École Centrale de Nantes

Projet REDEV

Recherche et Développement : Analyse Visuelle 360° des environnements urbains

Author:
Richard ARNAUD
Robin TANQUEREL

Instructors:
Thomas LEDUC
Myriam SERVIÈRES
Vincent TOURRE

January 19, 2021

Contents

Chapter 1

Introduction

Aujourd'hui, la majorité des piétons se déplacent avec un téléphone à la main ou dans leur poche. Ce téléphone dispose d'un grand nombre de capteurs. Pour nous repérer au quotidien, notre téléphone dispose d'un système de géolocalisation par satellites, de plus en plus précis avec les années. Notre téléphone dispose également d'un podomètre qui fonctionne, dans les téléphones portables, à l'aide d'un accéléromètre. Néanmoins, nous nous intéressons aux déplacements des piétons, c'est à dire à des déplacements dans un environnement urbain. Cet environnement sont souvent synonymes d'un environnement avec de nombreux autres piétons et avec de nombreux toits tout autour. Ces toits et ces autres piétons peuvent perturber toutes les mesures provenant de l'extérieur comme la qualité du signal GNSS ou la connexion au réseau 4G. Le signal GNSS en ville dispose de nombreux défauts : trop peu de signaux GPS captés, durée du calcul nécessaire une fois le signal reçu, contraintes liées à la distance entre le GPS et le récepteur (variation entre l'émission du signal et sa réception, dérive des horloges des récepteurs). En plus de cela, il faut rajouter des signaux dégradés par l'environnement dans lequel se trouve le piéton et toutes les réflexions des signaux GPS que peuvent provoquer les murs et les toits des bâtiments. Toutes ces perturbations rendent la localisation par satellites peu précise voire impossible en ville. Nous allons donc nous limiter aux informations provenant de l'appareil même pour les mesures. C'est à dire que nous n'allons utiliser que le flux vidéo ainsi que les capteurs inertiels pour localiser le téléphone à chaque instant. L'objectif de ce projet est aussi en lien avec le dérèglement climatique comme l'avance [1]. Avec le dérèglement climatique et la forte pollution présente dans les villes, la majorité des autorités publiques essaient de limiter les moyens de déplacement mauvais pour l'environnement et de favoriser les déplacements à pied, en transport en commun et en vélo. Cela veut donc dire plus de personne se déplaçant sur les trottoirs et donc plus de piétons en quête de leur chemin. Par ailleurs, ces piétons disposent en grande partie de téléphones portables qui contiennent différents capteurs pouvant être utilisés pour les localiser et estimer précisément leur position.

Nous avons donc pris comme objectif de mettre en oeuvre cette localisation en mêlant le flux vidéo et les données inertielles reçues par notre appareil. Cela dit, aujourd'hui les téléphones portables et les caméras n'ont pas une puissance de calcul suffisante pour se localiser en temps réel de cette manière. Par conséquent, l'ensemble de la localisation se fera hors-ligne sur un ordinateur après l'acquisition.

Par ailleurs, nous ne travaillerons pas avec un téléphone portable durant l'ensemble de ce projet. Nous utiliserons une caméra GoPro Max 360°. Comme indiqué dans son nom, cette caméra dispose de deux objectifs lui donnant un champ de vision sur 360°. Cette vision complète facilitera grandement le repérage d'objet même si elle ajoute de nouveaux problèmes. Peu de travaux ont été effectués sur ce type de caméra que ce soit sur sa calibration ou sur la localisation de cette caméra dans l'espace. Notre premier objectif sera donc de synthétiser ce qui existe aujourd'hui que ce soit pour des caméras traditionnelles ou des caméras 360°.

Comme nous l'avons dit, la première étape sera de calibrer la caméra 360° correctement. Une fois cela fait, il faudra implémenter la méthode avec laquelle nous allons localiser la caméra. Pour cette méthode, nous avons choisi d'utiliser du SLAM (Simultaneous Localization and Mapping). Implémenter cette méthode se fera en trois étapes. Premièrement, il nous faudra implémenter la localisation avec seulement les capteurs inertiels de la caméra. Il faudra mêler les données provenant de l'accéléromètre et du gyromètre de la GoPro pour obtenir une localisation dans un repère local du déplacement de la caméra. Deuxièmement, nous allons devoir implémenter la partie visuelle de ce SLAM. Cette partie est très importante puisque c'est la seule qui permettra un recalage dans l'espace de notre localisation. L'immense avantage de la caméra 360° est d'avoir deux "passages" sur chaque objet. En effet, lorsque nous avançons nous captons d'abord l'avant de l'objet et lorsque nous le dépassons, l'arrière rentre dans notre champ de vision. Enfin, il faudra mêler les informations visuelles aux informations inertielles. Il faut se rappeler que l'objectif final n'est pas seulement de localiser notre caméra. L'objectif est d'établir une carte 3D très précise de l'environnement urbain ce qui nécessite de stocker les images issues de notre SLAM et également d'avoir une localisation tout aussi précise de notre caméra.

Chapter 2

Glossaire

- Caméra Monoculaire : Une caméra traditionnelle comme nous les connaissons. Elles n'ont qu'un objectif
- Caméra Stéréo : Cette caméra a deux objectifs dont une partie du champ de vision ce recouvre. Dans cet intervalle de recouvrement, on peut déterminer la distance entre ce point et la caméra.
- Caméra RGB-d : Cette caméra n'a qu'un seul objectif mais dispose d'un capteur de profondeur.
- Caméra Omnidirectionnelle : Ce type de caméra a un champ de vision à 360°. En général, les déformations sont très importantes sur les bords des différents objectifs.
- IMU : Une centrale inertielle est un dispositif de mesure de l'accélération et de la vitesse angulaire. Une centrale inertielle dispose de trois accéléromètres et de trois gyromètres permettant ainsi de connaître les trois composantes de l'accélération et de la vitesse de rotation.
- Filtre de Kalman : Un filtre de Kalman est un filtre ayant pour but d'estimer les états d'un système dynamique à partir de mesures brouillées et/ou incomplètes. Un filtre de Kalman dispose d'un vecteur d'état qu'il met à jour à chaque nouvelle information fournie par les capteurs.
- SLAM dense/épars : Un SLAM dense reconstruit l'intégralité de l'environnement. Un SLAM épars donne un nuage de point correspondant à l'environnement autour du dispositif.
- SLAM Direct/Indirect : Un SLAM direct prend directement les données fournies par les capteurs et les utilisent pour les calculs. Le SLAM direct est aussi appelé "feature-less" puisqu'ils n'utilisent aucun détecteur de points d'intérêts ou descripteurs. Un SLAM indirect fait des étapes de pre-processing prenant en compte les bruits de mesure et leurs conséquences avant d'utiliser les données. Par exemple, toutes les méthodes utilisant ORB ou SURF sont indirectes ou "feature-based".
- Fermeture de boucle : Le SLAM ne donne qu'une localisation locale qui dérive peu à peu avec le temps. La fermeture de boucle consiste à recaler notre estimation lorsque l'on "reconnaît" une zone qui a déjà été cartographier précédemment. On peut donc reprendre la position que nous aviosn à cette étape et replacer correctement les points détectés entre-temps.

Chapter 3

État de l'art

3.1 Calibration de caméra traditionnelles

3.2 Calibration de caméra omnidirectionnelles

3.3 Méthodes existantes

Dans cette section, nous allons étudier les différentes méthodes de localisation utilisées ces dernières années. Nous privilégierons les méthodes utilisant au moins une caméra, qu'elle soit monoculaire, stéréo ou omnidirectionnelle, en lien avec le matériel avec lequel nous travaillons. Nous commencerons par revenir aux bases en redéfinissant le SLAM et nous développerons différentes méthodes de SLAM. Il faut se rappeler dans l'exposé de ces différentes méthodes qu'il n'y en a pas forcément de meilleures que d'autres. Chaque méthode que nous allons développer par la suite vise un objectif précis et utilise un matériel précis. Pour comparer les différentes méthodes de localisation il va évidemment falloir comparer la précision obtenue à la fin mais aussi le matériel utilisé, le temps de calcul, la qualité de la carte produite si l'on étudie une méthode qui cartographie l'environnement, la gestion de grande quantité de données, les différences de précision en fonction de l'environnement...

3.3.1 Odométrie Visuelle

L'odométrie visuelle est la base de tout cela. On travaille uniquement avec une caméra et on essaie de repérer cette caméra avec seulement ses images. Le principe est simple. On extrait un certain nombre de points d'intérêts d'une image puis de la suivante. Ensuite, on calcule la transformation entre les deux images et on en déduit le déplacement. Néanmoins, nous n'avons aucun moyen de déterminer le facteur d'échelle dans les positions et les orientations estimées. En effet, il n'y a aucune information métrique lorsque l'on utilise une caméra. Peut-être que les points d'intérêts se seront déplacés de quelques pixels mais comment faire pour transformer cette différence de pixels en différence de mètres. Il est possible d'estimer les déplacements à l'échelle à l'aide de points de repère connus ou à partir d'une translation initiale d'une longue connue. Cela permet ainsi de résoudre les problèmes du facteur d'échelle initial. Toutefois, image après image cette translation initiale va finir par être "oubliée" et le facteur d'échelle finira par diverger avec le temps. Cette divergence dégrade petit à petit les mesures conduisant à une erreur qui s'accumule un peu plus à chaque image. Cette divergence peut se corriger de plusieurs manières. Nous pouvons commencer par la localisation d'un objet 3D de taille connue. Cela fonctionnerait évidemment pour redonner une notion de distance physique à l'algorithme. De même, il est possible d'utiliser des points de repère. Une étude a été réalisé pour calculer le facteur d'échelle à l'aide de la silhouette des immeubles environnants et de leurs modèles dans [19]. Une idée pourrait être de partir d'un accéléromètre pour estimer le facteur d'échelle en intégrant deux fois ces mesures. Toutefois, cette double-intégration rajouterait de l'erreur dans

le calcul et n'est pas envisageable.

3.3.2 SLAM

Le SLAM est bien plus qu'une simple méthode de localisation. Comme l'indique son nom, Simultaneous Localization and Mapping, le SLAM crée une carte de l'environnement tout en permettant la localisation. En seulement l'espace de quelques années, le SLAM est devenu un sujet d'étude crucial dans le domaine de la robotique et de la réalité augmentée. Le SLAM peut se servir de nombreux capteurs différents, caméra, lidar, centrale inertielle et chaque capteur peut être utilisé de manière différente.

3.3.2.1 vSLAM

vSLAM ou Visual SLAM est un SLAM n'utilisant que les données d'une caméra.

Le SLAM visuel le plus basique est le MonoSLAM. Ce premier SLAM utilise un filtre de Kalman étendu pour estimer le déplacement de la caméra et la structure 3D de l'environnement. Le vecteur d'état du filtre est composé des 6 degrés de liberté de la caméra et les positions 3D des points d'intérêts. On ajoute petit à petit des points au vecteur d'état avec le déplacement de la caméra. On commence par initialiser le MonoSLAM avec un objet connu ce qui permet d'obtenir un système de coordonnées globales.

Le PTAM est une version qui fait comme le MonoSLAM mais en parallélisant les tâches. Le tracking et le mapping sont effectués en parallèle. Cela est permis par l'abandon du filtre de Kalman étendu en utilisant à la place une étape d'optimisation appelée Bundle Adjustment. Le PTAM s'initialise avec l'algorithme des 5 points. La pose de la caméra est estimée évidemment à l'aide de la comparaison entre la carte et les nouvelles images. Les positions 3D des points d'intérêts sont ensuites obtenus par triangulation et sont par la suite optimisé à l'aide du Bundle Adjustment. Enfin, on utilise une recherche d'arbre randomisée pour trouver l'image la plus proche de la dernière image à rentrer dans le programme. L'abandon du filtre de Kalman pour le Bundle Adjustment permet de mieux gérer de grands volumes de données. En effet, la précision du SLAM dépend presque entièrement du nombre de point d'intérêt et donc du nombre d'opérations à effectuer par image.

Le DTAM (Dense Tracking and Mapping) se trouve dans cette catégorie. Il est le premier dans cette catégorie à créer une carte dense de l'environnement détecter. C'est également la première méthode directe c'est à dire que l'on utilise directement l'image sans aucune abstraction, sans descripteur et sans détecteur de point d'intérêt. Le tracking est fait en comparant la dernière image de la caméra avec la vue générée précédemment par le mapping.

ORB-SLAM2 est un autre exemple clé de vSLAM [12]. ORB-SLAM est l'évolution d'ORB-SLAM et fonctionne entièrement à l'aide de l'extraction des points clés des images avant d'effacer ces images. La suite des opérations est ensuite basée sur ces points clés. ORB-SLAM 2 est né du besoin d'avoir un algorithme de SLAM Open-Source très puissant qui serait une base pour le travail d'un plus grand nombre. Cette méthode prouve que le Bundle Adjustment est meilleur qu'un filtre de Kalman que ce soit pour la précision ou pour les ressources informatiques demandées.

Le LSD-SLAM est l'un des SLAM les plus utilisés ces dernières années. Comme le précédent, c'est un SLAM direct qui fonctionne avec une odométrie visuelle semi-dense. Semi-dense signifie qu'à la place de reconstruire l'intégralité de la vue, on va se limiter à la reconstruction de zones présentant un fort gradient. Cette méthode se limite à une caméra monoculaire, il

est donc impossible d'estimer la profondeur. Le facteur d'échelle est impossible à déterminer et il divergera avec le temps. Cela est problématique mais cela permet au moins de passer d'environnement très restreint à des environnements très ouverts. Le LSD-SLAM est une méthode directe puisqu'il a été estimé que l'étape d'extraction de points d'intérêts supprime trop d'informations importantes pour fonctionner correctement en ville. Utiliser l'intégralité des informations de l'odométrie visuelle s'avère plus robuste en ville. Toutefois, il est impossible de travailler avec des images complètes sans filtrer et c'est ce que l'on applique ici. On utilise ce que l'on appelle un "semi-dense depth filtering". Le processus du LSD-SLAM se découpe en trois phases : localisation, estimation d'une carte de profondeur et optimisation de la carte. L'étape de localisation est assez traditionnelle et estime la pose de la caméra comparée à la pose de la caméra à l'image précédente et à la carte de profondeur calculée à l'étape précédente. L'estimation de la carte de profondeur utilise alors la nouvelle image pour remplacer ou raffiner l'image-clé actuelle qui est utilisée pour calculer la pose de la caméra. Sur cette image-clé on calcule la profondeur de chaque pixels par rapport aux cartes de profondeur établies à l'image précédente. Si la caméra s'est trop déplacée, on crée une nouvelle image-clé. Enfin, une fois que notre image-clé devient une référence pour la localisation de la caméra, on l'inclut dans la carte globale. Pour détecter une divergence du facteur d'échelle ou une fermeture de boucle, on essaie d'estimer une similitude entre les différentes images-clés et optimiser le tout. Pour l'initialisation, il ne faut rien faire de particulier à part une translation suffisante de la caméra pour que la caméra trouve des images-clés et puisse commencer à estimer la profondeur. Chaque image-clé est donc composée d'une image de la caméra, d'une carte de profondeur et d'un bruit sur chaque pixel de la carte de profondeur. Pour éviter la divergence du facteur d'échelle, une méthode simple est utilisée : la moyenne de la profondeur de chaque image-clé est ramenée à 1.[7] explique plus clairement certains points de cette méthode.

C'est dans cette catégorie que l'on trouve le Dynamic SLAM [24] [11]. Il part du problème que le SLAM fonctionne correctement dans un environnement statique mais dans un environnement dynamique il y a de nombreuses interférences et erreurs dues au déplacement des objets. Pour cela, on rajoute du deep learning au SLAM. L'intérêt du deep learning est qu'il est difficile de reconnaître plusieurs fois le même objet entre les changements d'angles et les déplacements des objets. Le but est alors d'apprendre les objets au fur et à mesure. On trace alors des boîtes qui entourent les objets et détecte leurs points d'intérêts. Le SLAM en dynamique a toujours été insurmontable puisqu'il faut arriver à comprendre qu'un objet est dynamique à partir d'une image plan sachant qu'ils sont particulièrement difficiles à détecter et à traquer. C'est là que rentre en jeu le deep learning. Malheureusement, les dispositifs de SLAM dynamique nécessitent pour l'instant un moyen d'obtenir la profondeur que ce soit un lidar ou une caméra stéréo. Le principe est de faire du ORB-SLAM2 avec lequel on rajoute une détection des objets dynamiques. Cette détection sépare les points d'intérêts statiques et dynamiques et permet de traiter les parties dynamiques comme des cas particuliers. On estime donc la pose à l'aide des seuls points statiques. Pour les points dynamiques, une compensation pour toutes les détections ratées est mise en place pour permettre une meilleure reconnaissance des objets.

Enfin, on trouve le Pop-up SLAM [20]. Le Pop-Up SLAM vise des environnements extrêmement restreints où l'on trouve peu de détails. En effet, les méthodes existantes de SLAM ne sont pas robustes dans les environnements avec peu de points d'intérêts. De ce fait, la cartographie et la localisation sont très imprécis et ne donnent que peu d'informations. L'objectif de ce travail est de créer un modèle 3D "pop-up" provenant d'une seule image. Pour cela, il faut trois étapes. La première est une détection de sol grâce à des réseaux de neurones. Ensuite, il faut tracer des lignes sur les contours du sol détecté. Enfin, le plan représentant le sol ayant été représenté par les lignes tracées à l'étape précedente, on peut extraire le modèle 3D de la pièce puisque l'on sait que le sol est plan. La solution est de mettre en place un graphe de facteur qui comprend les différentes poses détectées, les plans par rapport auxquels on peut se reprérer,

les mesures de ces plans provenant du modèle pop-up 3D et les informations provenant du modèle pop-up 3D et les informations provenant de l'odométrie. Pour la localisation on utilise alors 3 données provenant de ce graphe : la différence entre les normales des différents plans, la distance entre les deux plans et le recoupement des deux plans après projection de l'un sur l'autre. Les résultats ne cette méthode sont très bons dans les environnements qu'elle cible. On obtient bien une carte 3D dense de ces environnements difficiles. Dans des environnements comme des couloirs sans aucun points d'intérêts, on obtient moins de 5% d'erreur. Toutefois, il faut garder à l'esprit que cette méthode ne fonctionne pas du tout dans des environnements plus ouverts.

3.3.2.2 viSLAM

Le viSLAM ou Visual Inertial SLAM est un SLAM combinant les données de capteur inertiels et d'une caméra (et également d'un magnétomètre en général) lors de son fonctionnement

C'est dans cette catégorie que l'on trouve Kimera. Kimera provient du fait qu'il n'y ait aucune bibliothèque Open-Source pour du SLAM visuel-inertiel qui reconstruit la surface des objets. Ses programmeurs ont donc créé Kimera qui fait de l'odométrie visuelle-inertielle, une estimation de la trajectoire dans un repère global, un module léger de reconstruction rapide 3D des surfaces et un module de reconstruction 3D. Kimera fait ce qu'on appelle de la reconstruction "dense" de l'environnement, c'est à dire que l'environnement est reconstruit par bloc. Tous les points d'intérêts correspondant à un même objet sont rassemblés pour tracer les contours de cet objet. Le plus compliqué est donc évidemment qu'il faut simultanément estimer la géométrie 3D d'une scène et réussir à rassembler tous les points d'intérêt repérés sur les objets correspondants. A l'inverse des autres méthodes développées aujourd'hui, on travaille avec une simple caméra monoculaire et une centrale inertielle. Kimera est donc une bibliothèque Open-Source programmée en C++ qui possède plusieurs modules : un module VIO spécialisé pour une estimation précise de l'état de l'IMU, un module RPGO qui propose une élimination plus robuste de certains cas particuliers, un module Mesher qui calcule pour toutes les images les surfaces des objets repérés pour permettre d'éviter les obstacles et un module Semantics plus lent dans son exécution mais qui calcule les surfaces des objets 3D plus précisément en utilisant une approche volumique. Kimera peut calculer la pose à l'aide des données inertielles à plus de 200Hz. La vision est cependant plus lente. Le module le plus lourd de Kimera est Kimera Semantics qui peut afficher les surfaces des objets à une fréquence de 10Hz. Les autres modules de Kimera tournent entre 25Hz et 100Hz.

3.3.2.3 Amélioration cartographie SLAM

[21] propose une amélioration de la représentation et du calcul de large nuage de point lors du mapping. Le SLAM traditionnel donne un nuage de point épars lorsqu'il cartographie le lieu et cela est possible en temps réel. Si l'on veut une cartographie dense ou semi-dense la majorité des méthodes ne fonctionnent qu'en hors-ligne. Cette amélioration consiste à remplacer le stockage de nuages de points par le stockage de segments 3D pour contenir un maximum d'information sur la structure de la scène avec le maximum d'efficacité. Le but est de reconstruire les objets à l'aide d'un ensemble de segments 3D. Il faut donc réussir à extraire avec précision des lignes depuis les résultats données par le SLAM. Ce papier se focalise sur une extraction précise des lignes sans chercher à améliorer la localisation de la caméra. Pour cela, on ne détecte pas des lignes 2D pour ensuite trouver les points 3D et en déduire la ligne 3D qui relie les points. A la place, on calcule la carte de profondeur du nuage de point, puis on calcule un segment 3D reliant deux bords et enfin on regroupe et on filtre les lignes obtenues par rapport aux lignes précédemment extraites. Cette méthode augmente immensément la qualité de la cartographie. L'extraction des lignes permet de relier les points ayant une géométrie marquée ensemble menant à une réelle amélioration pour l'affichage des surfaces.

3.3.3 VIMMU

VIMMU ou Visual Inertial and Magnetic Measurement Unit est une méthode de localisation des piétons à l'aide de la fusion de l'estimation donnée par les différents capteurs inertiels et magnétiques avec des données provenant de la caméra. Dans [16], [18],[5],[17],[15],[14] les données de la caméra sont à la fois une odométrie visuelle mêlée à l'utilisation de GIS, Geographical Information System.

Cette méthode part du constat qu'il est aujourd'hui difficile de localiser les piétons dans un environnement urbain. C'est un environnement dynamique où il y a de nombreuses réflexions spéculaires.Il est extrêmement difficile de se localiser dans cet environnement sur de longues distances. Toutefois cet environnement difficile dispose d'un avantage qu'aucun autre n'a : c'est un environnement qui est visité tous les jours et qui est extrêmement bien connu. C'est dans ces zones où les GIS sont les plus remplis. Les GIS ou Geographical Information System sont un ensemble d'information sur un terrain, sur un bâtiment ou simplement sur un objet. Il peut contenir ses coordonnées satellites, des photographies de divers côtés de l'objet. Les GIS peuvent venir de diverses sources qu'elles soient officielles, souvent des villes elles-mêmes, ou collaboratives comme OpenStreetMap.

La solution proposée et développée par [16], [18],[5],[17],[15],[14] consiste à fusionner les vitesses angulaires, les accélérations, le champ magnétique et la vidéo enregistrée par l'outil. Les trois premiers permettent de calculer l'orientation de l'outil. Pour décomposer plus précisément le processus de localisation, on peut discerner deux étapes. La première étape est la partie IMMU qui utilise la centrale inertielle et le magnétomètre pour essayer de détecter les pas, calculer la longueur de ces derniers et enfin en déduire la position. La deuxième étape est l'étape de vision qui avec l'aide du champ magnétique, des GIS et de la reconnaissance d'objets essaie d'estimer l'orientation et l'attitude. Tout cela est contenu dans un vecteur d'état qui est utilisé dans un filtre de Kalman. Ce vecteur contient la position dans le repère de navigation, le quaternion de la rotation entre le corps et le repère de navigation, le quaternion d'erreur du gyromètre, le quaternion d'erreur de l'accéléromètre, la fréquence des pas, la longueur des pas, un vecteur contenant les coordonnées 2D des objets 3D détectés et enfin la rotation et la translation entre le repère de l'objet et le repère de la caméra.

Le matériel utilisé ici est assez particulier. On dispose d'une caméra équivalente à celle d'un téléphone portable auquel est greffé une centrale inertielle et un magnétomètre. Ce dispositif est construit pour être porté dans la main d'un piéton se déplaçant dans la rue. Le grand problème de cette méthode est que la main d'un piéton se déplace librement par rapport à son centre de masse. Cela fait que le déplacement de la main, et donc de la centrale inertielle, contient à la fois les déplacements indépendant de la main et chocs dus au déplacement. On pourrait se dire qu'utiliser un podomètre et une longueur de pas fixe pourrait résoudre le problème facilement. Cela fonctionnerait bien lors d'une randonnée mais nous étudions des déplacements urbains. Ce modèle de pas fixe ne pourrait pas représenter la réalité du fait qu'un piéton doit éviter d'autres piétions, qu'un piéton doit s'arrêter aux feux de circulation, qu'il faut traverser les passages protégés... Chaque mouvement a sa propre vitesse et cette vitesse est unique à chaque piéton.

La partie IMMU se décompose elle-même en plusieurs phases. La première est évidemment une phase de calibration et de choix sur les différents capteurs. Pour calibrer le gyromètre et l'accéléromètre, il faut choisir un modèle pour en déduire le bruit de mesure qui sera plus tard utile dans le filtre de Kalman. Un modèle simple a été choisi pour ces deux capteurs. On considère que le résultat est égal à la mesure à laquelle on ajoute le biais de mesure et un bruit blanc. Ce résultat est envoyé dans un algorithme appelé MAGYQ (Magnetic Acceleration fields

and GYroscope Quaternion) qui renvoie en continu les quaternions d'erreur du gyromètre et de l'accéléromètre dans le vecteur d'état. Pour calibrer le magnétomètre c'est encore plus simple. Le champ résultat est égal au champ mesuré auquel on ajoute un bruit blanc. Toutefois, il faut quand même calibrer le magnétomètre avec chaque expérimentation. Après cette phase de calibration, il est temps de passer à la partie expérimentation. Une première idée pour cette partie IMMU serait d'intégrer deux fois les données de l'accéléromètre. Le problème est comme nous l'avons vu précédemment que pas après pas l'erreur de mesure va s'accumuler si l'on n'a aucun moyen de la remettre à zéro. Pour calibrer de nouveau, il faut effectuer ce que l'on appelle une ZUPT (Zero Velocity Updates) mais cela n'est pas possible de rester à l'arrêt dans le contexte du déplacement d'un piéton. Par ailleurs, l'objet est tenu dans la main du piéton et non à son centre de masse et en intégrant deux fois nous n'arriverons pas à différencier les mouvements du corps entier du mouvement de la main seule. La méthode privilégiée est différente. Pour rappel l'objectif de la partie IMMU est de détecter les pas, leur fréquence et leur longueur. Pour cela, on fait une détection de pic sur le gyromètre et l'accéléromètre pour trouver l'instant où le pied touche le sol. Il y a toutefois des paramètres qui ne peuvent pas être déterminés par le programme en lui-même. En entrée, avant l'expérimentation, il faut donner certaines informations sur le porteur de l'outil comme sa taille ou son sexe. Une fois que l'on sait détecter les pas et leur longueur il suffit de prendre la localisation au pas d'avant et d'ajouter le nouveau pas. Cette étape s'appelle le PDR (Pedestrian Dead Reckoning). Il faut toutefois connaître l'orientation du dernier pas comparé au pas précédent ce qui provient du magnétomètre et de l'algorithme MAGYQ. Il y a 6 phases dans le processus de PDR : mesures des capteurs, détection de la phase de déplacement, estimation de la fréquence de pas, méthode de transport, détection des pas et estimation de la longueur de ces pas. Une difficulté supplémentaire est à considérer en ville pour cette partie. Le champ magnétique peut être perturbé à proximité des bâtiments à cause des matériaux ferromagnétiques utilisés.

La partie vision et magnétomètre se décompose elle aussi en plusieurs phases. On peut mentionner tout d'abord que la caméra est ramenée à 10Hz pour diminuer les temps de calcul. La première phase est toujours une phase de calibration. On utilise un échiquier pour déterminer les paramètres intrinsèques et ainsi dédistordre les images. Ensuite vient la phase d'utilisation de la caméra. On dispose donc d'une base de données d'objets parfaitement connus ainsi que les coordonnées de ces objets. Pour les reconnaître, on extrait les points d'intérêts des objets que l'on trouve avec SURF. Ensuite, on compare ces points extraits par SURF avec les points d'intérêts des objets pré-enregistrés. Ensuite, les points mis en correspondance sont filtrés avec MSAC pour supprimer les intrus. Enfin, on estime la transformation entre les points de référence et les points d'intérêts extraits de la vision de la caméra et cela nous permet de retrouver les coordonnées de notre objet 3D dans le repère de l'image. Maintenant, il nous faut retrouver les coordonnées de la caméra comparée à l'objet. Pour cela, on estime la rotation et la translation entre le repère de l'objet et le repère de la caméra. Ensuite, il suffit d'utiliser un algorithme EPnP couplé avec une optimisation de Gauss-Newton pour estimer les paramètres extrinsèques de la caméra. On peut rassembler le processus de la caméra en six étapes : acquisition des images, extraction SURF des points d'intérêts, écart des moindres carrés pour reconnaître les points d'intérêts, suppression des intrus, triangulation et enfin estimation de la pose. Cette partie vision amène ses propres difficultés. La précision de la localisation à l'aide de GIS et de recherche d'image dans une base de données décroît très fortement avec les changements de luminosité, d'apparence, de saison ou de condition climatique. Par ailleurs, en ville les travaux sont communs et les piétons sont nombreux. Ces deux éléments peuvent obstruer le champ de vision de la caméra. Enfin, un élément crucial de l'utilisation de GIS est qu'il est très long de mettre en place ce genre de base de données.

Maintenant il faut obtenir la hauteur à laquelle se trouve le dispositif. Cette hauteur se trouve à l'aide des GIS. En plus de cela, une excellente estimation de la hauteur de la main qui

tient l'outil est de considérer qu'un piéton tient son téléphone à environ 90% de sa taille. Pour conclure, on utilise un filtre de Kalman étendu pour coupler les résultats. Avec un objet pré-enregistré en vue, l'erreur est extrêmement faible et de l'ordre d'une trentaine de centimètres. Néanmoins, sans objet en vue depuis plus d'une centaine de mètre, l'erreur est de l'ordre de plusieurs dizaines de mètres. On peut également compléter tout ce processus avec de l'odométrie visuelle. Cette odométrie peut ainsi utiliser tout le processus de PDR qui donne des mesures en mètre. Ces mesures permettent donc de déterminer le facteur d'échelle. Premièrement, il faut interpoler les données pour les remettre sur la même base de temps entre la caméra et la centrale inertielle. Ensuite, le facteur d'échelle est facilement calculable. En négligeant les déplacements verticaux on a Svo(t) = ||C(t)-C(t-1)|| où C(t) est le centre optique de la caméra à l'instant t. Nous pouvons faire de même pour l'IMMU : Sstep(t) = STEPk(DTimage/DTstepk) où DTimage est la durée entre deux images, DTstepk la durée entre deux pas et STEPk la longueur estimée du pas entre les instants k-1 et k. Il suffit alors de faire le rapport des deux pour obtenir le facteur d'échelle : s(t) = Sstep(t)/Svo(t). Avec cette méthode supplémentaire, la précision augmente encore un peu plus. Cette méthode montre également que les pas en ville ne font pas du tout la même longueur et ont presque 40% d'écart faisant entre un mètre et un mètre quarante.

Chapter 4

Lecture Article par Article

4.1 Pedestrian Track Estimation with Handheld Monocular Camera and Inertial-Magnetic Sensor for Urban Augmented Reality

[16]

- Difficile de localiser des piétons en environnement urbain. Téléphones avec nombreux capteurs peuvent permettre une estimation de la pose de l'outil. Aujourd'hui, il existe des GIS (Geographical Information System) qui constituent une base de données gigantesque pour aider à la localisation. Pedestrian Dead-Reckoning (PDR) à définir.
- Méthode utilisée : fusionner l'estimation donnée par les capteurs inertiels et magnétiques avec la connaissance de modèle 3D et de leur coordonnées.
- Objectifs : Premièrement rendre plus précise la localisation des piétons en ville. Deuxièmement, permettre d'utiliser la RA en ville, sur les devantures des bâtiments par exemple.
- Problémes : Le GNSS (Global Navigation Satellites Systems) ne fonctionne pas ou fonctionne mal dans les canyons urbains et à l'intérieur. Les capteurs utilisés sont souvent de faibles qualités.
- Solution : Fusionner les vitesses angulaires, accélérations, champ magnétique et vidéo enregistrés par l'outil. Les trois premiers permettent de calculer l'orientation de l'outil. La caméra avec l'aide de 3D GIS permettent d'estimer la pose lorsque l'on arrive à détecter un des objets enregistré préalablement. Les GIS sont choisis à partir de source fiable comme les villes ou tirés de OpenStreetMap.
- Travail similaire : Pour la localisation des piétons, la méthode traditionnelle est par caméra et par reconnaissance d'objet (Object Recognition ou Content Based Image Retrieval). En général, on utilise les données fournies par les différents capteurs pour limiter le temps de recherche des différents objets dans la base de données. Traditionnellement, un filtre de Kalman est utilisé pour estimer la position du piéton, limiter la divergence de la centrale inertielle, calculer les erreurs et limiter les inconsistences. Malheureusement, sans autre capteur que l'IMU, cette méthode donne une précision à plusieurs mètres près.
- Méthode VIMMU (Visual Inertial and Magnetic Measurement Unit) finalement utilisée. Basiquement deux parties. Une partie IMMU qui essaie de détecter les pas, calcule la longueur des pas et en déduit la position. Une partie Vision avec l'aide du champ magnétique, des GIS et de la reconnaissance d'objet pour trouver l'orientation et l'attitude avec MAGYQ qui corrige la position donnée par l'IMMU. Le vecteur d'état est x = [Position dans le repère de navigation, quaternion de la rotation entre le corps et le repère de navigation, quaternion d'erreur du gyro, quaternion d'erreur de l'accéléro, fréquence des pas, taille des pas, un vecteur contenant les coordonnées 2D des objets 3D détectés et enfin la rotation et la translation entre le repère de l'objet et le repère de la caméra]

- Partie IMMU
  - Pour calibrer le gyroscope et l'accéléromètre il faut choisir un modèle pour trouver le bruit de mesure. Un modèle simple a été choisi. En gros on fait y = ymesuré + biais de mesure + bruit blanc pour le gyro et pour l'accéléro. Le tout est envoyé dans MAGYQ (Magnetic Acceleration fields and GYroscope Quaternion) qui renvoie en continue les quaternions d'erreur du gyro et de l'accéléro dans le vecteur d'état
  - Pour calibrer le magnétomètre : On calibre toujours avant de débuter l'expérimentation. On considère cette fois-ci que le champ = champ mesuré + bruit blanc.
  - Détection des pas, fréquence des pas et longueur des pas : Détection de pic sur le gyro et l'accéléro pour trouver l'instant où le pied touche le sol. Il y a toutefois des paramètres qui dépendent des proportions du porteur de l'outil, taille, sexe...
  - Localisation : On prend la mesure du pas d'avant et on rajoute un pas avec l'aide de MAGYQ qui donne la direction du pas. Pour l'instant initial on prend une localisation par GNSS.
- Partie Vision + Magnétique, caméra haute définition ramenée à 10Hz pour réduire les temps de calcul
  - Calibration : On dédistort les images et on détermine les paramètres intrinsèques après cela à l'aide d'un échiquier
  - Reconnaissance : On connait parfaitement certains objets et leur localisation. Pour les reconnaître, on extrait avec SURF les points d'intérêts des objets que l'on trouve et on les compare aux points extraits par surfs des objets pré-enregistrés. On filtre avec MSAC pour supprimer les intrus. Ensuite, on estime la transformation entre les points de référence et les points d'intérêts extraits de la vision de la caméra et cela nous permet de retrouver les coordonnées de notre objet 3D dans le repère de l'image.
  - EPnP : Maintenant il faut retrouver les coordonnées de la caméra comparée à l'objet. Pour cela, on estime la rotation et la translation entre le repère de l'objet et le repère de la caméra. Ensuite, on utilise un algorithme EPnP couplé avec une optimisation de Gauss-Newton pour estimer les paramètres extrinsèques de la caméra.
- Ensuite, un filtre de Kalman couple les résultats. Avec un objet pré-enregistré en vue, l'erreur est faible de l'ordre d'une trentaine de centimètres. Néanmoins, sans objet en vue depuis plus d'une centaine de mètre, l'erreur est de l'ordre de plusieurs dizaines de mètres

4.2 Fusion of 3D GIS Vision.... Bref c'est exactement le même que celui d'avant à l'exception des résultats un peu plus détaillés

[18]

4.3 Solving Monocular Visual Odometry Scale Factor with Adaptive Step Length Estimates for Pedestrians Using Handheld Devices

[5]

- Motivation : Difficile de se localiser dans un environnement urbain précisément sur de longs trajets qui plus est avec des capteurs bas-coûts.
- Solution : On effectue une estimation continue de la pose avec une odométrie visuelle. Pour résoudre les problèmes de facteur d'échelle, on utilise un modèle de taille de pas adaptative pour les déplacements dans le plan horizontal. Pour les déplacements dans

le plan vertical, un modèle spécial a été mis en place avec en plus la localisation de la caméra à l'aide de GIS. Ces GIS permettent ponctuellement de recaler l'odométrie visuelle et revenir à une erreur nulle.

- Difficultés : Localisation difficile dans un environnement urbain à cause des canyons urbains. Par ailleurs, la main des piétons se déplace librement par rapport au centre de masse du piéton ce qui le rend encore plus difficile à localiser. Enfin, l'utilisation de capteur bas-coûts ne facilite pas le travail. Le SLAM fonctionne bien mais un piéton ne passe pas deux fois par le même endroit lors d'un déplacement et donc utiliser du SLAM n'est pas naturel pour ce genre d'application. Utiliser un podomètre ainsi qu'un modèle de pas constant est intéressant mais cela ne reflète pas la réalité. Un piéton doit éviter d'autre piéton, s'arrêter au feu, traverser des passages protégés... Chaque mouvement à sa vitesse et cette vitesse est unique à chaque piéton.

- Méthode : Choix d'une technologie entièrement portative et autonome avec une mémoire faible et aucune connexion extérieure.

- Travail similaire

  - GIS et recherche d'image dans une base de données : La précision de cette méthode décroit fortement avec les changements de luminosité, d'apparence, de saison, de condition climatique et même de travaux. Cela demande également énormément de travail
  - PDR : On calcule incrématelement la position du piéton par rapport à la position initiale. Pour que cela fonctionne, il faut fusionner les capteurs pour compenser les faiblesses de chacun.
  - Odométrie Visuelle : On incrémente la position du piéton à partir du déplacement entre chaque image de la caméra. Pour cela, on extrait un certain nombre de point d'intérêt d'une image puis de la suivante, on calcule la transformation entre les deux images et on en déduit le déplacement. Néanmoins, on ne peut pas déterminer le facteur d'échelle dans les positions et orientations estimées. Il est possible d'estimer les déplacements à l'échelle à l'aide de points de repère connus ou à partir d'une translation initiale d'une longueur connue. Cela permet de résoudre les problèmes de facteur d'échelle initiaux mais pas la divergence de celui-ci avec le temps. Cette divergence dégrade petit à petit les mesures conduisant à une erreur qui s'accumule un peu plus à chaque image. Cette divergence peut se corriger de plusieurs manières : localisation par reconnaissance d'objet 3D de taille connue, utilisation de points de repère ou plus particulièrement grâce à la silhouette des immeubles environnants et leurs modèles [19]. On pourrait également partir de l'accéléromètre pour estimer le facteur d'échelle mais cela demanderait d'intégrer d'effectuer une double-intégration rajoutant ainsi de l'erreur dans le calcul.

- Odométrie visuelle monoculaire avec calcul dynamique du facteur d'échelle : Avec une analyse des données inertielles du mouvement humain, c'est à dire une estimation de la longueur de pas, on essaie dynamiquement de calculer le facteur d'échelle.

  - PDR : Pour calculer la position 2D à l'aide d'une IMU, traditionnellement il suffit d'intégrer deux fois le résultat. Le problème est que pas après pas l'erreur de mesure s'accumule si l'on n'a aucun moyen de la remettre à zéro. Pour recalibrer, il faut un ZUPT (Zero Velocity Updates) mais ce n'est pas possible dans le contexte que nous nous sommes donnés. Par ailleurs, l'objet est tenu dans la main du piéton et non à son centre de masse, il faut donc réussir à différencier les mouvements du corps tout entier et les mouvements de la main seule. La méthode employée comprend 6 phases : mesures des capteurs, détection de la phase de déplacement, estimation de la fréquence

de pas, méthode de transport (en gros est ce qu'on écrit sur le téléphone, est ce qu'on le balance avec le bras le long du corps...), détection des pas et enfin estimation de la longueur de ces pas.

  - VO : Indépendamment de l'étape précédente, la caméra nécessite un certain nombre d'étape. Il faut commencer par la calibrer et la modéliser (échiquier et modèle pinhole) pour déterminer les paramètres intrinsèques et supprimer les distorsions. A chaque image, on garde en mémoire les deux précédentes pour comparaison. On a également six étapes : acquisition des images, extraction SURF des points d'intérêts, écart des moindres carrés pour reconnaître les points d'intérêts, suppression des intrus, triangulation et enfin estimation de la pose.
  - DTM (Digital Terrain Model) : bref GIS. En plus de ça on estime la hauteur de la main qui tient l'outil en fonction de la taille du piéton à environ 0.9*hauteur du piéton.

- Détermination du facteur d'échelle : premièrement on interpole les données pour les remettre sur la même base de temps. Ensuite, le facteur d'échelle est facilement calculable. En négligeant les déplacements verticaux on a : Svo(t)=norme(C(t)-C(t-1)) où C(t) est le centre optique de la caméra à l'instant t. Pour Sstep(t) = STEPk x (DTimage/DTstepk) où Sstep(t) est l'estimation de la longueur de pas STEPk entre la dernière image et le dernier pas. D'où s(t) = Sstep(t)/Svo(t)

- Résultats : Une longueur de pas dynamique semble nécessaire à la vue des résultats pour estimer le facteur d'échelle. Les pas font entre 1 mètre et 1 mètre 40 pour l'immense majorité d'entre eux. Avec ce modèle, l'erreur de localisation est en dessous de dix pourcents de la longueur parcourue. Avec une longueur de pas fixe, l'erreur peut atteindre les trente pourcents.

4.4 Continuous Pose Estimation for Urban Pedestrian Mobility Applications on Smart-handheld Devices

[17]

- Objectifs : comme d'hab

- Difficultés : Dans les espaces fermés ou près des bâtiments, le champ magnétique peut être perturbé. Caméra : réflexion spéculaire, changement d'illumination, changement urbain, occlusions, élément à longue distance. Même à l'arrêt la main du piéton bouge pouvant faire croire à un mouvement du piéton –> on préfère un modèle de pas fixe et avec une direction donnée par un magnétomètre plutôt que d'intégrer deux fois l'accéléro.

4.5 An Inertial, Magnetic and Vision Based Trusted Pose Estimation for AR and 3D Data Qualification on Long Urban Pedestrian Displacements

[15]

Useless

4.6 Hybrid Visual and Inertial Position and Orientation Estimation based on Known Urban 3D Models

[14]

- Un gros paragraphe sur l'ensemble des calculs pour EPnP, points d'intérêts et les paramètres de distorsion, extrinsèque et intrinsèque. A part ça c'est la même chose que tous les autres articles

4.7 SPLAT: Spherical Localization and Tracking in Large Spaces

[10]

- Objectifs : Faire de l'affichage de RA dans des stades, en intérieur ou en ville.
- Difficultés : Pour utiliser du SLAM, il faut suffisamment de déplacement dans la position de la caméra. La taille de ce déplacement est proportionnel à la taille de la scène. Par conséquent, dans des environnements urbains où certaines avenues peuvent faire plusieurs kilomètres. La clé est une localisation très précise et un calcul de l'orientation très robuste. Ici, le cas qui nous intéresse est relativement statique, on est censé être à l'intérieur ou dans un stade c'est à dire qu'on ne fait ni aucun déplacement ni seulement de la rotation.
- Solution : Une alternative au SLAM qui combine SLAM, une méthode de suivi 3D robuste et un "Spherical Structure from Motion". On se place dans le cas où la majorité des mouvements de l'utilisateur sont des mouvements de rotation.
- Méthode : Le téléphone est en général tenu à bout de bras ce qui fait qu'il décrit un mouvement sur une surface sphérique. Toutefois, on ne va ni considérer le mouvement comme une rotation pure autour du centre de la caméra ni un mouvement non contraint mais on va faire du SPLAT, SLAM + SfM + suivi D avec une estimation de pose 3D absolue. Pour cela, on utilise une extraction de points d'intérêts ORB.
- Travail déjà effectué dans le domaine :
- SLAM : Nombreux différents SLAM : une caméra, plusieurs caméras, autres capteurs. Récemment beaucoup de travail à une caméra. MonoSLAM, PTAM, ORBSLAM2, LSD-SLAM, SLAM++. Malgré les différences, toutes ces méthodes demandent des informations de profondeur correctes au moins pour l'initialisation. Quelle que soit le SLAM, le fonctionnement dépend grandement du déplacement de l'utilisateur et sur la taille de l'environnement.
- Article très intéressant mais on part du principe qu'il n'y aura pas/peu de déplacement de l'utilisateur ce qui ne concerne pas le déroulé du projet.

4.8 Dynamic SLAM : Semantic Monocular Visual Localization and Mapping based on deep learning in dynamic environment

[24]

- Difficultés : Le SLAM fonctionne dans un environnement statique mais dans un environnement dynamique il y a de nombreuses interférences et erreurs dues au déplacement des objets. Pour cela, on rajoute du deep learning au SLAM.
- Travail similaire : Visual SLAM, monoculaire et temps réel. Plus récemment LSD SLAM qui fait les choses mieux. Réseau de neurones : De plus en plus précis et performants. Avec la reconnaissance d'image c'est difficile parce qu'il faut reconnaître un objet depuis différents angles. R-CNN, SPP-Net, Fast R-CNN, Faster R-CNN apprennent tout seul de nouveaux objets, Single Shot MultiBox Detector trace les boîtes qui entourent les objets et détecte les points d'intérêts dans le SLAM dynamique. SLAMIDE : SLAM in Dynamic Environments deux catégories : détection des objets dynamiques et tracking ou détection

des objets dynamiques et filtrage. SLAMIDE a toujours été insurmontable puisqu'il faut arriver à comprendre qu'un objet est dynamique à partir d'une image plan et également parce qu'ils sont difficiles à détecter et à traquer. Comme c'est impossible, le SLAMIDE a été combiné avec du deep learning. Néanmoins, pour estimer la position des objets dynamiques avec plus de précision la quasi-intégralité des travaux sur le SLAMIDE ont été réalisé avec des caméras possédant un lidar ou un capteur de profondeur. Il y a très peu de recherche sur des caméras monoculaires traditionnelles.

- Solution : Dans cet article on fait du ORB-SLAM2 pour le Dynamic SLAM. Sur cet ORB-SLAM2 on rajoute une détection des objets dynamiques. Cette détection sépare les points d'intérêts statiques et dynamiques et permet de traiter les parties dynamiques comme des cas particuliers. L'estimation de la pose est faite à partir des points statiques. Pour les points dynamiques, une compensation pour toutes les détections ratées est mise en place pour permettre une meilleure reconnaissance des objets.
- Conclusion : Grâce à cette méthode on augmente largement la précision de la localisation, le taux de rappel d'un objet détecté

4.9 Kimera : an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping

[2]

- Difficultés : Il n'y a aucune bibliothèque Open-Source pour du SLAM visuel-inertiel qui reconstruit la surface des objets. Ils ont donc créé Kimera qui fait de l'odométrie visuelle-inertielle, une estimation de la trajectoire dans un repère global, un module léger de reconstruction rapide 3D des surfaces et un module de reconstruction 3D.
- Solution : Le plus compliqué ici est qu'il faut simultanément estimer la géométrie 3D d'une scène et réussir à rassembler tous les points d'intérêt repérés sur les objets correspondants. A l'inverse des autres méthodes développées aujourd'hui, on travaille avec une simple caméra monoculaire et une centrale inertielle. Kimera est donc une bibliothèque Open-Source programmée en C++ qui possède plusieurs modules : un module VIO spécialisé pour une estimation précise de l'état de l'IMU, un module RPGO qui propose une élimination plus robuste de certains cas particuliers, un module Mesher qui calcule pour toutes les images les surfaces des objets repérés pour permettre d'éviter les obstacles et un module Semantics plus lent dans son exécution mais qui calcule les surfaces des objets 3D plus précisément en utilisant une approche volumique.
- Conclusion : Kimera peut calculer la pose à l'aide des données inertielles à plus de 200Hz. La vision est cependant plus lente. Le module le plus lourd de Kimera est Kimera Semantics qui peut afficher les surfaces des objets à une fréquence de 10Hz. Les autres modules de Kimera tournent entre 25Hz et 100Hz.

4.10 Visual SLAM Algorithms : a survey from 2010 to 2016

[6]

- MonoSLAM : On utilise un filtre de Kalman étendu pour estimer le déplacement de la caméra et la structure 3D de l'environnement. Le vecteur d'état du filtre est composé des 6 degrés de liberté de la caméra et les positions 3D des points d'intérêts. On ajoute petit à petit des points au vecteur d'état avec le déplacement de la caméra. On commence par initialiser le MonoSLAM avec un objet connu ce qui permet d'obtenir un système de coordonnées global

- PTAM : A l'inverse du MonoSLAM, PTAM exécute le tracking et le mapping en parallèle. On l'initialise à l'aide de l'algorithme des 5 points. La pose de la caméra est estimée à l'aide de la comparaison entre la carte et les nouvelles images. Les positions 3D des points d'intérêts sont ensuite obtenus par triangulation et sont par la suite optimisés avec un Bundle Adjustment. Enfin, il utilise une recherche d'arbre randomisée pour trouver l'image la plus proche de la dernière image à rentrer dans le programme.
- MonoSLAM vs PTAM : La précision du SLAM dépend presque entièrement du nombre de point d'intérêt et du nombre d'opérations à effectuer par image. Pour cela, le PTAM fonctionne mieux puisqu'il gère mieux de grands volumes de données.
- DTAM (Dense Tracking and Mapping) : Cette méthode est directe et fonctionne avec une caméra stéréo. Le tracking est fait en comparant la dernière image de la caméra avec la vue générée précédemment par le mapping. On extrait ensuite l'information de profondeur en comparant les vues des deux caméras
- LSD-SLAM : Cette méthode est directe et fonctionne avec une Odométrie Visuelle semi-dense. Avec ce SLAM, on reconstruit seulement les objets avec un fort gradient plutôt que tous les objets. Comme on n'a qu'une seule caméra, pour l'initialisation on choisit des valeurs de profondeur de chaque pixel au hasard qui seront optimisées par la suite. La particularité du LSD-SLAM est sa détection de fermeture de boucle et son graphe de pose à 7 degrés de liberté pour avoir une carte géométriquement consistante.
- RGB-D vSLAM : Aujourd'hui les caméras RGB-D (c'est à dire les caméras traditionnelles auquel on ajoute un capteur de profondeur) de viennent plus accessibles. A l'aide de ce capteur de profondeur on peut donc obtenir la structure de l'environnement 3D directement.

4.11 Image-Based Camera Localization : an overview

[23]

- Une synthèse sur les objectifs, les méthodes et les termes de la localisation des caméras.

4.12 Direct Sparse Odometry

[9]

- Direct ou Indirect : Les méthodes directes prennent directement les données fournies par les capteurs et les utilisent pour les calculs. Les méthodes indirectes font une étapes de pré-processing prenant en compte les bruits de mesure et leurs conséquences.
- Dense ou Épars : Les méthodes éparses reconstruisent seulement un ensemble de points indépendents sélectionnés alors que la méthode dense reconstruit tous les pixels de l'image.
- Cet article propose un SLAM épars et non un SLAM dense comme nous cherchons à le réaliser dans ce projet. Nous avons donc décidé de ne pas en prendre plus de notes.

4.13 Incremental 3D Line Segment Extraction from Semi-dense SLAM

[21]

- Problème : Le SLAM semi-dense est devenu assez connu dans ces dernières années. Cependant, toutes les recherches l'utilisant manquent d'efficacité pour représenter et calculer de large nuages de points. Le SLAM traditionnel donne un nuage de point épars lorsqu'il

cartographie le lieu et permet de le faire en temps réel. Si l'on veut une cartographie dense ou semi-dense la majorité des méthodes ne fonctionnent qu'en hors-ligne.

- Solution : Pour stocker un maximum d'information sur la structure de la scène, on remplace le stockage de nuages de points par le stockage de segments 3D. Le but est de reconstruire les objets par un ensemble de segment 3D. Il faut donc réussir à extraire avec précision des lignes des résultats donnés par le SLAM. Dans ce papier, les auteurs se focalisent donc sur l'extraction de ses lignes de la manière la plus précise possible sans chercher à améliorer la localisation de la caméra. Pour cela, on ne détecte pas des lignes 2D et trouver ensuite les points 3D correspondants pour enfin trouver la ligne 3D qui relie les points. A la place, on calcule la carte de profondeur du nuage de point, puis on calcule un segment 3D reliant deux bords et enfin on regroupe et on filtre les lignes obtenues par rapport aux lignes précédemment extraites.
- Résultat : Le résultat est excellent comparé à un SLAM traditionnel. L'extraction des lignes permet de relier les points ayant une géométrie marquée ensemble menant à une réelle amélioration pour l'affichage des surfaces.

4.14 Accurate and Robust Monocular SLAM with Omnidirectional Cameras

[22]

- Problème : Pour les méthodes de SLAM traditionnelles, le modèle pinhole même en considérant un modèle de distorsion donne de mauvais résultats quand le champ de vision de la caméra est supérieur à 120°. Il faut donc pour utiliser une caméra omnidirectionnelle considérer un nouveau modèle de caméra. Ici le choix s'est porté sur le modèle EUCM (Enhanced Unified Camera Model) pour projeter les points 3D sur le plan image. La méthode de SLAM utilisée avec ce modèle est le ORB-SLAM pour sa robustesse contre les changements de points de vue. Le Bundle Adjustment optimise le calcul de la pose de la caméra en minimisant l'erreur de reprojection des points 3D. De nombreuses méthodes sont utilisées pour améliorer la vitesse d'exécution du Bundle Adjustment, notamment à l'aide de matrice jacobienne.

4.15 LSD-SLAM : Large-Scale Direct Monocular SLAM

[8]

- Problème : Le SLAM monoculaire ne permet pas de déterminer le facteur d'échelle ce qui conduit a une divergence avec le temps. Cela est problématique mais permet au moins de passer d'environnement très restreint à des environnements très ouverts. Pour cela, on utilise un SLAM monoculaire et non une caméra avec un capteur de profondeur ou utilisant une vision stéréo. Par ailleurs, les capteurs de profondeur ne fonctionnent pas dans des environnements très ouverts comme l'on cherche à étudier ici.
- Travaux similaires : On pourrait faire de l'extraction d'objets connus, c'est à dire que premièrement on extrait des points clés de l'image et on essaie de calculer la position de la caméra par rapport à ces seuls points extraits. Le problème est que cette étape d'extraction supprime trop d'informations intéressantes pour fonctionner correctement en ville. Deuxièmement, on pourrait faire de l'odométrie visuelle directe qui utilise l'intégralité des informations des images et s'avère plus robuste et plus précis en ville. Toutefois, peu de recherches ont été effectué pour une odométrie visuelle directe monoculaire. Les seules recherches effectuées sur des caméras monoculaires dans ce domaine donnent des temps de

calcul beaucoup trop longs pour être fait en temps réel à moins d'appliquer un "semi-dense depth filtering".

- Solution : Le processus du LSD-SLAM se découpe en trois phases : localisation, estimation d'une carte de profondeur et optimisation de la carte. L'étape de localisation est assez traditionnelle et estime la pose de la caméra comparée à la pose de la caméra à l'image précédente et à la carte de profondeur calculée à l'étape précédente. L'estimation de la carte de profondeur utilise les nouvelles images pour remplacer ou raffiner l'image-clé actuelle qui est utilisée pour calculer la pose de la caméra. Sur cette image-clé on calcule la profondeur de chaque pixels par rapport aux cartes de profondeur établies à l'image précédente. Si la caméra s'est trop déplacée, on crée une nouvelle image-clé. Enfin, une fois que notre image-clé devient une référence pour la localisation de la caméra, on l'inclut dans la carte globale. Pour détecter une divergence du facteur d'échelle ou une fermeture de boucle, on essaie d'estimer une similitude entre les différentes images-clés et optimiser le tout. Pour l'initialisation, il ne faut rien faire de particulier à part une translation suffisante de la caméra pour que la caméra trouve des images-clés et puisse commencer à estimer la profondeur. Chaque image-clé est donc composée d'une image de la caméra, d'une carte de profondeur et d'un bruit sur chaque pixel de la carte de profondeur. Pour éviter la divergence du facteur d'échelle, une méthode simple est utilisée : la moyenne de la profondeur de chaque image-clé est ramenée à 1. [7] explique plus clairement certains points de cette méthode.

4.16 ORB-SLAM2 : An Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras

[12]

- Problème : L'objectif est d'avoir un algorithme de SLAM Open-Source qui fonctionne sur un maximum de type de caméras. Pour être adapté à tous les types de caméras, on extrait des points clés des images avant de les effacer. La suite des opérations est ensuite entièrement basée sur ces points clés.

- Solution : Cette nouvelle implémentation d'un ORB-SLAM amélioré donne des résultats capables de tourner sur un ordinateur traditionnel et donne des résultats au moins équivalent aux meilleures méthodes de SLAM existentes lors de la publication de ce papier. Pour cette méthode, il a été prouvé que pour une caméra avec un capteur de profondeur il vaut mieux faire du Bundle Adjustment que d'autres méthodes pour obtenir la meilleure précision possible alors que le Bundle Adjustment consomme moins de ressources informatiques que ces autres méthodes.

4.17 Dynamic SLAM : The Need For Speed

[11]

- Problème : La majorité des SLAM supposent que le monde est majoritairmeent statique. Pour fonctionner cette méthode nécessite une estimation de la profondeur des objets sur l'image et donc soit une caméra stéréo soit une caméra avec un capteur de profondeur. En effet, le problème est de déterminer les objets qui se déplacent et les objets fixent pour éviter que les objets dynamiques faussent complètement les résultats du SLAM.

- Solution : Le principe est d'arriver à extraire des informations sémantiques de l'image pour reconnaître des objets et pouvoir identifier les objets fixes et les objets mobiles pour pouvoir les différencier pour la localisation. La suite de ce travail permet d'étendre l'algorithme de

Dynamic SLAM aux caméras monoculaires. Par ailleurs, aujourd'hui le temps de calcul est beaucoup trop important pour permettre de la localisation en temps réel.

4.18 Pop-up SLAM : Semantic Monocular Plane SLAM for Low-texture Environments

[20]

- Problème : Les méthodes existentes de SLAM ne sont pas robustes dans des environnements avec peu de points d'intérêts. Ainsi, dans ces cas là le mapping et la localisation sont très imprécis et donnent peu d'informations. Pour palier à ce problème, cet article propose un SLAM plan monoculaire en temps réel.
- Travail précédent : L'objectif est de créer un modèle 3D "pop-up" depuis une seule image. Pour cela, il faut trois étapes. La première est une détection de sol grâce à des réseaux de neurones. Ensuite, il faut tracer des lignes sur les contours du sol détectés. Enfin, le plan représentant le sol ayant été représenté par les lignes tracées à l'étape précédente, on peut extraire le modèle 3D de la pièce puisque l'on sait que le sol est plan.
- Solution : On met en place un graphe de facteur qui comprend les différentes poses détectées, les plans par rapport auxquels on peut se repérer, les mesures de ces plans provenant du modèle pop-up 3D et les informations provenant de l'odométrie. Pour la localisation on utilise alors 3 données provenant de ce graphe : la différence entre les normales des différents plans, la distance entre les deux plans et le recoupement des deux plans après projection de l'un sur l'autre.
- Conclusion : Cette méthode fonctionne là où les autres méthodes de SLAM donnent de mauvais résultats et permet d'obtenir une carte 3D dense de ces environnements difficiles. Dans des environnements comme des couloirs on peut avoir moins de 5 pourcents d'erreur en utilisant cette méthode. Néanmoins, cet algorithme ne fonctionne pas du tout dans des environnements plus ouverts.

4.19 Thèse : Estimation continue de la pose d'un équipement tenu en main par fusion des données visio-inertielles pour les applications de navigation piétonne en milieu urbain

[1]

- Problème : Avec le dérèglement climatique et la forte pollution présente dans les villes, la majorité des autorités publiques essaient de limiter les moyens de déplacement mauvais pour l'environnement et de favoriser les déplacements à pied, en transport en commun et en vélo. Cela veut donc dire plus de personne se déplaçant sur les trottoirs et donc plus de piétons en quête de leur chemin. Par ailleurs, ces piétons disposent en grande partie de téléphones portables qui contiennent différents capteurs pouvant être utilisés pour les localiser et estimer précisément leur position. Dans toute cette thèse, on supposera le téléphone du piéton comme autonome, c'est à dire sans aucune connexion extérieure que ce soit au réseau GNSS ou à Internet. Pour tous les capteurs, la ville représente un milieu très complexe. Pour les GPS les environnements relativement clos sont fréquents et les canyons urbains détériorient eux aussi grandement la qualité du signal GPS. Pour la caméra, nombre de réflexions spéculaires, de changement de luminosité ou de blocage de la caméra par le passage d'un autre piéton créent des difficultés. Pour les magnétomètres, les bâtiments sont composés de matériaux ferro-magnétiques et les autres piétons peuvent eux aussi transporter des objets capables de perturber le champ magnétique. Pour les

centrales inertielles, la démarche d'un piéton n'est absolument pas constante. Un piéton s'arrête pour traverser la rue, fait des grands pas pour la traverser puis reprend à une démarche plus lente. Enfin, le principe est de se limiter à des capteurs de faible qualité puisque ce sont les capteurs présents dans les téléphones communs. L'objectif final est donc la localisation précise de piéton dans un objectif de permettre un affichage du monde autour de lui en réalité augmentée.

- Pourquoi ne pas prendre le GPS : trop peu de signaux GPS captés, le calcul nécessaire une fois le signal reçu et enfin toutes les contraintes liés à la distance entre le GPS et le récepteur : variation entre l'émission du signal et sa réception, dérive des horloges des récepteurs... A tout cela, il faut rajouter des signaux dégradés par l'environnement dans lequel se trouve le piéton et toutes les réflexions des signaux GPS que peuvent provoquer les murs et les toits des bâtiments.

Bibliography

[1] Nicolas Antigny. Thèse : Estimation continue de la pose d'un équipement tenu en main par fusion des données visio-inertielles pour les applications de navigation piétonne en milieu urbain, 2018. [accessed 12-January–2020].

[2] Antoni Rosinol Marcus Abate Yun Chang Luca Carlone. Kimera : an open-source library for real-time metric-sematic localization and mapping. International Conference on Robotics and Automation, 2020. [accessed 30-December-2020].

[3] Cédric Demonceaux. Etude de caméras sphériques : du traitement des images aux applications en robotique. Computer Vision and Pattern Recognition. Universit´e de Bourgogne, 2012. <tel-00862979>, 2013. [accessed 12-January–2020].

[4] GoPro. Gpmf parser. `<https://gopro.github.io/gpmf-parser/>`. [accessed 22-December-2020].

[5] Nicolas Antigny Hideaki Uchiyama Myriam Servières Valérie Renaudin Diego Thomas Rin ichiro Taniguchi. Solving monocular visual odometry scale factor with adaptive step length estimates for pedestrians using handheld devices. `<https://www.mdpi.com/1424-8220/19/4/953/htm/>`, 2019. [accessed 22-December-2020].

[6] Takafumi Taketomi Hideaki Uchiyama Sei Ikeda. Visual slam algorithms : a survey from 2010 to 2016. IPSJ Transactions on Computer Vision and Applications, 2017. [accessed 30-December-2020].

[7] Jürgen Sturm Jakob Engel and Daniel Cremers. Semi-dense visual odometry for a monocular camera. 2013 IEEE International Conference on Computer Vision, 2013. [accessed 10-January–2020].

[8] Thomas Schöps Jakob Engel and Daniel Cremers. Lsd-slam : Large-scale direct monocular slam. European Conference on Computer Vision ECCV, 2014. [accessed 10-January–2020].

[9] Vladlen Koltun Jakob Engel and Daniel Cremers. Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018. [accessed 30-December-2020].

[10] Lewis Baker Jonathan Ventura Stefanie Zollmann Steven Mills Tobias Langlotz. Splat: Spherical localization and tracking in large spaces. International Conference on Virtual Reality and 3D User Interfaces (VR), 2020. [accessed 22-December-2020].

[11] Robert Mahony Mina Henein, Jun Zhang and Viorela Ila. Dynamic slam : The need for speed. IEEE International Conference on Robotics and Automation (ICRA), 2020. [accessed 10-January–2020].

[12] Raùl Mur-Artal and Juan D. Tardos. Orb-slam2 : An open-source slam system for monocular, stereo and rgb-d cameras. IEEE TRANSACTIONS ON ROBOTICS, VOL. 33, NO. 5, OCTOBER 2017, 2017. [accessed 10-January–2020].

[13] Alexis Dupuis Myrian Servières, Valérie Renaudin and Nicolas Antigny. Visual and visual-inertial slam : State of the art, classification and experimental benchmarking. Not Puslished Yet, 2021. [accessed 12-January–2020].

[14] Valérie Renaudin Nicolas Antigny, Myriam Servières. Hybrid visual and inertial position and orientation estimation based on known urban 3d models. International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2016. [accessed 22-December-2020].

[15] Valérie Renaudin Nicolas Antigny, Myriam Servières. An inertial, magnetic and vision based trusted pose estimation for ar and 3d data qualification on long urban pedestrian displacements. <https://hal.archives-ouvertes.fr/hal-01715807/>, 2017. [accessed 22-December-2020].

[16] Valérie Renaudin Nicolas Antigny, Myriam Servières. Pedestrian track estimation with handheld monocular camera and inertial-magnetic sensor for urban augmented reality. <https://ieeexplore.ieee.org/document/8115934>, 2017. [accessed 22-December-2020].

[17] Valérie Renaudin Nicolas Antigny, Myriam Servières. Continuous pose estimation for urban pedestrian mobility applications on smart-handheld devices. International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2018. [accessed 22-December-2020].

[18] Valérie Renaudin Nicolas Antigny, Myriam Servières. Fusion of 3d gis, vision, inertial and magnetic data for improved urban pedestrian navigation and augmented reality. <https://hal.archives-ouvertes.fr/hal-01896361/>, 2018. [accessed 22-December-2020].

[19] Kyle O'Keefe Paul Verlaine Gakne. Tackling the scale factor issue in a monocular visual odometry using a 3d city model. <http://www.itsnt.fr/upload/Abstracts/Tackling%20the%20Scale%20Factor%20Issue%20in%20a%20Monocular%20Visual%20Odometry%20Using%20a%203D%20City%20Model.pdf/>, 2018. [accessed 22-December-2020].

[20] Michael Kaess Shichao Yang, Yu Song and Sebastian Scherer. Pop-up slam : Semantic monocular plane slam for low-texture environments. IEEE International Conference on Intelligent Robots and Systems, 2016. [accessed 10-January–2020].

[21] Zichen Zhang Shida He, Xuebin Qin and Martin Jagersand. Incremental 3d line segment extraction from semi-dense slam. Proceedings - International COnference on Pattern Recognition, 2018. [accessed 30-December-2020].

[22] Lihui Fend Shouyuan Liu, Peng Guo and AIying Yang. Accurate and robust monocular slam with omnidirectional cameras. Sensors, 2019. [accessed 30-December-2020].

[23] Fuling Tang Yihond Wu and Heping Li. Image-based camera localization : an overview. Visual Computing for Industry, Biomedicine and Art (2018) 1:8, 2018. [accessed 30-December-2020].

[24] Linhui Xiao Jinge Wang Xiaosong Qiu Zheng Rong Xudong Zou. Dynamic slam : Semantic monocular visual localization and mapping based on deep learning in dynamic environment. Robotics and Autonomous Systems 117 1-16, 2019. [accessed 22-December-2020].