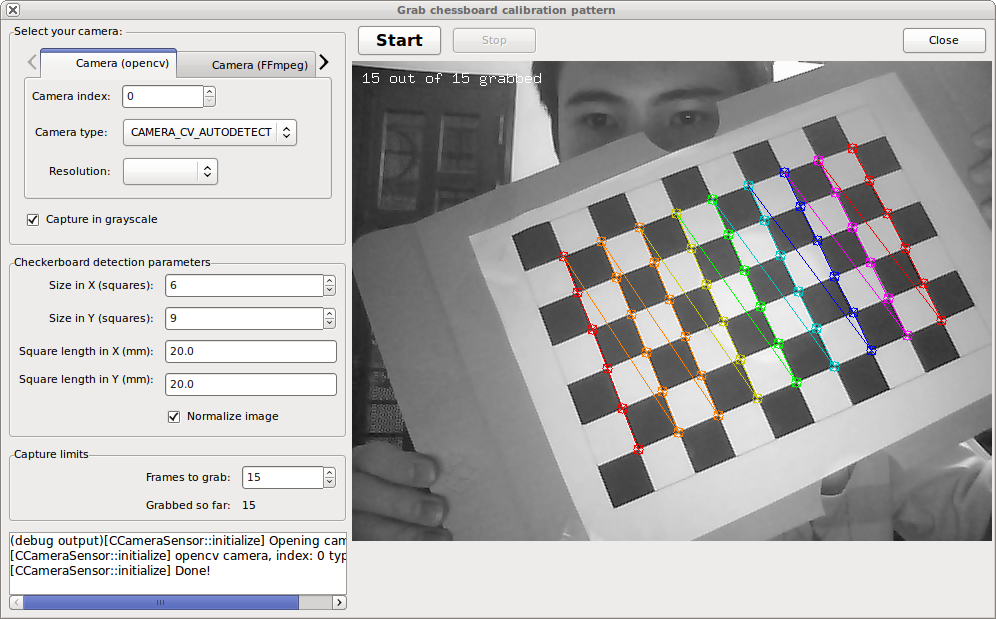


Figure L'interface de programme calibration

Dans le tableau, on a 9 points horizontaux, et 6 points verticaux. La distance entre deux points proches est 2.4 centimètres. Ces paramètres sont entrés au programme de la calibration de MRPT comme le cercle rouge dans la figure 1. On change la direction de tableau d’échec pour que cet outil détecte les points dans le tableau d’échec.

En fin, on obtient le résultat de la calibration. Ce sont la matrice d’intrinsèque et des coefficients de distorsion. Voici la matrice d’intrinsèque et les coefficients de distorsion obtenu à partir de la webcam de mon portable.

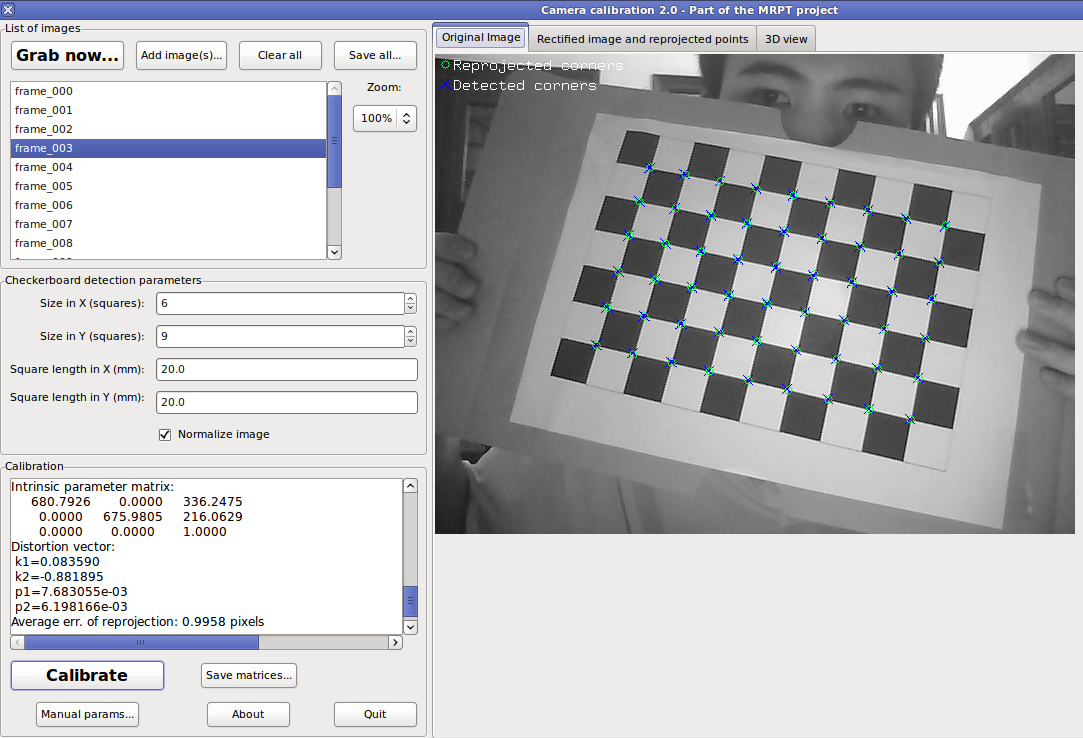


Figure Résultat de la calibration

La valeur de la calibration avec la résolution de 640x480.

Table La matrice intrinsèque de caméra avec la résolution de 640x480

|  |  |  |
| --- | --- | --- |
| 681 | 0 | 336 |
| 0 | 676 | 216 |
| 0 | 0 | 1 |

Table Le vecteur de distorsion de caméra avec la résolution de 640x480

|  |  |  |  |
| --- | --- | --- | --- |
| 0.08359 | -0.881895 | 0.00768 | 0.006198 |

L’erreur moyenne de projection est de 0.9958 pixels.

La valeur de la calibration avec la résolution de 320x240.

## Sélectionner les données

Le scénario de test est construit.

Les outils pour la préparation des données.

Convertir le fichier de vidéo en format .ogg à en format .avi. Parceque on rencontre des difficultés d’extraire des images à partir d’une vidéo .ogv. Donc, je dois convertir le vidéo à format .avi.

Outils d’extraction des images à partir d’une vidéo. J’utilise ffmpeg pour extraire les images à partir d’une vidéo .avi.

# Programme

## La configuration

La configuration de monoslam est stockée dans le fichier monoslam\_state.ini. Voici la liste des paramètres de la configuration :

Table Table de champs de configuration de monoslam\_state.ini

|  |  |  |
| --- | --- | --- |
| **Champs** | **Valeur initiale** | **Description** |
| **[Models]** |  | Les modèles |
| MotionModel | IMPULSE\_THREED | Le modèle de mouvement. La valeur est le nom de champs qui contient la configuration pour ce modèle. |
| NewFeatureMeasurementModel | CAMERA\_WIDE\_LINE\_INIT | Le modèle de mesurer de nouveaux points de repère. Le contenu de ce champ est le nom de champs de type de caméra. |
| **[CAMERA\_WIDE\_POINT]** |  | Le nom d’un type de mouvement. |
| Camera | UNIBRAIN\_WIDE\_CAMERA | Le nom de type de caméra utilisé dans ce modèle de mesurer |
| **[CAMERA\_WIDE\_LINE\_INIT]** |  | Un champ de type de mouvement |
| Camera | UNIBRAIN\_WIDE\_CAMERA | Le nom de type de caméra utilisé dans ce modèle de mesurer |
| **[UNIBRAIN\_WIDE\_CAMERA]** |  | Un champ de type de caméra qui contient des paramètres de caméra |
| CameraParameters | 320 240 162 125 195 195 9e-06 1 | Les paramètres d’une caméra.  La valeur de caméra incluse :  La taille des images : 320x240  La valeur de matrice d’intrinsèque de caméra :  (Fu, Fv, u, v) = (162, 125, 195, 195). Fu,Fv, est la longueur de focal en pixels. Les coefficients u, v sont le centre optique dans la direction u,v d’image.  La première valeur de la matrice de distorsion : 9e-06  L’écart type (standard deviation) de mesure des images : 1pixel.  Ces valeurs sont obtenues à partir de la calibration de caméra. |
| **[IMPULSE\_THREED]** |  | La région de la modèle de mouvement. |
| **[InitialState]** |  | Les états initiaux de système |
| MotionModel | IMPULSE\_THREED | Le modèle de mouvement « Impulse 3D » |
| xv | 0 0 -0.6 1 0 0 0 0 0 -0.1 0 0 0.01 | La position initiale de robot et la direction de caméra. C’est le vecteur d’état de robot avec 13 valeurs.  vector_etat.png |
| Pxx | 0.0004 0 0  0 0.0004 0  0 0 0.0004 | La matrice de covariance entre la position de robot et tous les points de repère. |
| **[KnownFeature1]** |  | Les valeurs de points de repère initiaux |
| FeatureMeasurementModel | CAMERA\_WIDE\_POINT | Le modèle de mesurer cette valeur |
| Identifier | known\_patch0.pgm | Le nom de fichier d’image qui représente ce point. |
| yi | 0.105 0.07425 0 | Les ordonnées (x,y ,z) de ce point de repère |
| xp\_orig | 0 0 -0.6 1 0 0 0 | La position originaire de robot où le robot détecte ce point de repère. |

## L’interface graphique

La fenêtre de contrôle des entrées.

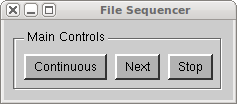


Figure Fenêtre de contrôle des entrées de monoslam

Dans la fenêtre de contrôle de séquence de fichier entré, on a trois boutons Continuous, Next, et Stop.

* Le bouton Next a pour l’objective de pousser l’image suivante à traiter.
* Le bouton Continuous a pour l’objective de pousser une séries des images séquentielles à traiter.
* Le bouton Stop a pour l’objective de arrêter de pousser une séries des images séquentielles à traiter. Après avoir cliqué ce bouton, le programme pause à observer.

La fenêtre de la simulation de Monoslam dans la librairie Scene.

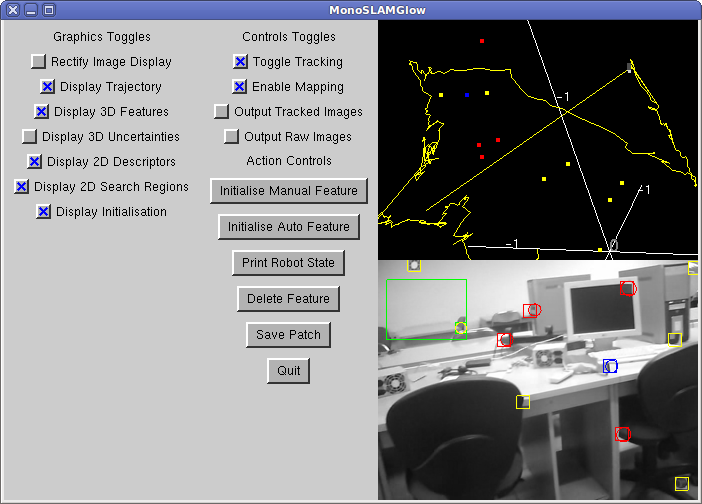


Figure L'interface de la simulation de monoslam

Dans le côté de gauche, on a des boutons de contrôler la simulation. Leurs fonctions sont présentées dans la table suivante.

|  |  |  |
| --- | --- | --- |
| No | Nom de bouton | Description de fonction |
|  | **Graphics Toggles** |  |
| 1 | Rectify Image Display | Afficher l’image rectifiée |
| 2 | Display Trajectory | Afficher la trajectoire de robot sur la carte |
| 3 | Display 3D Features | Afficher les points d’intérêt détectés sur la carte |
| 4 | Display 3D Uncertainties | Afficher les ellipsoïdes d’incertitude sur la carte |
| 5 | Display 2D Descriptors | Afficher les régions 11x11 dans l’image traitée |
| 6 | Display 2D Search Region | Afficher la région de recherche. C’est un 2D rectangle pour limiter la région de recherche |
| 7 | Display Initialisation | Afficher 4 points d’intérêts dans la processus d’initialisation |
|  | **Control Toggles** |  |
| 1 | Toggle Tracking | Decider de traquer des points d’intérêts |
| 2 | Enable Mapping | Decider de faire la cartographie |
| 3 | Output Tracked Images | Exporter les images traquées |
| 4 | Output Raw Images | Exporter les images traitées |
|  | **Action Controls** |  |
| 1 | Initialise Manual Feature | Réaliser l’initialisation avec les points d’intérêts sélectionnés manuellement |
| 2 | Initialise Auto Feature | Réaliser l’initialisation avec les points d’intérêts détectés automatiquement |
| 3 | Print Robot State | Afficher l’état courant de robot |
| 4 | Delete Feature | Supprimer des points d’intérêts sélectionnés |
| 5 | Save Patch | Exporter les régions de points d’intérêts sélectionnés |
| 6 | Quit | Quitter le programme |

Dans le côté de droite, on a la simulation. Le rectangle noir haut représente la carte obtenue. Le rectangle bas représente les images traitées avec les points d’intérêts marqués.

Caméra

Trajectoire de robot

Direction de camera

Le point d’intérêt dans la carte

Les axes

Région de recherche des points d’intérêt

Le point d’intérêt (région) incorrect (en jaune)

Le point d’intérêt initial (en bleu)

Le point d’intérêt correct (en rouge)

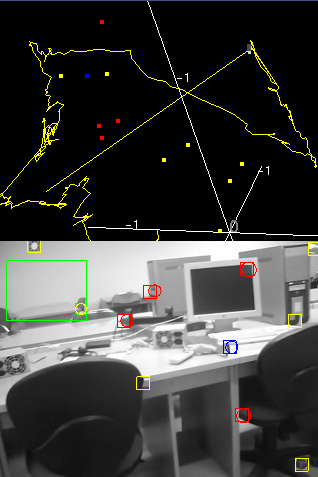


Figure Notice des notations dans l'écran de simulation

La correspondance entre la carte obtenue et l’image de traquer est présentée dans la figure 4.

|  |  |
| --- | --- |
| monoslamglow-simul.png | monoslamglow-simul.png |

Figure La correspondance entre la carte et l'image de traquer

## Structure

### La structure de Scene

La librairie Scene se compose 4 composants principaux :

|  |  |
| --- | --- |
| **Composant** | **Description** |
| VW34 | C’est la librairie d’aide de développer l’application de la vision par l’ordinateur. Ce paquet est maintenu par les membres de « Active Vision Lab, University of Oxford ». C’est un paquet vieux, n’est pas puissante que OpenCv. |
| Glow | GLOW est une librairie de développer l’application GUI basée sur OpenGL et Glut. |
| SceneLib | C’est une librairie de développer l’application de SLAM. Ce paquet inclus des fonctionalités de slam tels que EKF, monoslam, caméra, etc. |
| MonoSlam Glow | C’est un simulateur de monoslam en utilisant SceneLib, VW34, Glow. C’est le programme principal de Scene. |

VW34

Glow

SceneLib

MonoSlam Glow

### La structure de monoslam

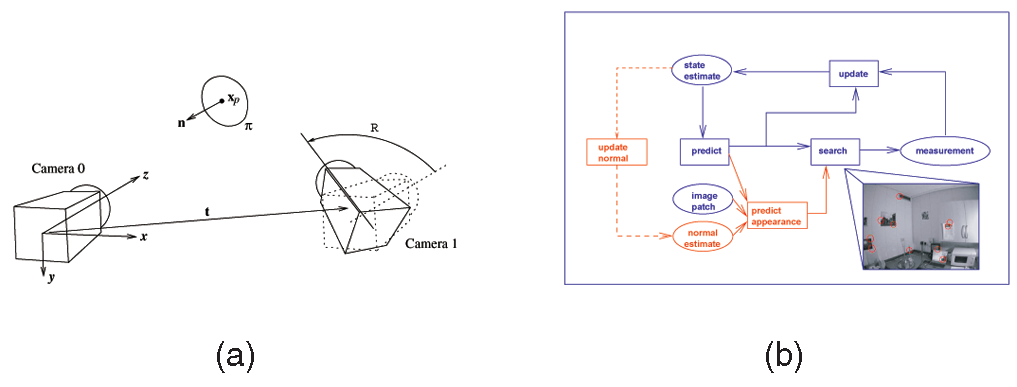


Figure L'observation d'observation par une camera

Dans cette figure, on a l’observation d’un point par le changement de la position de robot. Le déplacement de robot incluse le changement de la position par la vecteur t, et la direction par l’angle R.

Le calcul est calculé comme le modèle suivant

Estimer l’état

Prédire

Mise à jour

Chercher

Mesure

Prédire d’apparence

Image

Intégrer entre scene et mrpt 3D scene.

**Gérer la carte**

Il est important de décider d’enlever ou d’ajouter un nouveau features.

## La base de calcule

## Simulation

Moteur de monoslam

Monoslam Simulateur

Performance, Carte, Localisation Log

3DScene Log Bridge

3D Scene Log

MRPT 3D-Scene Viewer

Text-base Log Générateur

J’utilise le MRPT 3D Scene Viewer comme le simulateur de monoslam. Dans la librairie de Scene, il y a une simulation simple avec le changement de l’image observée, et la carte simple mais difficile à manipuler pour la simulation.

Séparer le viewer pour diminuer le temps de calculer du programme. On peut réutiliser les fichiers 3DScene. Dans le Scene, la sortie de traquer est des images observés marquées avec des points d’intérêt.

## Test et vérification

**La configuration de test**

Le processeur Intel Core TM 2 Duo Processor T6600, RAM 2GB. Fréquence de caméra est 25Hz.

### Simple scénario

Pour vérifier la correcte de fonctionnalité de monoslam, j’ai testé avec le simple scénario. Le but de teste est de vérifier les points bien alignés. Je mes les feuilles colorées arrangées sur le mur, et sur le tableau.

|  |  |
| --- | --- |
| simpletest.jpg  Le scénario simple avec les feuilles sur le mur | simpletest2.jpg  Le scénario simple avec les feuilles sur le tableau |

Figure Le simple scenario

****

Figure Le scenario simple avec les points bien aligné

Dans le figure précédent, nous pouvons vois les points d’intérêts étant les coins des feuilles avec l’intensité forte. Ce sont souvent les coins noirs. Dans ce cas simple, les points bien alignés sur le mur sont bien aligné sur la carte.

La comparaison entre le tableau et le mur – lumière, réflexion

### La cartographie

Le changement de l’incertitude

|  |  |
| --- | --- |
| a-0.jpg  Trame 59 | a-1.jpg  Trame 60 |
| a-5.jpg  Trame 64 | a-9.jpg  Trame 69 |

Figure La détection d'un point de repère

Le changement de l’incertitude est montré dans 4 images précédentes. Dans la trame 59, on commence à détecter point d’intérêt. Dans les trames 60 et 64, on a l’incertitude variante en jaune. En fin, dans la trame 69, on a un point d’intérêt correct en rouge.

Dans la cartographie, on a

|  |  |
| --- | --- |
| a-2.jpg  Trame 16 | a-12.jpg  204 |
| a-14.jpg  452 | a-26.jpg  478 |
| a-28.jpg  716 | a-36.jpg  850 |
| a-50.jpg  1056 | a-80.jpg  1071 |
| a-82.jpg  1242 | a-92.jpg  1284 |
| a-138.jpg  1420 | a-162.jpg  1510 |
| a-164.jpg  1511 | a-172.jpg  1517 |
| a-196.jpg  1593 | a-200.jpg  1598 |
| a-248.jpg  1686 | a-380.jpg  1787 |

Figure Les images de traquer de monoslam dans la salle de réseau

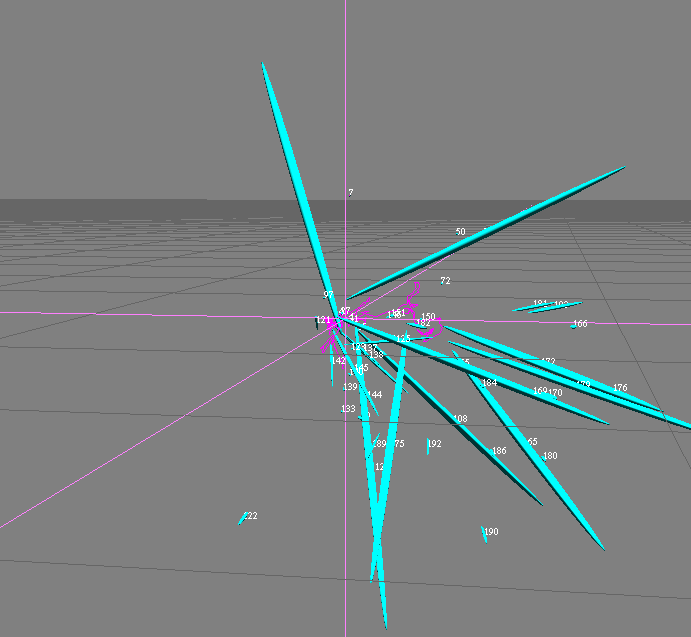


Figure Les ellipsoïdes aigues orthogonales avec la direction de mouvement

La forme des ellipsoïdes est très variante. Si l’incertitude est petite, l’ellipsoïde est en forme de petit point. Si l’incertitude est grande dans une ordonnée, l’ellipsoïde est très long. Et si l’incertitude est grande dans toutes les trois ordonnées, l’ellipsoïde est une grande sphère.

La forme des ellipsoïdes dépend la direction de déplacement de robot. Dans l’image précédente, la direction de mouvement est représentée par une ligne jaune, et l’incertitude de la position de tous les points d’intérêt est en forme aigue à la direction orthogonale avec la direction de mouvement de robot.

### La localisation

Monoslam appartient à type de FullSlam. Cet algorithme ne conserve pas les positions précédentes de robots. Il ne connait que la position dernière. Les positions précédentes ne sont pas corrigées après la mise à jour de la nouvelle position.

Le changement de la covariance

**La performance**

Le temps de calcul de chaque étape.

### La fermeture de trajectoire

La fermeture de trajectoire de robot est un problème important dans problème SLAM. Quand robot revient à sa position vieille, il est capable de reconnaitre des points de repères vieux. Et le robot recalcule sa localisation, et réutilise des points de repères vieux sans créer les nouveaux points de repère.

Dans l’implémentation de monoslam dans Scene, le robot n’est pas capable de fermer sa trajectoire. Parce que, dans monoslam, le robot conserve un nombre limité de points d’intérêts visible à traquer. Il ne fait pas la correspondance avec tous les points de repère détectés. Les points sont choisis de faire la correspondance en basant sur leur visibilité.

Donc, dans quelques fois, le robot revient à sa position, les points de repère vieux ne sont pas dans la liste de points de repère visibles. Dans ce cas là, le robot ne peut pas reconnaitre les points vieux, et il détectera les points nouveaux à la position vieille des points vieux.

### Le problème de robot kidnappé (kidnapped robot problem)

Le kidnapping est l’événement que le robot est pris à la nouvelle position quand il est en marche. Dans la nouvelle position, le robot ne reconnait pas d’un point de repère qu’il a déjà détecté. Le robot doit reconstruire sa localisation relative avec les points de repère vieux. Le problème « kidnapped robot » est un problème difficile de robot localisation. Il y a seulement quelques algorithmes SLAM qui sont capables de résoudre ce problème.

Dans l’implémentation de monoslam de Scene, on peut reconnaitre quelques événements de kidnapping. L’événement typique dans les vidéos que j’ai prises est que le robot ne reconnait soudainement pas les événements.

Au début, le robot est lointain à des points de repères détectés. Les points de repères détectés par la méthode de point d’intérêt. Les caractéristique typique de points d’intérêt est capable de s’adapter à l’échelle d’image. Mais, ils s’adaptent dans une borne. Si le robot se déplace vers les points de repères. A position proche, les points d’intérêt vieux sont échoués dans le fait de correspondance dans une nouvelle image. Donc, le robot ne peut pas détecter sa position. Et l’incertitude de sa position augmente rapidement.

Ce phénomène montre aussi la dépendance entre la localisation et la cartographie.

Avec l’environnement extérieur, la lumière change rapidement. Donc, cet événement apparait souvent.

Le taux entre le nombre de points d’intérêts supprimés

La direction de camera

Le taux de faux : l’incertitude

Faux après d’une duré

L’intérieur et l’extérieur

# Résultat

## Avantages et inconvénients

### Avantages

* Bon marché
* Visuel
* Monoculaire
* 3D

### Inconvénient

* Simple
* Non 3D

# Expérience

## Difficultés

**Difficulté à collectionner les données**

Les contraintes de données est la vitesse de mouvement et la vitesse angle est constant. Donc les données doivent être glissantes. Pour atteindre des données acceptables, il faut prendre la caméra avec la vitesse lente. Cela est très difficile si on prend la caméra à la main.

**Difficulté du choix de logiciel**

Dans le processus de choix de logiciel, c’est difficile à décider. En particulier, il y a beaucoup d’algorithmes de slam disponibles dans l’internet. Je dois identifier quels algorithmes étant de type Slam visuel. Et je ne trouve que deux algorithmes de Slam visuel : monoslam et vslam.

Le problème de compiler, et fixer des erreurs de la compilation est aussi ce que je rencontre. Heureusement, j’ai compilé réussitement le monoslam à partir de source Scene. Et j’ai échoué de le compiler dans l’autre source (MRPT). Pour vslam, il faut apprendre la plate-forme de ROS pour compiler le paquet vslam et l’odométrie visuelle. Mais malgré beaucoup de mes efforts, la compilation n’est pas réussie.

L’autre part, il faut profiter le support de mailing-list de chaque logiciel. Le mailing-list est un bon lieu pour poster des questions. Dans ce TP, le mailing-list de ROS est fréquenté que celui de MRPT.

En suite, le document de chaque logiciel n’est pas complet. Le document de Scene Lib est très peu. Celui de ROS est meilleur. Mais il n’y a pas de document des paquets qui sont en train développés.

**Difficulté du choix de caméra**

Dans le choix de caméra, il faut choisir la caméra avec laquelle on peut obtenir le meilleur résultat. Il faut utiliser la caméra qui a moins de fonctions automatiques. Il ne faut pas utiliser des appareils photos qui change son focal automatiquement.

**Difficulté de programmer entre deux systèmes différents**

Dans mon travail, je dois travailler avec la librairie Scene pour fonctionner le programme de monoslam. Ensuite, je dois extraire les données sous fichier 3D pour la simulation 3DScene de MRPT. Dans ce cas, je dois apprendre la librairie Scene et la simulation 3DScene de MRPT.

# Conclusion et Perspectives

## Contribution

Fixer les petites erreurs de Monoslam Scene version 1.0, et intégrer la simulation avec 3DScene Viewer de MRPT

Expériencier de la marche de monoslam

Mesurer et évaluer monoslam en particulier de l’environnement de l’IFI

## Perspectives

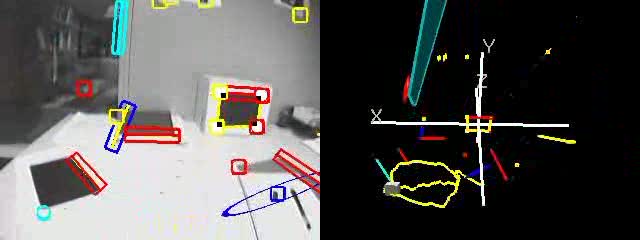
### Reconstruction la structure 3D de l’environnement

Il y a des recherches sur reconstruire la structure d’environnement en 3D. Et avec monoslam, on peut réaliser la reconstruction 3D. à partir des points d’intérêt, on peut reconstruire la structure d’environnement en 3D comme la figure suivante.



### Reconnaissance des lignes dans l’environnement

La carte construite à partir de monoslam est assez discrète. Pour avoir une carte mieux, on peut détecter des lignes dans l’environnement.



### Comparer avec l’autre méthode de SLAM visuel

L’autre méthode de slam visuel est vslam. Vslam est une méthode de SLAM visuel qui supporte la stéréoscoptique et la monoculaire. Vslam est la base de l’algorithme GammaSLAM et SigmaSLAM. D’après le mailing-list de la communauté de ROS, la nouvel implémentation de vSLAM sera fini à la fin d’été 2010. Pour comprendre mieux cette méthode, on peut réaliser les tests avec cet algorithme.

Il y a encore une autre implémentation de monoslam dans MRPT. D’après le mailing-list de la communauté de MRPT, la nouvel implémentation de monoslam sur MRPT sera fini à la fin de 2010.

Pour mieux comprendre tous les trois méthodes de SLAM (Monoslam sur Scene, vSLAM sur ROS, Monoslam sur MRPT), il faut comparer tous les trois algorithmes.

### La fermeture de boucle

Pour résoudre le problème de la fermeture de boucle, les auteurs de l’article [40] a utilisé la recherche de mot visuel sur le sac visuel quand l’incertitude devient très grande. Cette technique est aussi aide résoudre le phénomène de kidnapping de robot. Parce que, quand robot a l’incertitude grande, il doit faire les autres opérations, dans ce cas c’est la recherche de mot visuel, pour reconstruire la localisation. Les articles et vidéo relatives se situe à l’adresse <http://cogrob.ensta.fr/loopclosure.html> . Nous pouvons implémenter cette technique avec monoslam.

### Comparer l’odométrie visuelle et odométrie mécanique

Pour tester la correction de l’odométrie visuelle, la correction de la localisation de robot, nous pouvons comparer l’odométrie visuelle et odométrie mécanique. Ce travail peut etre réalisé par attacher la caméra sur un robot qui a l’odométrie mécanique. Et comparer des résultats d’odométrie visuelle par rapport d’odométrie mécanique.

A mon avis, deux propositions premières sont pour améliorer la présentation de la carte. Trois propositions dernières sont pour mieux comprendre l’algorithme de monoslam en particulière, et slam visuel en général. Donc, il faut prioriser trois propositions dernières.

## Conclusion

Ce rapport est pour l'état de l'art de SLAM en concentrant à SLAM visuel. Plusieurs de l'amélioration de la solution de problème de SLAM dans les années récentes. En particulier, la tendance de l'algorithme de SLAM visuel est de concentrer à utiliser une seule caméra pour résoudre problème de SLAM.

# Référence

1. R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” in Autonomous Robot Vehicles, I. Cox and G. Wilfong, Eds. Springer Verlag, New York, 1988, pp. 167 – 193.
2. S. Thrun, “Robotics mapping: A survey,” School of Computer Science, Carnegie Mellon University, Tech. Rep., Feb. 2002.
3. J. Guivant and E. Nebot, “Optimization of the simultaneous localization and map-building algorithm for real-time implementation,” IEEE Transactions Robotics and Automation, vol. 17, no. 3, pp. 242 – 257, 2001.
4. J. Guivant and E. Nebot, “Solving computational and memory requirements of feature based simultaneous localization and map building algorithms,” Australian Centre for Field Robotics, University of Sydney, Sydney, Tech. Rep., 2002.
5. F. Dellaert. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In Robotics: Science and Systems (RSS), 2005.
6. S. Thrun, D. Koller, Z. Ghahmarani, and H. Durrant-Whyte, “SLAM updates require constant time,” School of Computer Science, Carnegie Mellon University, Pittsburgh, Tech. Rep., 2002.
7. Mark A. Paskin, “Thin junction tree lters for simultaneous localization and mapping,” in Proceedings of the Eighteenth International Joint Conference on Articial Intelligence (IJCAI-03), G. Gottlob and T. Walsh, Eds., San Francisco, CA, 2003, pp. 1157–1164.
8. S. Thrun, “Probabilistic algorithms in robotics,” AI Magazine, vol. 21, no. 4, pp. 93–109, 2000.
9. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” J. Royal Statistical Society, Series B, vol. 39, no. 1, pp. 1–38, 1977.
10. G. J. McLachlan and T. Krishnan, The EM Algorithm and Extensions. New York: Wiley, 1997.
11. T. Duckett, S. Marsland, and J. Shapiro, “Learning globally consistent maps by relaxation,” in Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, 2000, pp. 3841–3846.
12. T. Duckett, S. Marsland, and J. Shapiro, “Fast, on-line learning of globally consistent maps,” Autonomous Robots, vol. 12, no. 3, pp. 287– 300, 2002.
13. U. Frese, P. Larsson, and T. Duckett, “A multilevel relaxation algorithm for simultaneous localisation and mapping,” IEEE Transactions on Robotics, vol. 21, no. 2, pp. 196–207, April 2005.
14. M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM 2.0: An improved particle ltering algorithm for simultaneous localization and mapping that provably converges,” in Proceedings of the Eighteenth Int. Joint Conf. on Articial Intelligence (IJCAI-03). San Francisco,
15. Michael Montemerlo and Sebastian Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In Proc. ICRA, 2003.
16. Dirk Haehnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In IROS, 2003.
17. Pantelis Elinas, Robert Sim, and James J. Little. SLAM: Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution. In Proc. 2006 IEEE ICRA, 2006.
18. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic bayesian networks. In 16th Conference on Uncertainty in AI, pages 176–183, 2000.
19. Stephen Se, Timothy Barfoot, and Piotr Jasiobedzki. Visual motion estimation and terrain modeling for planetary rovers. In Proc. ISAIRAS 2005, 2005.
20. Levin and R. Szeliski, “Visual odometry and map correlation,” in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), Washington, DC, USA, 2004.
21. Harris and M. Stephens, “A Combined Corner and Edge Detector, Proc”. Fourth Alvey Vision Conference, pp.147-151, 1988.
22. J. Shi and C. Tomasi, Good Features to Track, IEEE Conference on Computer Vision and Pattern Recognition, pp. 593- 600, 1994.
23. D. Nist´er. An Efficient Solution to the Five-Point Relative Pose Problem, IEEE Conference on Computer Vision and Pattern Recognition, Volume 2, pp. 195-202, 2003.
24. D. Nist´er. Preemptive RANSAC for Live Structure and Motion Estimation, IEEE International Conference on Computer Vision, pp. 199-206, 2003.
25. R. Haralick, C. Lee, K. Ottenberg and M. N¨olle, Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem, International Journal of Computer Vision,
26. Tim K. Marks, Andrew Howard, Max Bajracharya, Garrison W. Cottrell, and Larry Matthies. “Gamma-SLAM: Stereo Visual SLAM in Unstructured Environments Using Variance Grid Maps”. 2007.
27. M. Dailey and M. Parnichkun. Simultaneous localization and mapping with stereo vision. In Proc. ICARCV, 2006.
28. Henrik Andreasson, Tom Duckett and Achim Lilienthal, “Mini-SLAM: Minimalistic Visual SLAM in Large-Scale Environments Based on a New Interpretation of Image Similarity” in Robotics and Automation, 2007 IEEE International Conference. Apr, 2007
29. Pantelis Elinas, Robert Sim, and James J. Little. σSLAM: Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution. In Proc. 2006 IEEE ICRA, 2006.
30. N. Karlsson, E. D. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, “The vSLAM algorithm for robust localization and mapping,” in IEEE Int. Conf. on Robotics and Automation (ICRA), Barcelona, Spain, April 2005, pp. 24–29.
31. J. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in Proc. International Conference on Computer Vision, Nice, Oct. 2003.
32. D. Nister, O. Naroditsky, and J. Bergen, “Visual odometry,” in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), 2004, pp. 652–659.
33. J. Campbell, R. Sukthankar, and I. Nourbakhsh, “Techniques for evaluating optical ow for visual odometry in extreme terrain,” in IEEE/RSJ Int. Workshop on Robots and Systems (IROS-04), Sendai, Japan, October 2004.
34. R. Smith, M. Self, and P. Cheeseman, “A Stochastic Map for Uncertain Spatial Relationships,” Proc. Fourth Int’l Symp. Robotics Research, 1987.
35. A.J. Davison, “Real-Time Simultaneous Localisation and Mapping with a Single Camera,” Proc. Ninth Int’l Conf. Computer Vision, 2003.
36. N.D. Molton, A.J. Davison, and I.D. Reid, “Locally Planar Patch Features for Real-Time Structure from Motion,” Proc. 15th British Machine Vision Conf., 2004.
37. A.J. Davison, “Active Search for Real-Time Vision,” Proc. 10th Int’l Conf. Computer Vision, 2005.
38. H. Jin, P. Favaro, and S. Soatto, “A Semi-Direct Approach to Structure from Motion,” The Visual Computer, vol. 19, no. 6, pp. 377- 394, 2003.
39. Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann, “6D SLAM - 3D Mapping Outdoor Environments” Journal of Field Robotics (JFR), Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems, Wiley & Son, ISSN 1556-4959, Volume 24, Issue 8-9, pages 699 - 722, August - September, 2007.
40. Angeli, A., Filliat, D., Doncieux, S., & Meyer, J.-A. “A Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words”. IEEE Transactions On Robotics, Special Issue on Visual SLAM. 2008.