Une première approche du SLAM (Localisation et cartographie simultanées pour un robot)

Culture Sciences de l'Ingénieur

Camille KESSELER

Édité le 20/10/2023

école normale supérieure paris-saclay

Cette ressource est issue d'un travail personnel de Camille Kesseler, élève du Département d'Enseignement et de Recherche Génie Mécanique de l'ENS Paris-Saclay. Ce travail a été effectué lors de son master d'ingénierie à Seoul National University en Corée du Sud.

L'acronyme SLAM vient de l'anglais *Simultaneous Localisation And Mapping* et signifie littéralement localisation et cartographie simultanées. On peut commencer par définir les trois termes importants présent dans ce nom. La Localisation est le fait de connaitre sa position dans l'espace. La Cartographie est le fait de créer une représentation de son environnement. Et simultanées signifie que l'on veut faire en même temps les deux actions. On peut donc déjà en déduire le sens de cet acronyme : on veut créer une carte de son environnement inconnu tout en se positionnant dans le même temps correctement dans cet environnement.

On utilise principalement le terme pour parler du problème SLAM ou d'algorithmes de SLAM, les deux sont intimement liées, le deuxième étant la solution proposée au premier. Dans cette ressource nous allons nous concentrer sur la description de différents problèmes SLAM ainsi que sur des éléments simples de résolution pour les algorithmes.

Cette ressource a pour objectif principal de présenter le problème SLAM, son but, ses grands principes et enjeux ainsi que plusieurs exemples d'utilisation pour introduire le sujet a un débutant complet en robotique. On se concentre donc sur la compréhension des problématiques, en mettant de côté la théorie mathématique et le code.

1 - Le problème SLAM

L'introduction nous laisse supposer que l'on peut séparer le problème en deux questions :

- Localisation : Où suis-je ?
- Cartographie : À quoi ressemble mon environnement ?

Pour bien appréhender les concepts de SLAM, il faut d'abord pouvoir comprendre les deux problèmes séparément. Et donc dans un premier temps, nous allons les détailler tout en les illustrant d'exemples concrets.

1.1 - Localisation

On cherche ici à déterminer la position d'un robot dans un environnement connu pour répondre à la question : Où suis-je ?

C'est-à-dire, par exemple, je sais où se trouvent les murs, les arbres, la route et les objets dans la rue et je veux savoir où le robot se trouve au cours du temps dans son déplacement[1].

Il existe de nombreux problèmes différents de localisation avec chacun leurs difficultés propres. On peut introduire des dénominations pour essayer de caractériser et organiser ces problèmes de manières quasi-systématiques. Déterminer le type de problème auquel nous faisons face est toujours la première étape pour le résoudre, et voici quelques questions importantes à se poser :

La position initiale du robot est-elle connue ?

OUI

Localisation locale - Odométrie

Nous cherchons donc à **suivre la position** du robot en partant d'un point connu en s'accommodant des erreurs de mesures.

On suppose que l'erreur sur la position reste petite et locale autour de la position calculée.

NON

Localisation globale

Le problème devient plus difficile que la localisation locale car on ne peut pas capitaliser sur une première position certaine, ce qui entraine de plus large erreur.

Une variante est **le problème du robot kidnappé** pendant son fonctionnement et déposé à une autre position. On teste ainsi la capacite à se rétablir dans le cas d'une défaillance.

Figure 1 : Piétons et voitures sont des environnements dynamiques, source [1]

L'environnement change-t-il ?

OUI

Environnement statique

Les seules variables sont les quantités définissant la position du robot.

NON

Environnement dynamique

Des objets ont leurs caractéristiques qui changent, cela peut être leur position mais aussi leur couleur ou toutes autres grandeurs mesurées par les capteurs.

[1] Dans un cadre mathématique plus formel, bien que simplifié, on va dire que j'ai $[(x_{mi}, y_{mi})]_{i\in\{0,n\}}$ les coordonnées des n élément de mon environnement (murs, arbre, objets, …) précisément définies. Et je veux connaitre $[(x_R^t, y_R^t)]_{t\in\{0,T\}}$ les coordonnées de la position de mon robot au cours du temps pendant son déplacement.

Contrôle-t-on en même temps le déplacement du robot ?

NON

Approche passive

L'algorithme regarde seulement le robot bouger, pour faire des taches journalières préprogrammées par exemple ou faire des mouvements aléatoires.

OUI

Approche active

L'algorithme contrôle le robot pour minimiser les erreurs de localisation et améliorer le déplacement dans un environnement hasardeux.

Les résultats sont meilleurs pour la localisation mais l'implémentation est plus difficile.

Figure 2 : Essaim de robots, source [2]

Plusieurs robots sont-ils impliqués ?

NON

Robot seul

Le cas le plus commun où l'on utilise un seul robot, ce qui a l'avantage de centraliser les informations reçues sans avoir de problème de communication.

OUI

Multirobots

Il est possible d'utiliser chaque robot individuellement sans se soucier des autres (équivalent à un problème de robot seul avec environnement dynamique).

Mais si on a la possibilité de faire communiquer les robots entre eux, il est utile de centraliser toutes les informations disponibles car on peut corriger les erreurs de mesures d'un robot grâce aux autres.

Il est bien sûr possible de définir encore plus précisément le problème, notamment par exemple en stipulant le type d'information utilisé pour reconnaitre son environnement, mais en répondant à ces premières questions on peut déjà voir se dégager des pistes, des choix plus pertinents que d'autres pour la résolution. Sans entrer dans les détails, on peut lister des types algorithmes couramment utilisés pour des problèmes de localisation. Le lecteur peut se renseigner plus précisément sur eux avec des exemples de code Matlab pour en tester certains sur cette page web[2].

Pour la plupart les algorithmes sont basés sur de la localisation de Markov. L'hypothèse est que le mouvement du robot peut être représenté sous forme de probabilité avec du bruit. En fonction du modèle choisi pour le mouvement du robot, on obtient des algorithmes différents [8] :

[2] https://www.mathworks.com/help/nav/localization-algorithms.html

- Filtres de Kalman (SLAM basé filtrage probabiliste) : La position du robot est représentée par la moyenne et la variance et se base sur des probabilités gaussiennes ;
- Filtres avec des représentation discrète ou en grille (SLAM basé sur l'optimisation de graphe) : Décomposition de l'environnement en une grille avec une certaine résolution et la probabilité de la position est un histogramme ;
- Filtres à particule ou Monte Carlo algorithme (SLAM non paramétrique) : La position du robot est composée d'une multitude de positions possibles (particules) qui sont déplacées et pondérées.

1.2 - Cartographie

Dans cette partie, on connait les positions du robot dans le temps pendant son mouvement mais on ne connait pas son environnement (y a-t-il des arbres, des murs, … ?), ni la position de ces éléments dans l'espace autour du robot.

C'est en pratique, un cas fréquent car la plupart du temps on ne connait pas à l'avance la configuration des lieux. Être capable de reconstituer une représentation de ce qui entoure le robot est important pour une utilisation dans un contexte réel.

La position du robot dans un repère global $[(x_R^t, y_R^t)]_{t\epsilon\{0,T\}}$ est parfaitement connue (en réalité cela n'arrive jamais mais nous le considérons comme une hypothèse dans ces applications) dans un environnement inconnu. À chaque instant, le robot utilise ses capteurs pour obtenir des informations sur les alentours, ces données sont ensuite traitées pour créer une carte. Comme pour le problème de localisation, il existe différents problèmes de cartographie qui sont plus ou moins compliqués. On peut les caractériser selon différents critères :

- Taille de la carte
- La marge d'erreur
- Les parcours en boucle
- L'ambiguïté perceptuelle

Taille de la carte

Plus l'environnement est grand par rapport la distance de perception du robot, plus il est compliqué d'obtenir la carte.

Figure 3 : Cartographie d'un intérieur de maison, source [3]

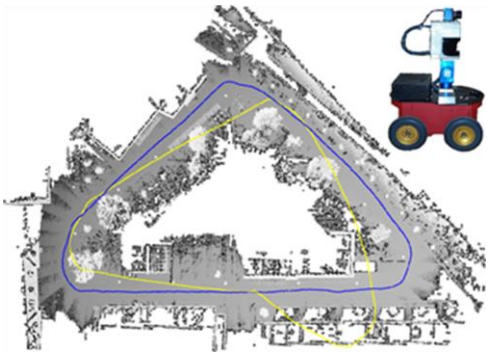

Figure 4 : Cartographie d'un parcours extérieur, source [4]

La marge d'erreur dans les capteurs et moteurs

S'il n'y avait pas d'erreur de mesure, créer la carte ne serait pas un problème. Bien entendu plus il y a d'erreurs de mesure plus il est compliqué de créer la carte.

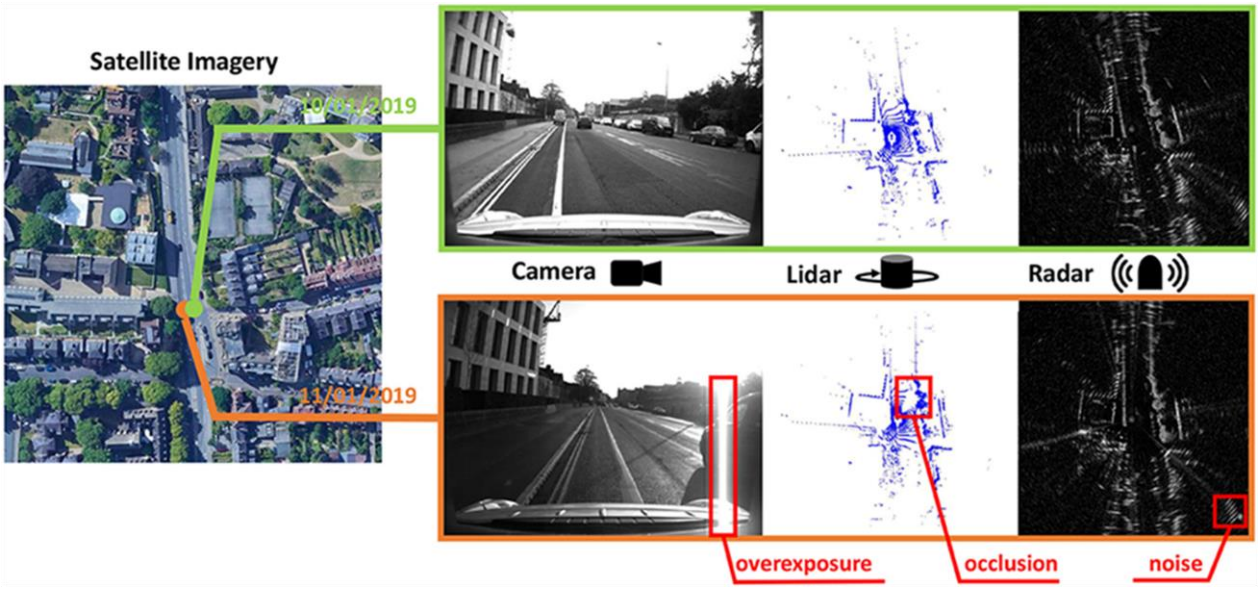

Figure 5 : Exemples de différentes incomplètes données de mesure capteurs, Source [5]

Les parcours en boucle

Créer une carte à partir d'un mouvement de va et vient dans le même corridor est plutôt facile car on corrige les erreurs au fur et à mesure. Mais si le robot suit un parcours en boucle, il revient à sa position de départ par un chemin différent que celui emprunté à l'aller, et, très souvent, on obtient d'énormes erreurs de mesure cumulées quand on ferme la boucle.

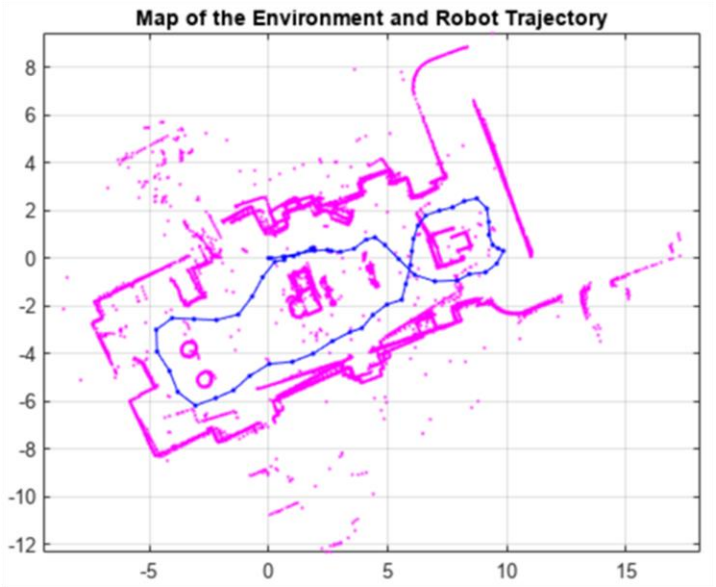

Figure 6 : Exemple de trajectoire avec un parcours en boucle

L'ambiguïté perceptuelle

Il arrive que des endroits différents se ressemblent beaucoup, et plus cela arrive plus est difficile d'établir une carte.

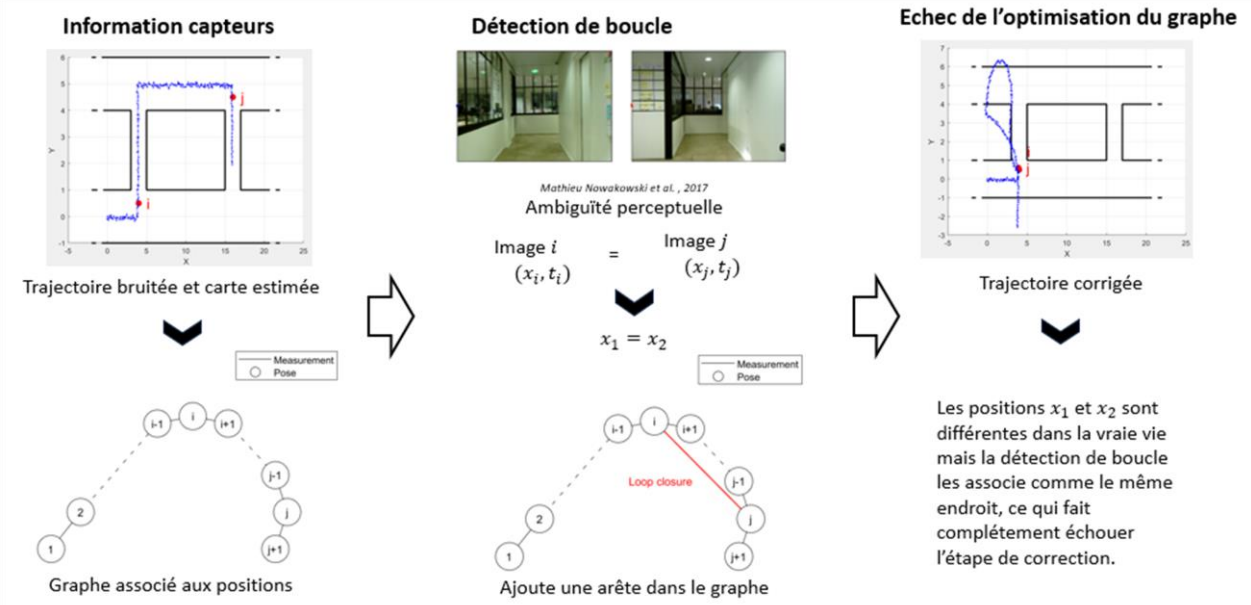

Figure 7 : Exemple d'échec d'association de données engendrant l'échec du SLAM

D'autres critères entrent en compte tel que les types de capteur sur le robot. Les algorithmes pour résoudre les problèmes de cartographie sont souvent classés en fonction du type de carte produit puisque les outils nécessaires à leur création peuvent grandement varier. On détaillera plus loin les différents types de carte, et on se concentre pour l'exemple sur une famille d'algorithme de cartographie avec poses connues : grille cartographique d'occupation ou *occupancy grid mapping*. Les algorithmes fonctionnent avec les cartes sous forme de grille fine sur l'espace continu des positions. On peut citer trois sous-types d'algorithmes basés sur la manière dont on met à jour la carte d'occupation : Algorithmes de mise à jour basés sur les capteurs (*Sensor-Based Updating*), algorithme de Bayes (*Occupancy grid*) et algorithme de fusion de cartes (*Map Fusion*).

1.3 - SLAM, ou l'union de la cartographie et de la localisation

Dans le cas où l'on veut résoudre un seul des deux problèmes, il existe des solutions donnant des résultats très satisfaisants. Malheureusement, la résolution de l'un des problèmes nécessite presque tout le temps de supposer l'autre problème déjà résolu. Et lorsque l'on veut résoudre les deux simultanément, on se retrouve dans une impasse : déterminer où se situe le robot précisément, nécessite de connaitre la carte de son environnement, mais construire la carte de l'environnement, nécessite de savoir où le robot se situe. Ce problème de dépendance des deux problématiques peut se voir comme un paradoxe (qui n'est pas sans rappeler le paradoxe de qui est apparu en premier entre l'œuf ou la poule).

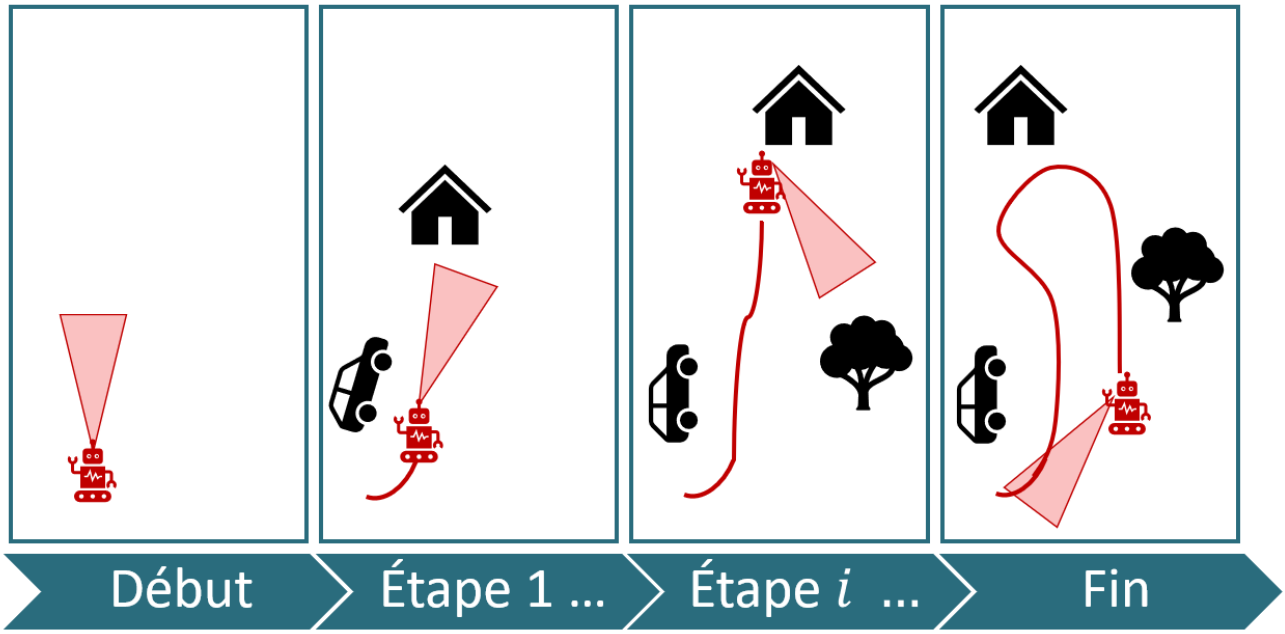

Figure 8 : Schéma des étapes d'un exemple de problème SLAM

Le schéma de la Figure 8 illustre très basiquement le fonctionnement voulu d'un algorithme SLAM. Classiquement, on commence avec peu ou pas d'information sur l'environnement et pas de position ou trajectoire du robot (Début). Puis le robot se déplace et prend des informations capteurs, ce qui lui permet de commencer à construire une carte et à estimer sa trajectoire (Étape 1). On répète ensuite la même opération en ajoutant un système de mise à jour de la carte et de la trajectoire pour corriger d'éventuelles erreurs. Par exemple la position de la maison a été mal estimée à l'étape 1 mais corrigée dans les étapes suivantes (Étape i). Finalement quand le robot a terminé son mouvement, on obtient une carte et une trajectoire générées en temps réel (Fin) ; ici on peut supposer que le robot reconnait sa position de départ et donc détecte un parcours en boucle pour plus encore corriger les erreurs. Il faut faire attention au fait que la carte et la trajectoire sont calculées relativement l'une à l'autre. Si on veut avoir une trajectoire absolue (mouvement par rapport à la terre), il faut faire un ancrage par rapport aux mondes, c'est-à-dire fixer une position par rapport à la terre.

À ce jour, on peut définir deux problèmes principaux qui apparaissent lors de la résolution du problème et en font un problème complexe. Le premier est que la carte et la position du robot sont inconnues et **corrélées l'une à l'autre**. Prendre en compte et conserver cette corrélation pendant la résolution du problème est extrêmement compliqué informatiquement parlant. Le deuxième problème est celui de **l'association de données** car le robot ne connait pas exactement ce qu'il voit et il est donc possible qu'il associe un élément détecté par ses capteurs à un autre élément précédemment mémorisé même s'ils sont en réalité deux éléments complètement différents. Cela peut entrainer le robot à devenir excessivement confiant dans sa position et un échec complet de la résolution du problème.

2 - Des exemples plus concrets

Les premiers processus technologiques SLAM ont vu le jour en 1981, ce qui en fait une technologie relativement nouvelle. Suite à la rapide augmentation de la puissance de calcul des processeurs, au développement de nouveaux algorithmes et la multiplication des codes open sources en ligne, on utilise maintenant le SLAM dans un grand nombre d'applications où l'on veut créer une carte 3D de son environnement.

2.1 - L'exemple du robot ultime : l'être humain

Le problème SLAM a vocation à être appliqué à des robots mobiles principalement et si l'on a vu précédemment que sa résolution parfaite restait un challenge à relever. On connait tous un exemple de « robot mobile intelligent » qui résout le problème SLAM naturellement et rapidement : l'être humain.

Tous les jours, à chaque instant nous nous positionnons dans notre environnement tout en le mémorisant (si nous nous concentrons dans l'hypothèse d'un être humain sans diminution physique ou mentale). Ce qui revient actuellement à résoudre en temps réel un problème SLAM, sans même le savoir ou faire d'effort particulier.

Au premier abord, l'être humain est bien différent des robots mobiles classiques que nous avons vu dans la partie précédente en exemple. Mais pourtant en y regardant de plus près, on peut voir se dégager des similitudes entre des groupes d'éléments répondant à la même problématique. Pour résoudre le problème SLAM, l'être humain utilise ses cinq sens afin d'obtenir des informations, son cerveau pour les analyser puis il se déplace en utilisant ses muscles. Le processus est quasiment instantané et sans effort pour un humain alors que les robots développés aujourd'hui sont loin de pouvoir rivaliser en termes de performance. Si l'on pousse encore l'analogie plus loin, on voit se dessiner des raisons : les cinq sens forment un incroyable réseau de capteurs très perfectionnés, la formidable capacité d'apprentissage du cerveau humain et des muscles permettant des déplacements précis.

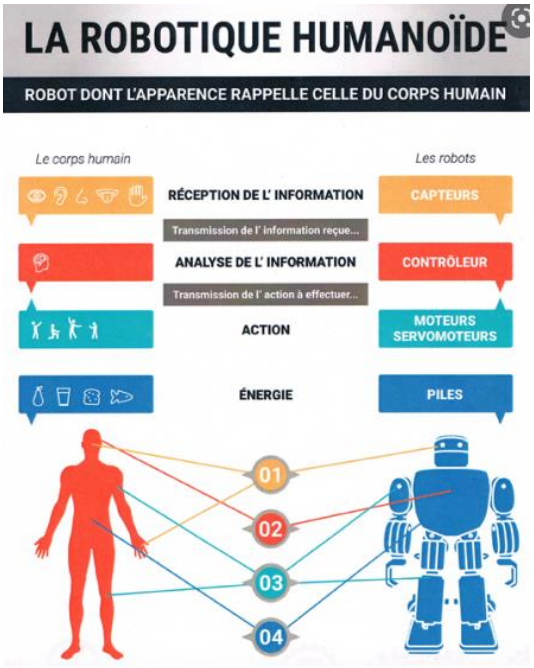

Figure 9 : Analogies robotique humanoïde-corps humain, source [6]

De part cette analogie, on peut commencer à voir se dégager de potentiels axes de résolution du problème SLAM pour un robot mobile. Car si la robotique tend parfois à copier la nature, d'autres fois essayer une autre approche est nécessaire.

2.2 - La voiture autonome

L'application la plus connue des algorithmes de SLAM est sans aucun doute la voiture autonome ou voiture à conduite automatisée. D'abord vue dans les films de science-fiction, elle est maintenant présente sur nos routes dans une certaine mesure comme, par exemple, avec le régulateur de vitesse. L'objectif principal de la voiture autonome est de se déplacer d'un point à un autre toute

seule sans intervention humaine. Ce problème est typiquement un problème de localisation et cartographie simultanées car la voiture doit se déplacer dans un environnement non connu tout en créant la carte pour détecter de potentiels dangers et en se localisant pour savoir où continuer son chemin pour arriver à point final.

Généralement pour résoudre le problème de SLAM en temps réel, on sépare le processus en deux parties : le front-end et le back-end. Dans la première partie (*front-end*), la voiture équipée de capteurs traite leurs informations et créé une version de la trajectoire mais avec des erreurs. Dans la second partie (*back-end*), la voiture essaye de corriger ses erreurs et produit une carte avec la mémoire des lieux préalablement visités, ou d'autres informations.

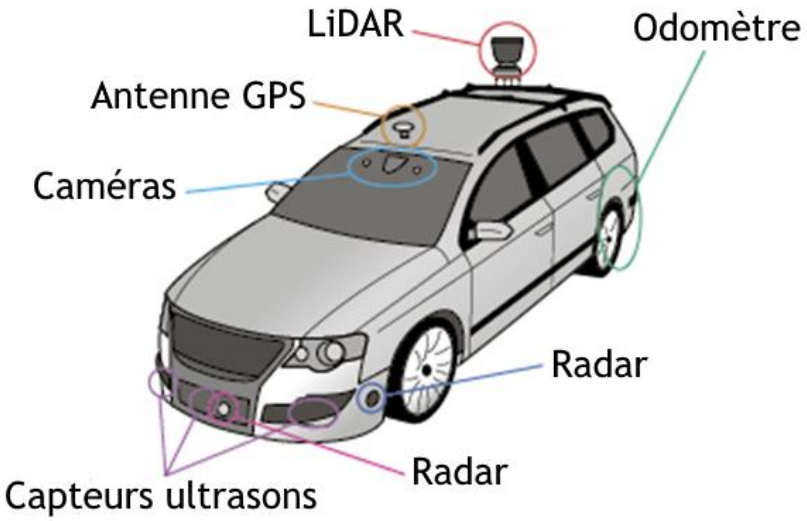

Figure 10 : Schéma des capteurs d'une voiture autonome, source [7]

Une voiture autonome a beaucoup de capteurs pour obtenir un maximum d'informations sur son environnement et des informations de plusieurs types (image, son, ondes, …). Toutes ces informations sont ensuite traitées par un ordinateur central embarqué qui va prendre les décisions de conduite. Les commandes sont ensuite envoyées au moteur et à la direction qui imposent le mouvement aux roues pour bouger la voiture.

3 - Les possibles axes de résolution

Comme avec la localisation et la cartographie, il existe de multiples façons de résoudre un problème SLAM. On peut cependant essayer de *séparer en grande partie la résolution du problème*. La première étape est bien sûr de définir le plus précisément possible le problème que nous voulons résoudre (la partie 1 a donné les premiers éléments pour cela). Ensuite la deuxième étape est de connaitre les entrées et sorties de l'algorithme pour décider d'une solution adaptée, elles peuvent être imposées par le choix du robot ou simplement une décision personnelle. Mais rentrons plus en détails sur ce que sont les entrées et sorties dans un problème SLAM. Les entrées sont les données collectées, elles sont l'unique source d'information que le robot possède. Les sorties sont la carte et la position du robot dans cette carte. Après ces définitions, on peut choisir parmi les différentes catégories d'algorithmes disponibles pour résoudre le problème.

3.1 - Les données et les capteurs

Dans le contexte du SLAM, les données sont essentielles pour permettre au robot de se localiser et de cartographier son environnement. Les données peuvent être collectées à partir de divers capteurs, notamment :

- **Capteurs LiDAR** : Ces capteurs utilisent des lasers pour mesurer la distance entre le robot et les objets environnants. Les données LiDAR[3] sont couramment utilisées pour créer des nuages de points 3D de l'environnement.
- **Capteurs de vision** : Les caméras sont souvent utilisées pour capturer des images de l'environnement, qui peuvent être traitées pour extraire des caractéristiques et des informations sur la position du robot.
- **Capteurs inertiels** : Les accéléromètres et les gyroscopes mesurent les mouvements du robot, ce qui peut être utilisé pour estimer sa position et son orientation.
- **Capteurs d'odométrie** : Ces capteurs mesurent le déplacement du robot en fonction de la rotation de ses roues, ce qui peut aider à estimer la trajectoire.
- **Capteurs GPS** : Les récepteurs GPS peuvent fournir des informations sur la position globale du robot en extérieur. Ce type de capteur ne fonctionne pas à l'intérieur de bâtiments.
- **Capteurs ultrasons** : Ils sont utilisés pour mesurer la distance entre le robot et les objets à proximité en émettant des signaux sonores.

3.2 - Les cartes et la position du robot

L'une des composantes fondamentales du problème de SLAM est la création et la gestion de cartes qui représentent l'environnement dans lequel évolue un robot autonome. Ces cartes jouent un rôle essentiel dans la navigation, la planification de trajectoire et la prise de décision du robot, lui permettant de comprendre et d'interagir avec le monde qui l'entoure. Dans cette section, nous explorerons les différents types de cartes utilisés dans le domaine du SLAM, ainsi que leur rôle crucial dans la résolution de ce problème complexe. De la représentation des obstacles à la modélisation de la probabilité de localisation, les cartes sont à la fois des outils de visualisation essentiels et des éléments clés dans la résolution du puzzle de la localisation simultanée et de la cartographie. Plongeons donc dans l'univers des cartes dans le SLAM pour comprendre comment elles contribuent à l'orientation et à la navigation de robots autonomes :

- **Carte topographique** : Une carte topographique représente l'environnement avec des détails géographiques, y compris les contours de terrain, les rivières, les routes, les bâtiments, etc. Elle est souvent utilisée comme référence pour la cartographie SLAM.
- **Carte d'occupation probabiliste (Occupancy Grid Map)** : Cette carte structure l'environnement en une grille de cellules et attribue à chaque cellule une probabilité d'occupation. Elle est couramment utilisée pour représenter la présence d'obstacles dans un environnement.
- **Carte de primitives (Feature Map)** : Au lieu de représenter l'environnement avec une grille, cette carte identifie et enregistre les points d'intérêt, tels que les coins, les bords ou d'autres caractéristiques distinctives. Elle est souvent utilisée dans les approches basées sur la vision.
- **Carte de grille de nuage de points (Point Cloud Grid Map)** : Cette carte représente l'environnement sous forme de nuage de points 3D, où chaque point correspond à un point de données LiDAR ou de capteur 3D. Elle permet une représentation détaillée de l'environnement en 3D.
- **Carte de hauteur (Height Map) :** Cette carte représente les variations de hauteur de l'environnement, ce qui est particulièrement utile pour la navigation en intérieur où les niveaux de sol sont importants.

[3] LiDAR : Light Detection And Ranging ou détection et estimation de la distance par la lumière

- **Carte de coût (Cost Map) :** Une carte de coût attribue des coûts aux différentes zones de l'environnement en fonction de la facilité de déplacement. Elle est souvent utilisée pour la planification de trajectoire.
- **Carte de probabilité de localisation (Localization Probability Map)** : Cette carte représente la probabilité de la position du robot à différents endroits de l'environnement. Elle est utilisée pour estimer la position du robot par rapport à la carte.
- **Carte de chaleur (Heatmap)** : Une heatmap attribue des valeurs de chaleur à différentes zones de l'environnement en fonction de la fréquence ou de la probabilité d'occurrence d'événements spécifiques. Elle peut être utilisée pour détecter des zones d'intérêt ou des anomalies.
- **Carte de navigation (Navigation Map)** : Cette carte est spécifiquement conçue pour aider le robot à planifier des trajectoires sûres et efficaces dans l'environnement en indiquant les zones accessibles et les obstacles.
- **Carte de détection d'objet (Object Detection Map)** : Cette carte peut être utilisée pour marquer la position et la classe des objets détectés dans l'environnement, tels que les piétons, les véhicules, etc.

Le choix du type de carte dépendra des besoins spécifiques de la tâche de SLAM, des capteurs utilisés et de la précision souhaitée pour la représentation de l'environnement. Certains projets de SLAM peuvent également utiliser une combinaison de plusieurs types de cartes pour obtenir une représentation plus complète de l'environnement.

3.3 - Les catégories d'algorithmes

Il existe plusieurs catégories d'algorithmes utilisés pour résoudre le problème SLAM, notamment :

- **SLAM basé sur la vision** : Ces algorithmes utilisent principalement des données visuelles pour la localisation et la cartographie. Ils peuvent inclure des techniques de vision par ordinateur, de suivi d'objets et de correspondance d'images.
- **SLAM basé sur LiDAR** : Ces algorithmes se concentrent sur l'utilisation de données LiDAR pour la localisation et la cartographie. Ils peuvent utiliser des méthodes de fusion de nuages de points, de filtrage probabiliste, et d'appariement de scans.
- **SLAM Multimodal** : Certains systèmes SLAM combinent plusieurs types de capteurs, tels que la vision et le LiDAR, pour améliorer la précision et la robustesse de la localisation et de la cartographie.
- **SLAM à base d'apprentissage automatique** : Des techniques d'apprentissage automatique, telles que les réseaux de neurones, sont de plus en plus utilisées pour améliorer la précision et la robustesse des estimations de SLAM.
- **SLAM temps réel vs hors ligne** : Certains algorithmes visent à fournir une localisation et une cartographie en temps réel, tandis que d'autres peuvent fonctionner hors ligne, en traitant des ensembles de données collectés précédemment.

Le choix de l'algorithme dépendra des spécificités du problème, des capteurs disponibles et des contraintes opérationnelles du robot. Chaque approche a ses avantages et ses inconvénients, et il est souvent nécessaire d'effectuer des compromis en fonction des besoins du projet.

4 - Conclusion

En concluant cette exploration du SLAM (*Simultaneous Localization and Mapping*), nous avons parcouru les fondements essentiels de ce domaine fascinant de la robotique. En mettant l'accent

sur la compréhension des problématiques, nous avons découvert que le SLAM repose sur un objectif majeur : permettre à des robots autonomes de se déplacer dans un environnement inconnu tout en créant une carte de cet environnement en temps réel. Cette capacité trouve des applications variées, de la navigation des véhicules autonomes à l'exploration de milieux hostiles par des robots de recherche et de sauvetage.

En chemin, nous avons exploré les principaux concepts, notamment la localisation, la cartographie et l'approche simultanée de ces deux tâches. Nous avons vu que la sélection des capteurs et le choix des algorithmes déterminent la réussite d'une solution de SLAM. Il est important de noter que le SLAM est un domaine en constante évolution, avec des avancées rapides qui ouvrent la voie à des applications toujours plus ambitieuses.

Références :

[1]: image du site Shutterstock in autoplus.fr

[2]: image du site https://robotnik.eu/what-is-fleet-management-in-robotics/

[3]: Crédit image : iRobot

[4]: de l'article : Active Monte Carlo Localization in Outdoor Terrains using Multi-Level Surface Maps

[5]: Référence : Radar-to-Lidar: Heterogeneous Place Recognition via Joint Learning

[6]: https://atelier-canope-95.canoprof.fr/eleve/Automates%20et%20robots/res/robot.dossierHtml/co/elementsRobots.html

[7]: https://www.annabac.com/annales-bac/les-ondes-au-service-de-la-voiture-du-futur

[8]: Probabilistic robotics, Thrun, S.; Burgard, W. & Fox, D. (2005), MIT Press, Cambridge, Mass