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 5, 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. 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.

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

État de l'art

2.1 Calibration

2.2 SLAM

2.3 Méthodes existentes

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

[9]

- 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

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

[11]

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

[3]

- 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 [12]. 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.

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

[10]

- 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.

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

[8]

Useless

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

[7]

- 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

2.10 SPLAT: Spherical Localization and Tracking in Large Spaces

[6]

- 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.

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

[16]

- 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

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

[1]

- 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.

2.13 Visual SLAM Algorithms : a survey from 2010 to 2016

[4]

- 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.

2.14 Image-Based Camera Localization : an overview

[15]

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

2.15 Direct Sparse Odometry

[5]

- 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.

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

[13]

- 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.

Bibliography

[1] 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].

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

[3] 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].

[4] 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].

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

[6] 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].

[7] 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].

[8] 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].

[9] 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].

[10] 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].

[11] 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].

[12] 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].

[13] 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].

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

[15] 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].

[16] 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].