REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Automatique

Laboratoire de Commande des Processus

Mémoire de :

Master en automatique

Intitulé

Comparaison entre deux approches pour la résolution du problème du SLAM 3D

Réalisé par

MERADI Dounia

Sous la direction de **Mr.M .CHAKIR**

Soutenu publiquement le 26 juin 2016

Membres du Jury

Président :	Dr O.Stihi (ENP)
Encadreurs :	Dr M.Chakir (ENP)
Examinateur :	Pr D.Boukhetala (ENP)

ENP 2016

ملخص

لاستكشاف بيئة ديناميكية غير معروفة بالنسبة لروبوت متحرك، ويتضمن دراسة وز رع خوارزمية لتحديد الموقع ورسم الخريطة وهو عبارة عن المشكل الدجاجة والبيضة. وهو مجال بحث نشيط جدا من عشرون سنة وبالإمكان الآن العمل ب SLAM 2D. التحدي الجديد هو العمل ب SLAM 3D. في هذا العمل قدمنا طريقتان لحل مشكل SLAM مرشح ذري وGraph-based. درسنا مزايا وعيوب كل طريقة.

كلمات مفتاحية:SLAM 3D ، مرشح ذري، Graph-based

Abstract

For a mobile robot exploring an unknown dynamic environment, localizing itself and building a map at the same time is a chicken-or-egg problem, known as Simultaneous Localization and Mapping (SLAM). It has been a very active field of research for more than twenty years and we now have solutions to address the 2D SLAM problem. The new challenge is to tackle the 3D SLAM question. In this project we have presented two approaches to problem solving SLAM particulate filter (FastSLAM) and Graph-based. We studied the advantages and disadvantage of each approach.

Keywords: SLAM 3D, FastSLAM, Graph-based.

Résumé

Pour un robot mobile, explorer un environnement dynamique inconnu, se localise et construire une carte en même temps est un problème de poule-œuf connu sous le nom Localisation et de cartographie simultanée, C’est un domaine de recherche très actif depuis plus de vingt ans et il est maintenant possible de faire du SLAM en 2D. Le nouveau challenge est de faire du SLAM en 3D. Dans ce projet nous avons présenté deux approches pour résolution de problème du SLAM filtre particulaire (FastSLAM) et Graph-based. Nous avons étudié les avantages et les inconvénients de chaque approche.

Mots clés : SLAM 3D, FastSLAM, Graph-based.

Dédicace

À l'homme le plus cher au monde mon père,

À mon trésor ma mère,

À mon frère et mes Sœurs,

À mes cousins Imad et Moetaz,

A mes chers amies Nour El-Hoda Gabour et Aicha Laidoudi

À toute la Famille MERADI et BAHRI

À tous mes enseignants,

À mes amis et camarades.

Meradi Dounia

REMERCIEMENTS

Je tiens tout d’abord à exprimer ma reconnaissance envers les membres du jury, dont la renommée et la qualité scientifique honore grandement ce travail :

- Monsieur Omar SETIHI enseignant chercheur de l’école national polytechnique à Alger pour m'avoir fait l'honneur de présider le jury de ma thèse.
- Monsieur Djamel BOUKHATELA professeur de l’école national polytechnique à Alger de m'avoir fait l'honneur d'être examinateurs de ce travail.

Je tiens à exprimer toute ma gratitude aux rapporteurs Monsieur Messoud CHAKIR et le professeur Mr Mohamed TAJINE d'avoir proposé le sujet sur lequel j'ai travaillé, et qui ont assuré la direction et l'encadrement du travail présenté dans ce mémoire.

TABLE DES MATIERS

Table des figures

Chapitre 1

Chapitre 2

Liste des abréviations

EKF	**E**xtended **K**alman **F**ilter.
G2O	**G**eneral Framework for **G**raph **O**ptimization
ICP	**I**terative **C**losest **P**oint.
ORB	**O**riented FAST and **R**otated **B**RIEF.
RANSAC	**RAN**dom **SA**mple **C**onsensus.
RGB	**R**ed, **G**reen, **B**lue.
RGB-D	**R**ed, **G**reen, **B**lue–**D**epth.
SIFT	**S**calar **I**nvariant **F**eature **T**ransform.
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping.
SURF	**S**peeded **U**p **R**obust **F**eature.
TORO	**T**ree-based netw**OR**k **O**ptimizer

Introduction générale

La thématique de la navigation autonome constitue l'un des principaux axes de recherche dans le domaine de la robotique mobile.

Les robots doivent donc d'une part être suffisamment autonomes pour remplir leur mission sans adaptation à l'environnement et sans la supervision de l'Homme et, d'autre part être capable de fournir à l'utilisateur des informations sur la topologie du lieu, le plus souvent sous la forme d'une carte.

En effet, sans informations suffisantes sur la position du robot (localisation) et sur la nature de son environnement (cartographie), les autres algorithmes (génération de trajectoire, évitement d'obstacles ...) ne peuvent pas fonctionner correctement.

Pour résoudre le problème de SLAM, la littérature présente des approches différentes qui peuvent être classés soit comme le filtrage ou lissage. Les approches de filtrage présentent le problème comme une estimation d'état en ligne où l'état du système consiste la position actuelle du robot et la carte. L'estimation est augmentée et affiné en intégrant les nouvelles mesures qu'ils deviennent disponibles. Les techniques les plus courantes sont le filtre de Kalman étendu (EKF) et les filtres à particules [1]. Pour mettre en évidence leur nature incrémentale, les approches basés sur le filtrage sont généralement appelés méthodes de SLAM sur la ligne. L'inconvénient de ces techniques est le coût de calcul pour le filtre à particules car il maintient plusieurs des plusieurs hypothèses est des variables d'état, qui peuvent se développer rapidement lorsque le robot se déplace, tandis que l'EKF conserve une seule hypothèse.

Contrairement, les approches de lissage estiment la trajectoire du robot à partir de l'ensemble des mesures. Ces approches portent sur le soi-disant problème SLAM plein, et ils comptent généralement sur moins-carrés techniques de minimisation d'erreur. Leur performance aurait pu être aussi un problème dans le passé, mais maintenant ils sont une alternative intéressante. Dans ce travail, nous allons donc étudier les résultats qui peuvent être obtenus à partir d'une approche basée sur les graphes. Contrairement aux techniques en ligne, le graphique de pose est dit hors ligne ou une technique est lourde que le traitement d'optimisation est déclenché par des contraintes spécifiques.

Dans ce document, nous allons d'abord présenter le problème du SLAM, sa formulation, ses difficultés, les étapes de sa résolution ... Nous nous intéresserons ensuite aux solutions les plus connues pour ce problème. Dans le deuxième chapitre, nous détaillons deux approches

pour résoudre le problème du SLAM l'une des approches basée sur les méthodes de lissage et l'autre basée sur le filtrage.

1

SLAM en général

1.1 Introduction

Dans ce chapitre, nous allons présenter le problème du SLAM de manière générale, afin de mieux situer le contexte du travail du master. Nous commençons d'abord par définir le problème du SLAM (*Simultaneous Localization And Mapping*), ensuite formuler ce problème. Enfin, nous allons présenter différentes méthodes de résolution du SLAM.

1.2 SLAM

L'un des principaux objectifs de la robotique est de développer des robots qui disposent d'une autonomie presque totale à long terme dans des environnements non structurés et inconnus. Une telle autonomie ne sera atteinte grâce à des algorithmes qui permettent aux robots de percevoir, d'interpréter et d'interagir avec le monde qui l'entoure.

Le problème de localisation et cartographie simultanées (SLAM) traite deux questions importantes dans la robotique mobile. La première question est : « Où suis-je? ». La réponse à cette question définit la localisation du robot. La deuxième question concerne les caractéristiques de l'environnement du robot : « À quoi ressemble l'environnement où je me trouve? »[2].

1.2.1 Formulation du problème de SLAM

Le SLAM est composé d'un ensemble de méthodes permettant à un robot de construire une carte d'un environnement et en même temps de se localiser en utilisant cette carte. La trajectoire du robot et la position des amers[1] dans la carte sont estimées au fur et à mesure, sans avoir besoin de connaissances a priori.

Considérons un robot se déplaçant dans un environnement inconnu, en observant un certain nombre d'amers grâce à un capteur embarqué sur le robot. La figure 1.1 montre une illustration du problème.

A l'instant k on définit les quantités suivantes :

- x_k : le vecteur d'état. Il contient la position du robot.
- u_k: le vecteur de contrôle. L'application de u_k à l'instant $k-1$ mène le robot de l'état x_{k-1} à l'état x_k.
- m_i: vecteur contenant la position de l'amer i.
- z_k : l'observation à l'instant k.

On définit aussi les ensembles suivants :

- $x_{0:k} = \{x_0, x_1, ..., x_k\} = \{x_{0:k-1}, x_k\}$: l'ensemble des vecteurs d'état jusqu'à l'instant k.
- $u_{0:k} = \{u_0, u_1, ..., u_k\} = \{u_{0:k-1}, u_k\}$: l'ensemble des vecteurs de commande jusqu'à l'instant k.
- $z_{0:k} = \{z_0, z_1, ..., z_k\} = \{z_{0:k-1}, z_k\}$: l'ensemble des observations jusqu'à l'instant k.
- $m = \{m_1, m_2, ..., m_n\}$: la carte de l'environnement contenant une liste d'objets statiques.

1 : qui sont les points d'intérêt ou points de repère (points remarquables comme les coins).

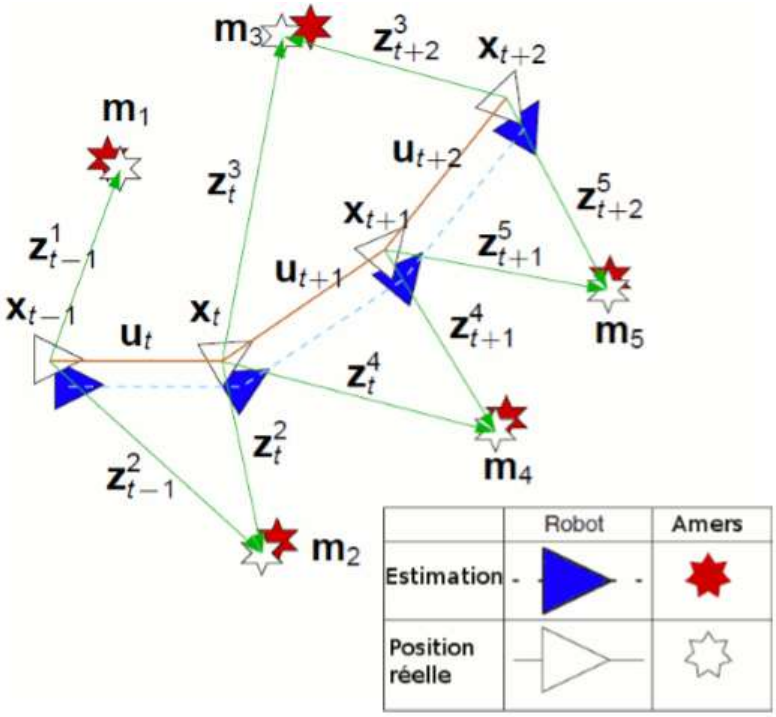

Figure 1.1 : l'idée de base du SLAM

1.2.1.1 La localisation

Le problème de localisation du robot consiste à estimer sa position dans un environnement donné, en utilisant l'historique de ses observations, l'historique des commandes et la connaissance de l'environnement. La figure 1.2 schématise ce principe.

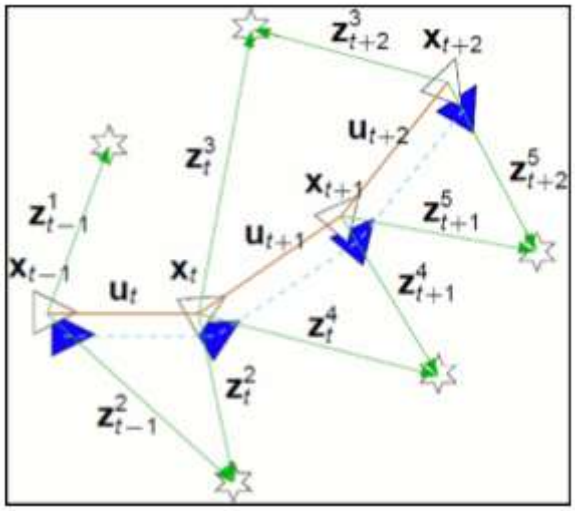

Figure 1.2 : La localisation : le système cherche à estimer sa position en utilisant les informations sur l'environnement dont il dispose

On peut analytiquement représenter cette opération par l'estimation de la probabilité de distribution :

$$P(x_k|Z_{0:k}, U_{0:k}, m) \quad (1.1)$$

L'estimation d'une telle quantité définit la localisation globale, dans la mesure où on utilise toutes les données de l'historique des observations et des commandes pour estimer la position. On obtient ainsi une estimation robuste de la position a postériori, mais on augmente largement la complexité des calculs.

Afin de simplifier l'algorithme, on peut définir une localisation locale, où on utilise uniquement les données de l'instant (k−1) pour estimer la position à l'instant k. On représente analytiquement cette opération par l'estimation de la distribution de probabilité :

$$P(x_k|z_{k-1}, u_{k-1}, x_{k-1}, m) \quad (1.2)$$

En utilisant cette méthode, on simplifie largement la complexité de l'algorithme, mais on risque de dévier de la position correcte du robot, sans pouvoir corriger cela.

1.2.1.2 La cartographie

Le problème de cartographie consiste à déterminer la carte d'un environnement, en utilisant les données des capteurs et l'historique des positions réelles du robot. Sur le schéma de la figure 1.3, le système connait sa position exacte et estime la carte de l'environnement en utilisant les données de ses capteurs.

On peut exprimer cela analytiquement ainsi :

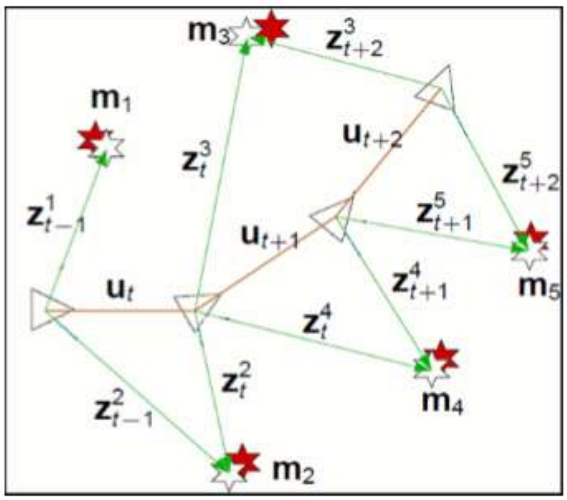

Figure 1.3 : La cartographie : le système crée la carte de l'environnement en se basant sur sa position connue et les informations de ses capteurs.

1.2.3 Le SLAM

La formulation probabiliste du problème de SLAM nécessite le calcul, à chaque instant k, de la quantité de probabilité :

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) \quad (1.3)$$

Ce calcul est généralement effectué récursivement. On commence par la probabilité $P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1})$, puis on utilise le théorème de Bayes pour déduire la quantité $P(x_{k,}m|Z_{0:k}, U_{0:k})$ à partir de z_k et u_k. Afin d'effectuer cette déduction, nous avons besoin de connaître $P(x_k|x_{k-1}, u_k)$ et $P(z_k|x_k, m)$. Le terme $P(z_k|x_k, m)$ désigne le modèle d'observation. Il définit la probabilité d'avoir une mesure z_k connaissant l'état du véhicule x_k et une carte de l'environnement m. Le terme $P(x_k|x_{k-1}, u_k)$ définit le modèle de transition (modèle de mouvement du véhicule robotisé). Il permet de prévoir l'état x_k du système, qui ne dépend que de l'état précédent x_{k-1} et de la commande de contrôle appliquée. Dans ce cas, le processus de transition entre les états du système $x_k$est dit Markovien.

On définit donc le problème du SLAM en deux parties par les équations 1.4 et 1.5 :

- Une partie de mise-à-jour de la position :

$$P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) = \int [P(x_k|x_{k-1}, u_k) * P(x_k, m|Z_{0:k}, U_{0:k}, x_0)\ ]\, dx_{k-1} \quad (1.4)$$

- Une partie de mise-à-jour de l'observation :

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \frac{P(z_k|x_k, m) * P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{k-1}, U_{0:k})} \quad (1.5)$$

En utilisant ainsi l'estimation a posteriori à l'instant (k − 1) donnée par le terme $P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$, on peut calculer la prédiction (équation 1.4) et déduire ensuite l'estimation a posteriori à l'instant k. On a donc :

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \eta * P(z_k|x_k, m) * \int [P(x_k|x_{k-1}, u_k) * P(x_k, m|Z_{0:k}, U_{0:k}, x_0)\ ]\, dx_{k-1} \quad (1.6)$$

Sachant que $\eta = \frac{1}{P(z_k|Z_{k-1}, U_{0:k})}$ est une constante de normalisation dépendant du modèle d'observation et du modèle de transition.

Cette structure du SLAM est représentée sur le schéma de la figure 1.3. Sur ce schéma, les cercles gris représentent les données connues, tandis que les cercles blancs désignent les quantités à estimer.

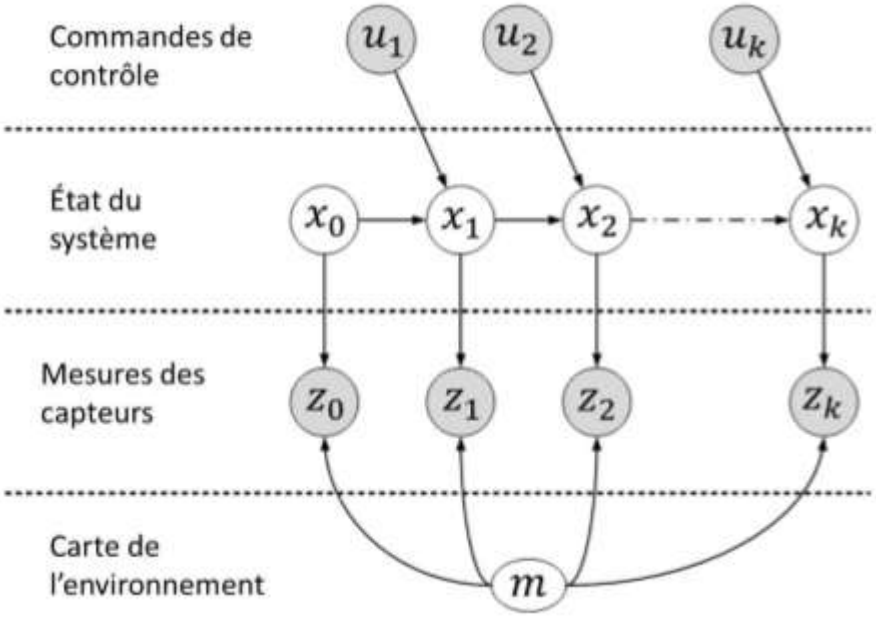

Figure 1.3 : Représentation graphique du problème de SLAM

1.3 Les différentes approches de SLAM

Nous allons présenter dans cette partie trois méthodes largement utilisées pour la résolution du problème de SLAM. Chaque méthode a des avantages et des inconvénients.

1.3.1 EKF-SLAM

Depuis les années 1980, plusieurs méthodes de traiter le problème du SLAM ont vu le jour. Sans doute la plus utilisée, reste le SLAM par filtre de Kalman étendu (*EKF : Extended Kalman Filter*) [3].

Elle utilise un vecteur d'état pour représenter la position du robot et des amers dans la scène, auquel est associe une matrice d'erreur représentant les incertitudes sur les positions, les observations et les corrélations entre les différentes variables du vecteur d'état. Cette matrice est mise à jour régulièrement à chaque fois que le robot change de position [2] par conséquence sa taille grandit quadratiquement en O (n2) (où n est le nombre d'amers de la carte).

Les inconvénients majeurs de cet algorithme c'est qu'elle est complexe, se base uniquement sur la probabilité gaussienne, En plus il atteindra un point où il ne pourra pas mettre à jour sa carte en temps réel et il n'est pas toujours facile d'extraire des amers correctes et intéressants dans un environnement non-structuré ou externe.

1.3.2 SLAM basé sur les filtres particulaires

Un filtre particulaire est un filtre récursif qui permet d'estimer l'état a postériori en utilisant un ensemble de particules. Le principe est de suivre un grand nombre d'hypothèses en parallèle qui

sont autant de trajectoires possibles. Ces différentes hypothèses correspondent à un échantillonnage de la distribution de probabilité des trajectoires.

L'utilisation du filtrage particulaire souffre de plusieurs problèmes à cause de sa complexité grandissante qui augmente exponentiellement avec le nombre d'amers de l'environnement et des difficultés rencontrées lors de la définition du nombre de particules.

1.3.3 Graph-based SLAM

Le *Graph-based SLAM* reste une solution intéressante grâce à sa simplicité et son efficacité en temps de calcul, L'idée principale est de construire un graph dans lequel les poses du robot sont modélisées par des nœuds qui sont reliés par des contraintes. Après la construction le graph doit être optimisé, pour se faire il existe des approches diverses comme HOG-Man, G2O, TORO **…..**[4].

Le gros avantage du Graph-SLAM est qu'il permet de gérer les cartes composées d'un très grand nombre de nœuds ce qui est impossible avec les autres techniques. Cependant l'optimisation du graphe peut être très lourde.

1.4 Conclusion

Dans ce chapitre, nous avons étudié une partie du problème du SLAM. Nous avons ainsi commencé par une présentation de la problématique générale de la localisation et la cartographie simultanées. Nous avons ensuite formulé le problème mathématiquement, afin de pouvoir l'intégrer dans le cadre probabiliste de sa résolution. Et la fin du chapitre nous présentons brièvement les différentes approches pour la résolution du SLAM.

2

Graph-based et le filtre particulaire

2.1 Introduction

Dans ce chapitre nous présentons deux approches pour résoudre le problème du SLAM, l'une des approches est basée sur le filtrage qui est la méthode filtre particulaire en utilisant la méthode FastSLAM, l'autre est basée sur le lissage qui est la méthode de graph-based.

2.2 Résolution par filtre particulaire : le FastSLAM [5]

Nous présentons dans cette section la résolution du problème du SLAM probabiliste avec un filtre particulaire Rao-Blackwellisé. Nous présentons dans un premier temps le principe général et les équations. Ensuite, nous listons les principaux avantages et inconvénients de cette méthode, avant de donner des résultats de convergence et de consistance.

2.2.1 Principe

2.2.1.1 Introduction

Nous avons vu précédemment que l'EKF-SLAM a une complexité algorithmique importante du fait qu'elle croît avec le carré du nombre d'amers M. En 2003, ce poids calculatoire était considéré critique; il était alors très intéressant de développer un algorithme de SLAM pouvant traiter de nombreux amers et de grandes trajectoires. L'algorithme de FastSLAM a été introduit dans cette optique et utilise une implémentation astucieuse de l'algorithme de filtrage particulaire.

L'algorithme de filtrage particulaire classique consiste à effectuer un échantillonnage aléatoire de tout l'état (dans le cas du SLAM, il s'agit de l'état du robot et des amers). Chaque échantillon est tiré selon une densité de probabilité choisie à l'avance, une étape de pondération utilisant le vecteur de mesure permet ensuite de donner plus d'importance aux échantillons (nommés par la suite particules) les plus probables. Une description détaillée concernant le fonctionnement général des filtres particulaires pourra être trouvée dans [6] [7].

L'algorithme de filtrage particulaire classique n'est pas utilisable dans le cas du SLAM. En effet, cet algorithme impose d'échantillonner selon toute la dimension de l'état (à savoir la position du robot et les positions des amers). Le nombre de particules nécessaires pour représenter convenablement l'espace des solutions devient trop important : les temps de calculs et l'espace mémoire nécessaires seraient rédhibitoires.

L'algorithme de FastSLAM est basé sur une factorisation du problème, qui permet ensuite d'utiliser une variante du filtrage particulaire : le filtre particulaire Rao-Blackwellised. Ce filtre est utilisé lorsque la densité de probabilité d'une partie de l'état peut être connue analytiquement conditionnellement au reste de l'état. Dans ce cas, l'échantillonnage particulaire n'intervient que sur la seconde partie de l'état [8]. Dans le cas du FastSLAM, l'idée sera de n'échantillonner que la position du robot.

2.2.1.2 Factorisation du problème de SLAM

On peut montrer que la structure du SLAM permet de factoriser la densité précédente de la manière suivante :

$$P(x_{0:k}, m|Z_{0:k}, U_{0:k}) = \underbrace{P(x_{0:k}|Z_{0:k}, U_{0:k})}_{\text{post trajectoire}} \prod_{i=1}^{M} \underbrace{P\left(m_{(i)}\middle|x_{0:k}, z_{0:k}, U_{0:k}\right)}_{\text{estimateur amers}} \quad (2.1)$$

La factorisation présentée dans l'équation 2.1 montre que conditionnellement à la trajectoire du robot, chaque amer est indépendant des autres.

2.2.1.3 FastSLAM

Le principe du FastSLAM est de tirer parti de la factorisation précédente dans l'utilisation d'un filtre particulaire à N particules.

Lorsque la densité d'une partie de l'état peut se déduire analytiquement de la densité de la seconde partie de l'état (par un filtre de Kalman linéaire dans le cas idéal), on peut limiter l'échantillonnage d'importance à la seconde partie de l'état et tirer partie de la relation existante pour estimer la première partie de l'état. Il s'agit de la Rao-Blackwellisation [8] [9].

Dans le cas du SLAM, cette propriété est utilisée : les densités de probabilités des amers sont obtenues conditionnellement à la trajectoire du robot (et ce de manière indépendante étant donné le caractère multiplicatif dans l'équation 2.1). En conséquence, l'échantillonnage particulaire ne porte que sur l'état du robot. Ensuite, un filtre de Kalman étendu permet d'obtenir une estimation des amers pour chaque particule. Plus intuitivement, le principe du FastSLAM est de générer stochastiquement N hypothèses concernant la trajectoire du robot. Pour chacune de ces hypothèses, on applique un filtre de Kalman par amer en utilisant le vecteur de mesures. Chaque hypothèse est enfin pondérée en fonction de sa vraisemblance avec les mesures.

2.2.1.4 Stratégies d'échantillonnage

Nous présentons ici deux algorithmes de FastSLAM. Ils différent par leur stratégie d'échantillonnage (méthode pour générer la trajectoire) :

FastSLAM 1.0

Le principe est d'échantillonner en utilisant la chaîne de Markov $P(x_k|x_{k-1}, u_k)$ Lorsque le bruit associé à la chaîne de Markov est gaussien et additif, on a $(x_k|x_{k-1}, u_k) = N(f\ (xt - 1, ut), Qt$. Dans ce cas, pour prolonger une particule donnée, il suffit d'appliquer la fonction f

(avec comme arguments la dernière valeur de la particule et le vecteur d'entrées) puis d'ajouter un échantillon généré selon $N(0, Q_t)$.

Le calcul des poids (non normalisés) est aisé avec ce type de filtre. Pour la particule [m], on a:

$$w_k^{[m]} \propto w_{k-1}^{[m]} p\left(z_k | x_{0:k}^{[m]}, z_{k-1}, u_{0:k}\right) \quad (2.2)$$

Cette stratégie n'utilise donc pas le vecteur de mesures pour échantillonner. Ceci peut être problématique si le modèle est ponctuellement mauvais (glissement non modélisé par exemple); dans ce cas, les particules seront toutes biaisées et le vecteur de mesure ne servira qu'à donner un poids important à la moins mauvaise particule.

FastSLAM 2.0

Le principe est d'échantillonner en utilisant le vecteur de mesures, ce qui permet d'améliorer la robustesse à des erreurs de modèle. Pour étendre la particule [m], on va appliquer un filtre de Kalman de la façon suivante :

- Etendre $x_{k-1}^{[m]}$ avec le modèle (fonction f). Le résultat obtenu est une prédiction $\hat{x}_k^{[m]}$ associée à la matrice de variances-covariances de modèle Q_t.
- appliquer une étape de correction de Kalman avec le vecteur de mesures. Le résultat obtenu est une estimation de l'état avec une matrice de variances-covariances.
- effectuer un tirage aléatoire gaussien avec pour paramètre le résultat de l'étape précédente.

Le calcul des poids est légèrement plus compliqué que pour le FastSLAM 1.0.

Cette stratégie d'échantillonnage améliore la qualité des échantillons. En conséquence, le nombre de particules nécessaires pour représenter correctement la trajectoire du robot est plus faible que pour le FastSLAM 1.0.

2.2.2 Avantages et limites pratiques

Le FastSLAM possède plusieurs avantages importants :

- l'approximation linéaire sur le modèle de mouvement est évitée,
- la complexité algorithmique est plus faible que pour l'EKF-SLAM. Dans le cas du FastSLAM 1.0.

Par exemple, on a un EKF à appliquer par particule et par amer. Etant donné que chaque filtre de Kalman a une dimension fixe, la complexité est au pire en O(NM). Ensuite, des améliorations algorithmiques permettent de la faire descendre en O (N log(M)),

- il est possible de résoudre explicitement le problème d'associations des données durant le filtrage. Nous n'avons pas mentionné cet aspect dans le paragraphe précédent. Le lecteur intéressé se réfèrera à [10] pour plus de détails. Rappelons tout de même que ceci n'est pas possible avec l'EKF-SLAM.

Par ailleurs, il existe un résultat de convergence pour le FastSLAM 2.0 :

Théorème : L'algorithme linéaire gaussien de FastSLAM converge vers la vraie carte avec M = 1 particule si tous les amers sont observés une infinité de fois et que la localisation d'un d'entre eux est connue à l'avance.

La démonstration de ce théorème peut être trouvée dans [10] pp 88–92. Ce résultat montre qu'il est possible de converger vers la bonne carte sans maintenir les corrélations entre les amers. Néanmoins, l'intérêt pratique de ce résultat est faible :

- la démonstration est faite pour le cas linéaire gaussien,
- il est seulement fait état de convergence finale. On n'a aucune information quant à la consistance au cours du processus de filtrage.

Malgré son avantage certain en termes de complexité algorithmique, le FastSLAM possède des inconvénients qui font que cette solution n'est pas idéale :

- le phénomène de dégénérescence des particules est problématique et nécessite de régulièrement rééchantillonner. Le moment adéquat n'est pas évident à déterminer. Dans sa thèse, Montemerlo effectue cette étape à chaque instant. D'autres auteurs définissent un seuil sur le nombre de particules effectives [11],
- le rééchantillonnage diminue la diversité des particules, ce qui pose problème car on raisonne en termes de trajectoire : l'élimination des particules conduit à rendre déterministe les parties plus anciennes de la trajectoire. La conséquence est la perte des informations de corrélation entre les amers anciennement observés,
- de par son aspect particulaire, le résultat final de l'algorithme de FastSLAM n'est pas clair.

En effet, on ne possède qu'un jeu de particules pondérées pour approcher une densité de probabilité; il reste à effectuer l'estimation finale. On peut choisir de retourner la particule avec le plus fort poids pour l'estimation, ou encore d'effectuer une moyenne pondérée. Par ailleurs, l'obtention d'une information sur la qualité du résultat est délicate. Par exemple, comment donner une incertitude sur la position si on n'utilise que la particule la plus forte? De plus, les matrices de variances-covariances associées à la carte sont données conditionnellement à la trajectoire de la particule. Elles ne reflètent pas directement l'incertitude globale sur les amers.

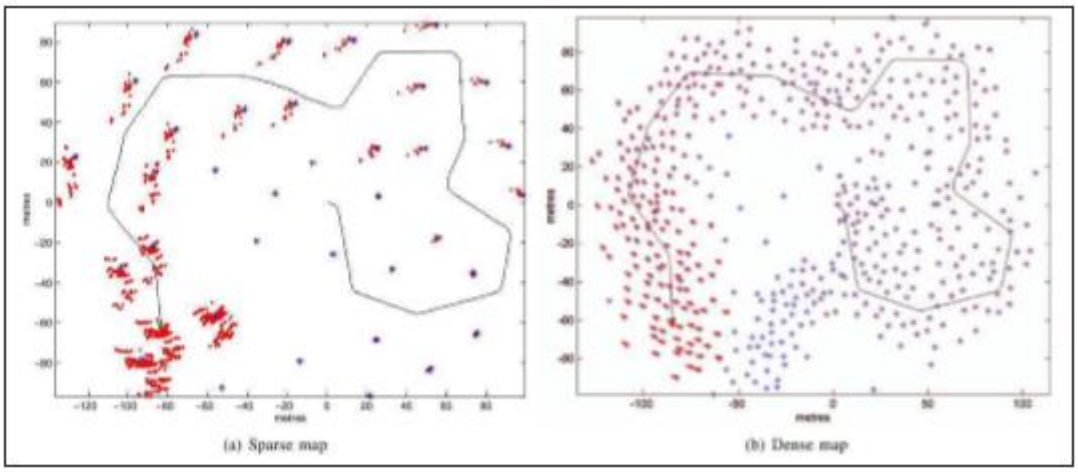

Figure 2.1 : Les deux cartes utilisées (bleus) avec les résultats d'un filtre particulaire à 1000 particules (rouge) [12]

La fusion de toutes les " cartes particulaires " peut être très délicate, surtout si on laisse l'algorithme effectuer l'association des données (l'association des données est valable pour une particule donnée; certaines particules peuvent par exemple avoir plus d'amers que d'autres si certains appariements n'ont pas été trouvés). En général, cet aspect n'est pas abordé dans la littérature.

2.2.3 Consistance

Dans ce paragraphe, nous présentons une étude de la consistance de l'algorithme de FastSLAM 2.0 réalisée dans [12]. Deux points sont abordés :

1. la diversité des particules
2. la capacité de l'algorithme à être consistant

Pour les deux aspects, tous les résultats sont obtenus en simulation, en testant quatre configurations censées illustrer les cas d'utilisation possibles. Les auteurs testent deux densités de carte (cf. figure 2.1) et deux nombres de particules (N = 100 et N = 1000) Dans ce qui suit, les résultats de simulations sont obtenus avec les mêmes modèles de prédiction et de mesure que dans l'étude de consistance de l'EKF-SLAM.

2.2.3.1 Diversité des particules

Par " diversité ", nous entendons le nombre d'hypothèses restantes parmi les hypothèses particulaires initiales. La perte de diversité des particules vient du fait que le filtre élimine les particules de poids trop faible et duplique celles de poids fort. Ceci contribue à éliminer des hypothèses pour le placement de certains amers, donnant alors " l'impression " que la précision de ces amers est améliorée.

Les auteurs montrent empirement ce problème en représentant le nombre d'hypothèses initiales encore présentes à chaque instant (cf. figure 2.2). Ainsi, il ne reste plus que quelques hypothèses sur le placement des amers initiaux en moins de 40s. Ceci posera problème si la boucle doit être fermée et que l'hypothèse unique retenue pour ces amers ne correspond pas parfaitement au moment de la fermeture de boucle.

Par ailleurs, les auteurs de la publication remarquent que cette décroissance est plus rapide dans le cas d'une carte dense, lorsqu'il y a donc beaucoup de mesures. Ce résultat peut s'expliquer de manière analogue à celui du " mauvais comportement " bien connu des filtres particulaires lorsque les mesures sont trop précises par rapport au modèle. Dans notre cas, chaque mesure permet de mettre à jour le poids de chaque particule. Le caractère indépendant des mesures fait que cette mise à jour de poids est multiplicative. En conséquence, une particule peu probable aura plus d'incréments multiplicatifs proches de zéro dans le cas d'une carte dense; le déséquilibre des poids est donc plus important lorsque le nombre de mesures est important. Ceci explique intuitivement pourquoi la diversité diminue plus rapidement lorsqu'on ajoute des mesures.

2.2.3.2 Consistance

Une étude de la consistance analogue à celle présentée pour l'EKF-SLAM est faite par les auteurs de [12]. Etant donné que le résultat d'un filtre particulaire n'est qu'un jeu de particules, nous présentons d'abord quelles données sont utilisés pour effectuer les tests de consistance :

Estimation de l'état : calcul de la moyenne pondérée des particules

Incertitude de l'estimation : calcul de la variance empirique des particules avec la moyenne pondérée des écarts quadratiques à la moyenne.

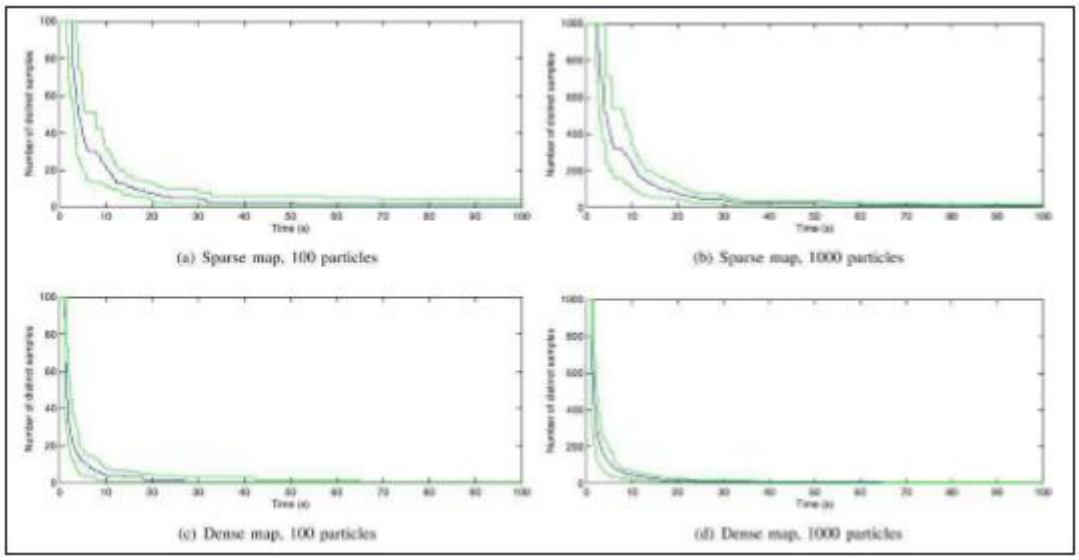

Figure 2.2 : Perte de diversité des particules calculée sur 50 simulations : valeur moyenne, bornes max et min [12]

La figure 2.3 compare la variance de l'erreur d'estimation (sur 50 simulations) à la variance moyenne fournie par le filtre. On remarque que le filtre est inconsistant très rapidement. Au bout de 200s, la variance prédite baisse brutalement. Ceci correspond à la fermeture de boucle: pratiquement toutes les hypothèses sont éliminées. Seules quelques particules très concentrées subsistent, ce qui provoque la baisse de la variance prédite.

La figure 2.3 met également en évidence le fait qu'un nombre plus élevé de particule permet de rendre le filtre plus consistant. Enfin, une carte moins dense permet d'obtenir de meilleurs résultats.

Ces résultats semblent être liés à la diversité des particules.

Un calcul de consistance avec l'indice NESS est également effectué dans la publication étudiée et montre clairement l'inconsistance du filtre.

2.2.3.2.1 Conclusion sur la consistance

Au final, l'algorithme FastSLAM est inconsistant. La raison qui semble expliquer ce phénomène est la perte de diversité des particules. Cette perte de diversité entraîne un oubli de l'incertitude du début de la trajectoire.

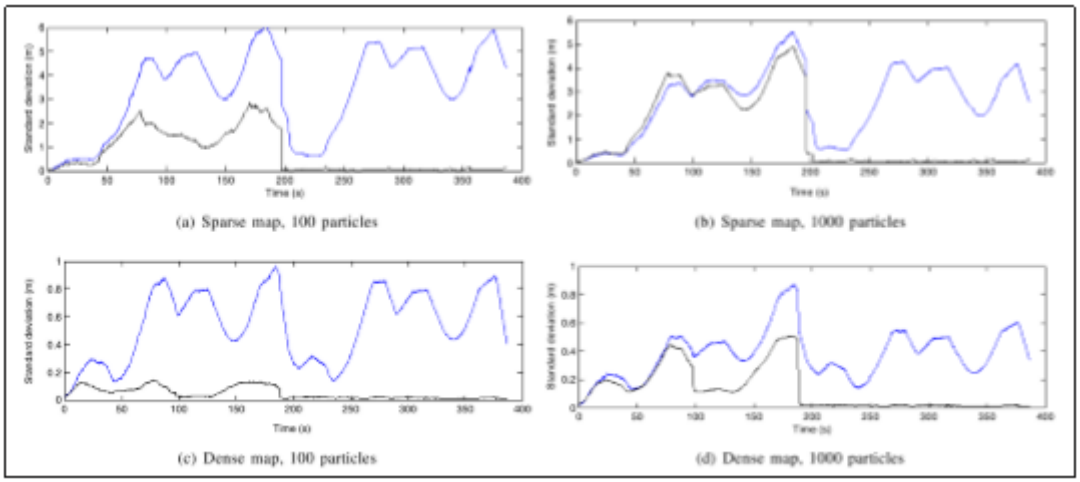

Figure 2.3 : variance de l'erreur du filtre (courbe bleu) et variance prédite courbe noire [12]

Ce mauvais résultat du filtre particulaire est logique. Le problème de dégénérescence des particules est en effet bien connu. En pratique, il n'a pas d'influence sur le résultat pour les systèmes qui oublient exponentiellement leur passé. Ce n'est malheureusement pas le cas du SLAM : les anciennes erreurs subsistent dans la carte.

Ainsi, les auteurs de [12] estiment que le FastSLAM n'est valable que pour l'initialisation et ne doit pas être utilisé pendant très longtemps. Pourtant, l'intérêt du FastSLAM est sa rapidité en termes de temps de calculs pour les grandes cartes.

2.3 Graph-based SLAM

L'algorithme de GraphSLAM tire profit du caractère épars de la matrice d'information associée à l'état (trajectoire du robot et amers). Ceci provient de l'interprétation graphique du problème du SLAM, et notamment du fait que conditionnellement à toute la trajectoire, les amers sont indépendants entre eux (propriété qui était déjà utilisée dans le cadre du FastSLAM). Le principe du GraphSLAM est de définir un état qui contient toutes les positions depuis l'instant initial ainsi que la position des amers.

2.4 Exemple d'un algorithme qui utilise Graph-based SLAM

2.4.1 VSLAM

Le grand défi pour un robot autonome c'est être capable d'interagir avec son environnement, la localisation et la cartographie simultanée avec le capteur Kinect embarqué sur le robot, appelé RGB-D SLAM ou V-SLAM, a été développé à cet effet, ce qui déclenche de nombreuses exigences pour les algorithmes de traitement de l'information. Par conséquent,

la gestion des données de capteur est très importante.

Capteur Kinect

Le capteur Kinect est une barre horizontale reliée à une petite base avec un pivot motorisé, conçu pour être placé au-dessus ou en dessous de l'affichage vidéo (téléviseur, écran d'un ordinateur). Le dispositif comporte une caméra RGB, un capteur de profondeur constitué par un projecteur laser et une caméra infrarouge (IR) et un microphone ; Comme le montre la figure suivante :

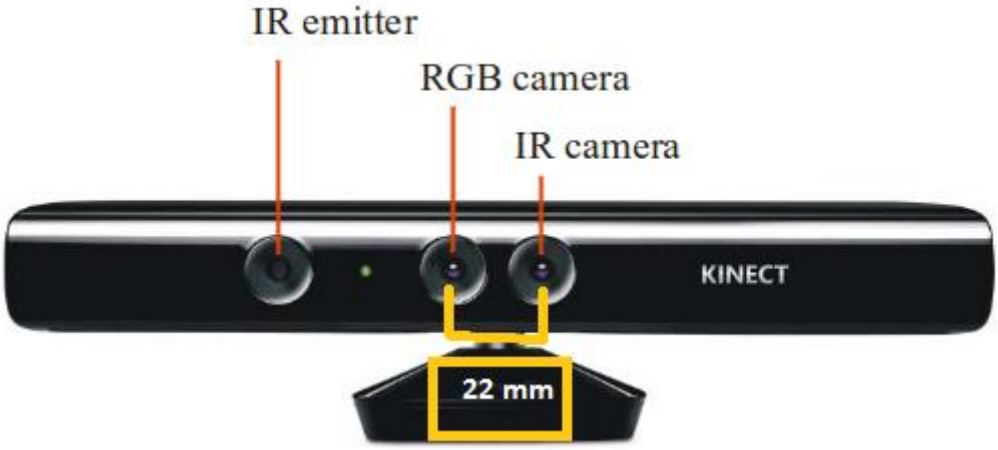

Figure 2.4 : Face avant de la Kinect : caméra émettant une mire de points dans l'infrarouge, caméra infrarouge, caméra couleur et réseau de microphones

Le V-SLAM est divisé en deux parties : "Front-end" et "back-end». Le "front-end" permet de construire la carte, et le "back-end" permet d'optimiser cette carte.

2.4.1.1 Front-End

Entre deux itérations la Kinect permet l'acquisition de deux images RGB et de deux images de profondeur de la scène observée. Ensuite, La détection des points d'intérêts et l'extraction des descripteurs sont faits à partir des deux images couleurs, des algorithmes performants permettent de mettre en relation ces images à partir de ces points, grâce aux images de profondeur, les points d'intérêts sont projetés dans l'espace 3D et forment des nuages, l'association des points de ces deux nuages donne une estimation de la pose en calculant la transformation entre ces deux nuages.

Les algorithmes du front-end sont montrés dans la figure suivante:

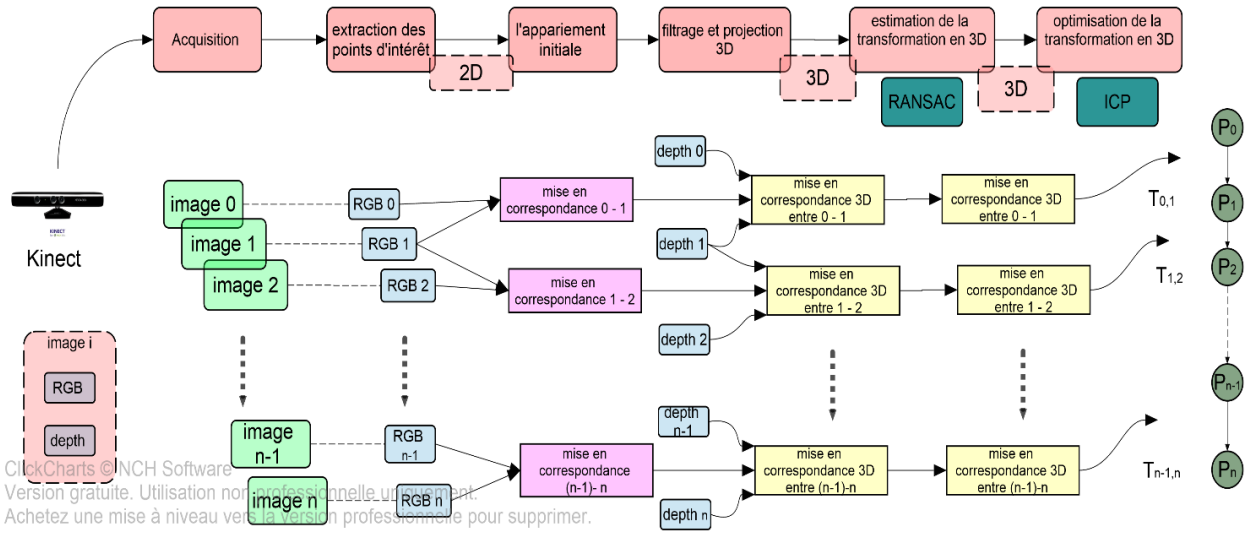

Figure 2.5 : Les algorithmes de front-end de RGB-D SLAM

La détection des points d'intérêts et l'extraction des descripteurs

Point d'intérêt

Un point d'intérêt est un point qui caractérise de façon unique une partie de l'image. Plus précisément il s'agit d'un point localisé finement, augmenté de son voisinage. Pour être utile ce point doit être très bien identifié à l'aide d'un descripteur unique et il doit être possible de le retrouver facilement. Il existe une multitude de méthode de détection/description de points d'intérêt parmi lesquelles : SIFT [14], SURF [15] et ORB [13]. Il s'agit de méthodes particulièrement appréciées pour leur robustesse et leur efficacité.

L'avantage de la Kinect est de fournir à la fois des images couleurs et des images de profondeur. Il y a deux aspects concernant le calcul des points d'intérêt : la détection d'un point d'intérêt lumineuse particulière et offrant une grande stabilité, qui identifie une zone d'intérêt, et son descripteur, qui caractérise cette zone. Typiquement, le détecteur identifie une région contenant une forte variation d'intensité lumineuse, Le centre de cette région représente un point d'intérêt. Le descripteur est généralement calculé en mesurant les principales orientations des points environnants, conduisant à un vecteur multidimensionnel qui identifie le point d'intérêt donné, Ceci se fait aisément avec la bibliothèque Open CV.

La mise en correspondance en 2D

Une fois les points sont calculés, il est possible de mettre en correspondance les deux images. La première mise en correspondance est faite en 2D par la recherche du plus proche voisin,

Cette étape consiste à mettre en relation les deux images à partir des descripteurs calculés précédemment en cherchant la distance euclidienne minimale entre les points.

Pour augmenter la robustesse, les points d'intérêt dont le rapport de la distance du plus proche voisin et la distance du deuxième proche voisin est supérieure à un seuil donné sont rejetés.

Estimation de la transformation

Après avoir déterminé les coordonnées 3D des points, On dispose donc de deux nuages en 3D formés par les points d'intérêt, la transformation du mouvement peut être estimée Selon la position des points dans les deux nuages ce qui permet d'estimer efficacement les déplacements de la camera entre deux instants.

La recherche de la transformation qui permet de recoller convenablement ces deux nuages de points se fait avec la combinaison RANSAC [16] plus ICP[17].

L'algorithme RANSAC qu'est utilisé pour déterminer les points qui permettent d'estimer cette transformation entre les deux nuages le plus correctement possible.

L'estimation de mouvement avec RANSAC est en générale assez bonne mais dans le cadre de la construction d'une carte il est préférable d'affiner cette estimation avec une autre méthode robuste comme ICP. Ainsi on obtient une première estimation du mouvement avec une méthode rapide puis on l'affine par la suite avec une méthode précise initialisée avec ce résultat. On bénéficie donc d'une vitesse de calcul relativement bonne et d'une précision correcte du fait du second algorithme.

Nous supposons que le robot est dans une position initiale notée P_{init}. Ensuite le robot se déplace dans un environnement vers une nouvelle position P_{fin} par rapport au repère de référence. La différence approximative de la position à partir de la position P_{init} (translation et rotation relatives) est habituellement connue par l'information odométrique. Cependant, cette information est souvent imprécise et cela est dû au patinage des roues. Notre tâche consiste à déterminer la vraie position du robot par rapport au repère de référence en utilisant les données issues de notre capteur la Kinect.

Dans le cas de l'approche globale, la position initiale P_{init} est toujours définie par $(0,0,0)$ qui correspond à la première image captée par la caméra. Alors que dans l'approche locale, P_{init} désigne la position du robot.

Soit P^i et P^{i+1} l'ensemble des points $3D$ qui appartiennent à $R^{3\times N} \times R^{3\times N}$, associés aux points d'intérêt extraits, pour deux images acquises en deux instants consécutifs, alors pour tous k=1,2,3,…,N. Chaque point $P_k^i \in R^{3\times 1}$ de P^i correspond aux points $P_k^{i+1} \in R^{3\times 1}$ de P^{i+1}.

Pour chaque paire de points associés P^i et P^{i+1}, l'objectif est d'estimer la matrice de

transformation T de dimension (4×4) rotation R de dimension (3×3) et le vecteur de translation t de dimension (3×1) expriment le mouvement du robot entre deux acquisitions successives. Tels que :

$$P_k^{i+1} = RP_k^i + t \tag{2.3}$$

Avec

$$P_k^i = \begin{bmatrix} X_k^i \\ Y_k^i \\ Z_k^i \end{bmatrix} \tag{2.4}$$

$$P_k^{i+1} = \begin{bmatrix} X_k^{i+1} \\ Y_k^{i+1} \\ Z_k^{i+1} \end{bmatrix} \tag{2.5}$$

La matrice de transformation

Connaissant la pose initiale, la première étape consiste à déterminer une estimation de n'importe quelle pose après une succession de transformations. Considérant une séquence finie d'itérations [image0, …, imageN], soit P_k la pose liée a k^{eme} itération tel que k ∈ [0; N], Cette pose peut être représentée par une transformation homogènes 4x4, où R est une matrice de rotation 3x3 et t est un vecteur de translation:

$$P_k = \begin{bmatrix} & & & t_x \\ & R & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Pour i > 0, nous avons la transformation T_{i-1}^i qui liée la position P_{i-1} à la position Pi. Chaque transformation T est représentée par une matrice homogène 4x4. Si P_0 détermine la position de départ, on peut alors calculer la position Pi en combinant toutes les transformations :

$$P_k = \prod_{i=k}^{1} T_{i-1}^i \, P_0 \tag{2.7}$$

Comme un choix arbitraire, nous pouvons définir la position initiale d'être à l'origine du système de coordonnées, qui est donnée par la I_4 de la matrice d'identité.

$$P_0 = I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Estimation de la pose 3D : Théoriquement, trois paires de points suffisent pour estimer la matrice de transformation. Cependant, compte tenu des bruits dans la reconstruction 3 et des erreurs dans l'association des points d'intérêts qui entachent les données, il est nécessaire d'utiliser un nombre de paires 3. Nous pouvons formuler le critère d'estimation de la pose, comme la minimisation de l'erreur quadratique moyenne de la distance définie entre les points 3 reconstruits. Cette erreur est une fonction de la matrice de rotation et le vecteur de translation. Le critère à minimiser s'écrit comme suit :

$$F(R,t) = \frac{1}{N}\sum_{k=1}^{N}\left\|\left(RP_k^{i+1} + t\right) - P_k^i\right\|$$

2.4.1.2 Back-End

Dans le chapitre précédent, nous avons vu comment déterminer une transformation entre deux itérations. A partir des transformations, une estimation de pose de la caméra peut être calculée. Chaque pose est ensuite insérée dans un graphe sous forme d'un nœud. Une fois une fermeture de boucle est détectée, de nouvelles contraintes peuvent être insérés dans le graphe, Cette nouvelle contrainte permet de recalculer la configuration du graphe et de réduire considérablement l'erreur accumulée depuis le départ, Le graphe peut alors être optimisée en minimisant une erreur, et les poses sont mises à jour en fonction des nouvelles données après l'optimisation, ce qui réduit la dérive globale.

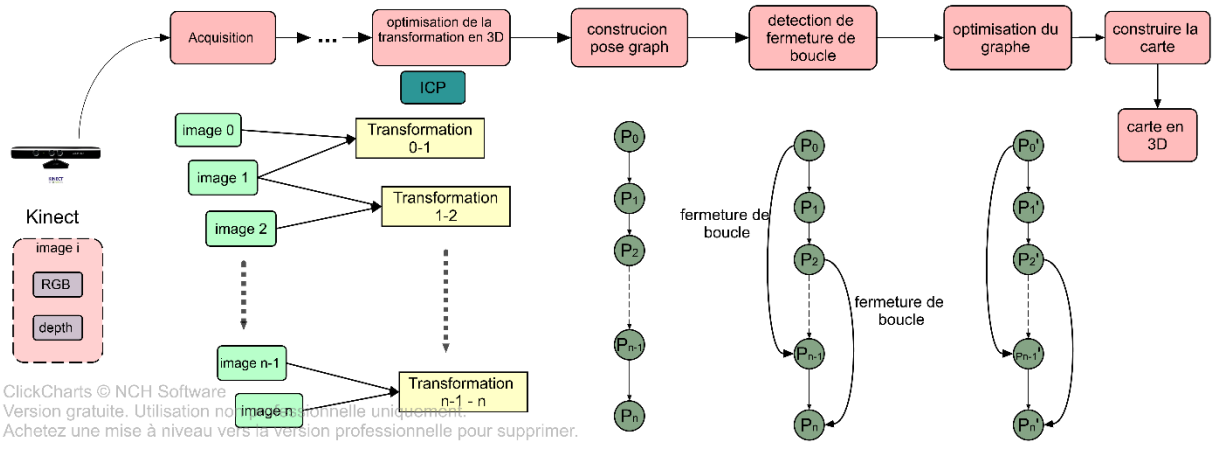

Figure 2.6 : Les algorithmes de back-end de RGB-D SLAM

La fermeture de boucle

Pour chaque pose, un nœud est inséré dans un graphe avec une contrainte reliant la nouvelle pose avec la précédente. La contrainte représente la transformation entre les deux itérations.

Afin de minimiser l'erreur, l'optimisation du graphe repose sur des contraintes entre les nœuds. Un événement intéressant se produit lorsque le robot se déplace à une position déjà visité dans le passé. Ceci est référé par la détection de fermeture de boucle. Ceci est illustré sur la figure 2.6. A chaque fois une fermeture de boucle est détectée entre deux poses, une nouvelle contrainte est insérée dans le graphe.

Sans faire une hypothèse sur la trajectoire suivie par le robot, et simplement en gardant l'histoire des images passées, il est possible de vérifier si l'image actuelle correspond à une image passée. Si l'observation actuelle contient suffisamment de similitudes avec une autre observation vue dans le passé, une transformation peut être calculée entre ces images et une nouvelle contrainte peut être insérée de celle-ci. Ceci peut être répété plusieurs fois. Grâce à ces nouvelles contraintes, l'erreur cumulée du graphe peut être considérablement réduite.

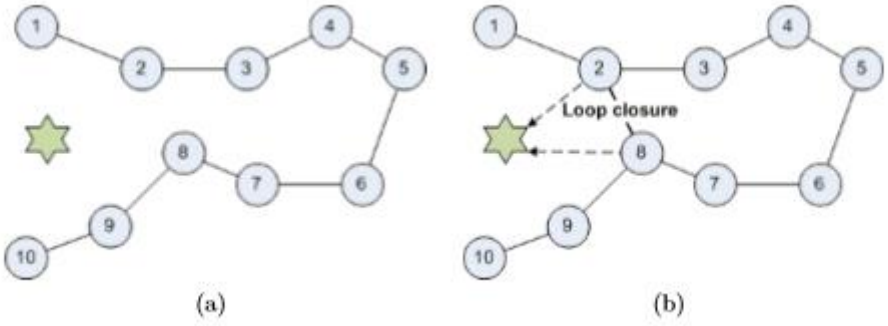

Figure 2.7: Détection de fermeture de boucle. (a) Les poses sont reliées entre elles par des contraintes, chacune d'elle représente une transformation. (b) L'observation d'un point d'intérêt commun déjà vu dans le passé, peut déclencher la création d'un nœud

Optimisation du graph

La détection de la fermeture de boucle conduit à des contraintes nouvelles qui sont insérés dans le graphe reliant les poses connexes. L'optimisation du graphe consiste à trouver la configuration optimale des nœuds en tenant compte toutes les contraintes données, comme illustré sur la figure suivante :

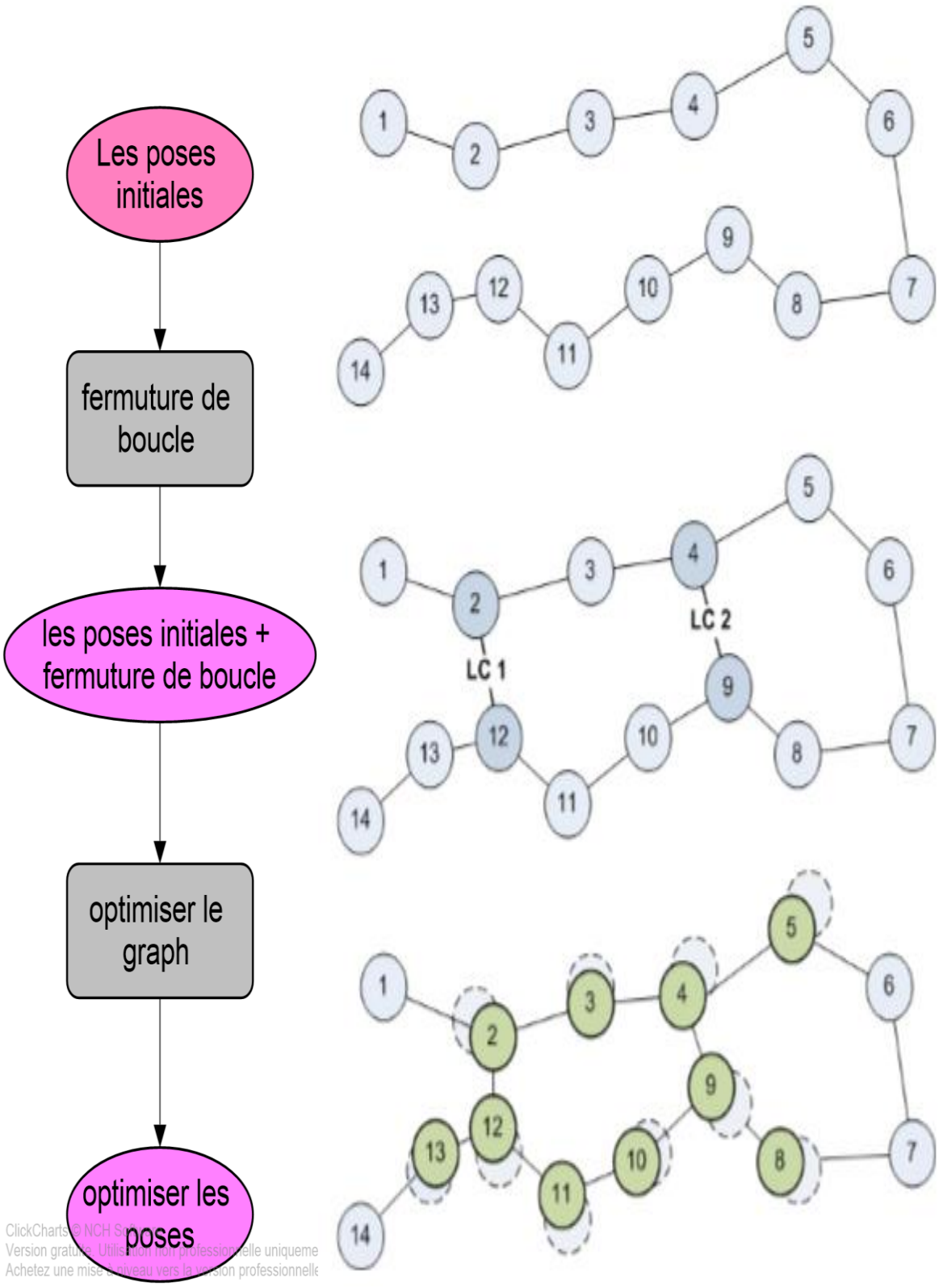

Figure 2.8 : la procédure d'optimisation du graphe

Une fois le graphe a été optimisé, les nœuds sont corrigés et les nouvelles estimations de la pose de la caméra peuvent être extraites.

2.5 Conclusion

Dans ce chapitre nous avons détaillés deux approches de SLAM. Nous avons étudié dans la première approche filtre particule la méthode de FastSLAM, on a écrire les avantages et las limites de cette approche, ensuite la deuxième approche qui est le Graph based, nous avons détaillé dans cette partie la méthode de VSLAM qui utilise cette approche.

Conclusion générale

La localisation et cartographie simultanées (SLAM) est l'un des problèmes les plus fondamentaux, encore plus difficiles dans la robotique mobile. Il existe plusieurs méthodes pour la résolution du problème de SLAM. Chaque méthode a des avantages et des inconvénients. Dans ce mémoire nous avons détaillés deux approches largement utilisés qui sont le filtre particule et le graph-based.

Nous avons détaillés deux exemples qui utilisent ces approches. D'abord l'algorithme FastSLAM qui a des avantages que l'approximation linéaire sur le modèle de mouvement est évitée et que la complexité algorithmique est plus faible que pour l'EKF-SLAM ; on peut résumer les avantages et les inconvénients de filtre particulaire FastSLAM comme suit :

Les avantages :

- Distribution non-gaussiennes,
- Robustesse accrue grâce à l'échantillonnage.

Les inconvénients :

- Grand espaces (complexité),
- difficultés de paramétrage,
- Association de données => objet plus discriminants,
- Problèmes ouvert à environnement non structurés et naturels, environnement dynamique.

La deuxième approche est *Graph-based*, on a détaillons la méthode de V-SLAM qui utilise cette approche, *Graph-based SLAM* est une solution intéressante grâce à sa simplicité et son efficacité en temps de calcul mais l'optimisation du graphe est très lourde.

Bibliographie et référence

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series). Intelligent robotics and autonomous agents. The MIT Press, August 2005.

[2] Oussama El Hamzaoui, Localisation et cartographie simultanées pour un robot mobile équipé d'un laser à balayage : CoreSLAM, l'école nationale supérieure des mines de Paris, 2012.

[3] P. Maybeck, ''The kalman filter: An introduction to concepts, Autonomous Robot Vehicles'', pp. 194-204, 1990.

[4] VIRGILE HÖGMAN, Building a 3D map from RGB-D sensors, Computer Vision and Active Perception Laboratory Royal Institute of Technology (KTH), Stockholm, Sweden.

[5] Cyril Joly, Contributions aux méthodes de localisation et cartographie simultanées par vision omnidirectionnelle. Thèse de doctorat, école nationale supérieure des mines de Paris, juin 2010.

[6] Arnaud Doucet, Simon Godsill et Christophe Andrieu : On sequential monte carlo sampling methods for bayesian filtering. Statistics and Computing, 10(3):197–208, 2004.

[7] Cyril Joly, David Bétaille et François Peyret : Etude comparative des techniques de filtrage non-linéaire appliquées à la localisation 2D d'un véhicule en temps réel. Traitement du Signal, 5 (3):201–220, 2009.

[8] Arnaud Doucet, Nando de Freitas, Kevin Murphy et Stuart Russell : Rao-blackwellised particle filtering for dynamic bayesian networks. In Uncertainty in Artificial Intelligence, 2000.

[9] GEORGE CASELLA et CHRISTIAN P. ROBERT : Rao-Blackwellisation of sampling schemes. Biometrika, 83(1):81–94, 1996. URL http://biomet.oxfordjournals.org/cgi/content/abstract/83/1/81.

[10] Michael Montemerlo : FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association. Thèse de doctorat, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[11] Carine Hue, Jean-Pierre Le Cadre et Patrick Pérez : ´Etude comparative des techniques de filtrage non-linéaire appliquées à la localisation 2D d'un véhicule en temps réel. IEEE Transactions on Aerospace and Electronic Systems, 28(3):791–812, 2002.

[12] Tim Bailey, Juan Nieto et Eduardo Nebot : Consistency of the fastslam algorithm. In IEEE International Conference on Robotics and Automation, 2006b.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: an efficient alternative to sift or surf,” in Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 2564–2571, IEEE, 2011.

[14] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International journal of computer vision, vol. 60, no. 2, pp. 91–110, 2004.

[15] Herbert Bay, Tinne Tuytelaars et Luc Van Gool, « SURF: Speeded Up Robust Features », Proceedings of the European Conference on Computer Vision, 2006, p. 404-417.

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

[17] P.J. Besl and N.D McKay, "A Method for Registration of 3D Shapes", Proc. of IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, pp.239-256, 1992.