Optimization of visual SLAM by semantic analysis of the environment

Mathieu Gonzalez

▶ **To cite this version:**

Mathieu Gonzalez. Optimization of visual SLAM by semantic analysis of the environment. Other. Université Rennes 1, 2022. English. NNT : 2022REN1S069 . tel-03967982

HAL Id: tel-03967982
https://theses.hal.science/tel-03967982

Submitted on 1 Feb 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *AST – Signal, Image, Vision*

Par
Mathieu GONZALEZ

Optimization of visual SLAM by semantic analysis of the environment

Thèse présentée et soutenue à Rennes, le 30 Novembre 2022
Unité de recherche : IRISA UMR CNRS 6074
Thèse N° : TBA

Rapporteurs avant soutenance :

Marie-Odile BERGER	Directrice de recherche Inria, INRIA Nancy Grand Est
Cyrill STACHNISS	Professeur, Université de Bonn

Composition du Jury :

Président :	François CHAUMETTE	Directeur de recherche Inria, Centre Inria de l'Université de Rennes
Rapporteurs :	Marie-Odile BERGER	Directrice de recherche Inria, INRIA Nancy Grand Est
	Cyrill STACHNISS	Professeur, Université de Bonn
Examinateurs :	Javier CIVERA	Professeur, Université de Saragosse
Dir. de thèse :	Eric MARCHAND	Professeur des Universités, Université de Rennes 1, IRISA
Co-dir. de thèse :	Jérôme ROYAN	Ingénieur de recherche, IRT b<>com

Invité(s) :

Amine Kacete	Ingénieur de recherche, IRT b<>com

Acknowledgement

First of all I would like to thank the members of the jury, Pr. Cyrill Stachniss, Pr. Marie-Odile Berger, Pr. Javier Civera and Pr. François Chaumette for reviewing my thesis and for all their constructive remarks and questions.

Then I would like to thank my advisors, Pr. Eric Marchand, Dr. Jérôme Royan and Dr. Amine Kacete for their guidance and trust during those 3 unforgettable years. I could have never dreamt to have better advisors.

I'd like to thank Eric Marchand for teaching me 3D vision and how to do research, for his complete availability and his time, Jérôme Royan, for his experience about SLAM and the industry and his trust and Amine Kacete for always being nice, smiling and sharing his experience of research.

Thank to all my colleagues at IRT b<>com Albert, Taha, Matthieu, Pierre, Stéphane, Guillaume, Duong, Baptiste, Orhiane, Yasser, Vincent, Lucie, Yitian, Maxime, Nathan, Jean-Marie, Maud, Martin, Nicolas,Boris. They are all brilliant persons and working with them was a pleasure during those years.

A special thank to my in-laws, Anne, René, Denis and Guillaume, who have welcomed me in their home, in their family and who have supported me, particularly during rough covid times.

This thesis would have never been possible without the always support and love of my parents Hélène and Manuel, they own 50% of this work. Thank you for your never ending trust and love.

Finally, I deeply thank my SO, Lucie, who has always been here for me, listening me talking about Jacobians and Lie algebras, supporting me, trusting me and helping me diconnecting from work. Thank you for your patience and never ending love. You also own 50% of this work.

Those past 3 years were incredible, filled with emotions, hard work and amazing encounters, I will always remember this experience.

Abstract

Visual Simultaneous Localization and Mapping (SLAM) has been heavily studied in the past couple decades. The reasons for this infatuation are the promising future technologies requiring camera pose estimation, such as augmented reality, self-driving vehicles and robotics. The goal of SLAM is to estimate the pose of a sensor (in our case a camera) moving in space while simultaneously building a map of the environment. The past few years have seen the implementation of SLAM systems able to accurately estimate the pose of a camera in small and large scenes with a precision of a few to a few tens of centimeters. However those approaches represent the world using purely geometric and appearance information. This kind of low level data is complex to use for other algorithms. For example an augmented reality application that would try to overlay virtual content on specific parts of the scene may need higher level data to be able to cluster the scene into multiple objects. Thus there is a *semantic gap* between the internal SLAM representation and the real world in which the system is evolving.

For the past decade deep learning has gained an increased popularity in computer vision, allowing to solve complex problems such as image classification, object detection or image segmentation. Those networks are today accurate and fast enough to be integrated into a real time SLAM system, resulting in a semantic SLAM system. Such system is able to build a higher level semantic map in which 3D points are associated with an object class information, such as *road, car, person, cat.* Some approaches have been developed to use this piece of information to improve the SLAM, yet the integration of semantic information into SLAM remains relatively partial.

Our goal in this manuscript is to build a SLAM system that can harness both semantic information and a priori knowledge about object classes to push forward the limits of SLAM. To this end, our contributions are:

- We first propose a light neural network to estimate the pose of objects in the scene. Objects can serve as high level landmarks for a SLAM system, improving camera pose and adding information into the map. This network however has to be trained for specific objects, which can limit its applicability in the SLAM system.
- Thus in our second work we propose a SLAM system that can create clusters

of 3D points corresponding to generic objects in the scene. With some a priori knowledge about object classes we can estimate their geometry in real time to improve both the map and camera pose estimation. However this approach only takes into account static objects, which limits the scenarios in which it can be used.

- To solve the problem of moving objects we propose a new SLAM implementation able to robustly estimate camera pose in dynamic scenes and to estimate the trajectories of all moving objects in the scene. A priori knowledge allows us to constrain the movement of objects to be plausible with respect to the structure of the world. We show in various experiments on public datasets that our approaches produce competitive results with state-of-the-art.
- Finally we improve upon this last approach by injecting LiDAR data into our visual SLAM system. Doing so we show that we can estimate object canonical poses and track them with an improved accuracy while also recovering object shapes.

We think that those contributions are steps towards an accurate worldwide SLAM system able to build a digital twin of our world, in which all objects are mapped and tracked.

Résumé

Les algorithmes de localisation et cartographie en simultané (Simultaneous Localization And Mapping, SLAM) ont été étudiés en profondeur durant les deux dernières décennies. Les raisons de cet engouement résident dans les technologies futures prometteuses qui nécessitent une estimation de la pose de la caméra, telle que la réalité augmentée, les véhicules autonomes et la robotique. Le but du SLAM est d'estimer la pose d'un capteur (dans notre cas une caméra) se déplaçant dans l'espace tout en construisant une cartographie de l'environnement. Depuis quelques années des implémentations de SLAM permettent d'estimer précisément la pose de la caméra dans des scènes petites et grandes avec une précision de quelques centimètres à quelques dizaines de centimètres. Cependant ces approches représentent le monde en utilisant uniquement des information géométriques et d'apparence. Ces informations bas niveau sont complexes à utiliser. Par exemple une application de réalité augmenté pourrait avoir besoin d'afficher du contenu sur des parties spécifiques de la scène. Cela nécessite d'avoir accès à une information haut niveau pour partitionner la scène en différents objets qui la composent. Ainsi il y a un *écart sémantique* entre la représentation interne du SLAM et le monde réel dans lequel le système évolue.

Au cours de la dernière décennie, l'apprentissage profond a gagné une popularité croissante dans le domaine de la vision par ordinateur en permettant de résoudre des problèmes complexes comme la classification d'images, la détection d'objets ou encore, la segmentation d'images. Ces réseaux sont aujourd'hui suffisamment précis et rapides pour être intégrés dans un système de SLAM, créant ainsi un SLAM sémantique. Un tel système est capable de construire une cartographie sémantique haut niveau dans laquelle les points 3D possèdent une information de classe d'objet, comme *route, voiture, peronne, chat.* Certaines approches ont été développées pour utiliser ces informations afin d'améliorer le SLAM mais l'intégration de l'information sémantique dans le SLAM reste relativement partielle.

Notre but dans ce manuscrit est de construire un système de SLAM qui peut exploiter l'information sémantique ainsi qu'une connaissance a priori sur les objets pour repousser les limites du SLAM. Dans ce but nous proposons plusieurs contributions:

- Tout d'abord un réseau de neurones léger pour estimer la pose d'objets dans la

scène. Les objets peuvent servir de repères haut niveau pour un SLAM, améliorant la pose de la caméra et ajoutant de l'information dans la cartographie. Le principal inconvénient de ce réseau est qu'il nécessite d'être entraîné pour des objets spécifiques, ce qui peut limiter les cas d'usages du SLAM.

- Ainsi dans notre seconde contribution nous proposons un SLAM capable de créer des groupes de points 3D correspondant à des objets génériques dans la scène. En utilisant une connaissance a priori sur la classe des objets nous pouvons estimer leur géométrie pour améliorer la cartographie et la pose de la caméra. Cette approche en revanche ne prend en compte que les objets statiques, ce qui limite les scénarios dans lesquels on peut l'utiliser.
- Pour résoudre le problème de scènes dynamiques avec des objets mobiles nous proposons un SLAM capable d'estimer la pose de la caméra ainsi que la trajectoire de tous les objets dans la scène. Un a priori sur les objets nous permet de contraindre leurs mouvements afin qu'ils soient cohérents avec la structure du monde. Nous démontrons dans diverses expériences sur plusieurs jeux de données que notre approche produit des résultats compétitifs avec l'état de l'art.

Nous pensons que ces contributions sont des étapes vers un SLAM à très large échelle, précis, capable de contruire un jumeau numérique de notre monde, dans lequel les objets sont cartographiés et suivis.

Table of Contents

N'oublie jamais, celui qui croit savoir n'apprend plus.
Pierre Bottero, *Ellana*

Le doute est une force. Une vraie et belle force.
Veille simplement qu'elle te pousse toujours en avant.
Pierre Bottero, *Ellana*

SYNTHÈSE EN FRANÇAIS

Optimisation du SLAM visuel par analyse sémantique de l'environnement réel

Introduction

La navigation visuelle est une compétence inée chez de nombreux êtres vivants. Elle leur permet de se localiser dans l'espace afin d'y naviguer et de l'explorer lorsqu'il est inconnu. Cette capacité est tellement naturelle que nous avons parfois du mal à nous rendre compte à quel point elle est puissante. Qu'il pleuve ou non, que l'environnement soit encombré ou vide et qu'il soit changeant ou statique nous sommes capables de comprendre où nous nous situons. Depuis plusieurs décennies il y a une demande croissante de la part de l'industrie afin d'intégrer une telle capacité dans un algorithme. Elle est en effet essentielle pour de nombreux domaines tels que les robots et véhicules autonomes ainsi que la réalité augmentée. La plupart des systèmes autonomes requièrent en effet de connaître leur position dans l'espace afin qu'ils puissent s'y déplacer. Il en va de même pour un système de réalité augmentée qui a besoin de cette information pour supperposer du contenu virtuel a une image réelle de manière consistente. Plusieurs capteurs différents peuvent être utilisés, comme par exemple des centrales inertielles, LiDARs, GPS mais les caméras sont particulièrement intéressantes. En effet elles peuvent être de petites tailles, avoir une faible consommation, arrivent a représenter efficacement l'environment qui les entoure et sont déjà présentes dans de nombreux appareils tels que les téléphones portables. Ainsi dans ce manuscrit nous nous concentrerons particulièrement sur les problématique de localisation basée caméra. Les algorithmes de SLAM (Simultaneous Localization And Mapping) permettent, à partir du flux vidéo d'une caméra se déplaçant dans l'espace, d'estimer en temps réel la pose de cette dernière tout en créant une cartographie de

l'espace qu'elle observe. Ils permettent donc de répondre à la problématique posée pour la navigation autonome et la réalité augmentée. Cependant, pour certains systèmes, la connaissance de la seule pose de la caméra n'est pas suffisante. Connaître la pose des autres objets ou agents dans la scène peut également être nécessaire. C'est par exemple le cas des véhicules autonomes qui ont besoin d'estimer la pose et la trajectoire des autres véhicules et des piétons dans l'environnement afin de pouvoir y naviguer de façon sûre. De même pour la réalité augmentée, certaines applications peuvent avoir besoin d'afficher du contenu virtuel en interaction avec des objets et ainsi de connaître leur pose par rapport à la caméra. Ainsi, des algorithmes d'estimation de pose objet, capables de prédire à partir d'une seule image la pose relative entre la caméra et un objet d'intérêt dans la scène ont été développés. Ces algorithmes ont récemment profité des développements de l'apprentissage profond pour gagner en précision. Ils peuvent être vus comme une version mono image et duale du SLAM. En effet le SLAM cherche à prédire la pose de la caméra relativement au monde alors que l'estimation de pose objet cherche a prédire la pose de l'objet relativement au monde. Si l'objet est statique ces deux transformations sont l'inverse l'une de l'autre à une transformation constante près. Les problématiques de SLAM et d'estimation de pose objets sont donc profondément liées. Résoudre ces deux problématiques en parallèle a donc un intérêt particulier.

Nos objectifs durant cette thèses étaient:

- D'implémenter un SLAM capable de créer une cartographie sémantique.
- De tirer avantage de l'information supplémentaire fournie par la sémantique afin d'améliorer la pose de la caméra et la cartographie, particulièrement dans des scénarios dynamiques.
- D'estimer la pose d'objets mobiles dans la scène.
- D'implémenter un système suffisamment générique pour être appliqué à des séquences variées.

Pour répondre à ces objectifs nous avons développés plusieurs algorithmes que nous présentons dans ce manuscrit et que nous résumons dans cette synthèse.

État de l'art

Un effort particulier a été apporté à la rédaction d'un état de l'art sur le SLAM sémantique lors de cette thèse. Dans cette section nous en présentons un rapidé résumé.

SLAM classique

Un des papiers de SLAM les plus cités est ORB-SLAM 2 [Mur-Artal and Tardós, 2017]. Dans cet algorithme, inspiré de [Klein and Murray, 2007; Mouragnon et al., 2006], la localisation de la caméra et la cartographie s'effectuent en parallèle. Ceci permet de suivre la caméra en temps réel sans avoir a re-cartographier des points à chaque instant. De plus certaines images clés (appelées keyframes) sont sélectionnées parmi toutes les images. Ceci permet de diminuer la quantité d'information à traiter par l'algorithme d'optimisation d'ajustement de faisceaux (ou bundle adjustment). Cet algorithme étant coûteux, cela permet tout de même grâce à la parallelisation de l'appliquer à un SLAM temps réel. ORB-SLAM 2 est un SLAM indirect car il extrait des points caractéristiques (ou keypoints) des images et optimise la pose de la caméra et la cartographie en minimisant l'erreur de reprojection entre les points 3D et les points caractéristiques. D'autres algorithmes, tel que LSD-SLAM [Engel et al., 2014], sont qualifiés de direct car ils minimisent directement l'erreur photométrique, c'est à dire la différence de luminosité entre les pixels. Ce faisant ils sont capables de fonctionner dans des scénarios où les points caractériques ne sont pas disponibles (e.g. des zones non texturées ou des images floues).

Utilisation de la sémantique pour la création de cartes sémantiques

Plusieurs algorithmes se sont concentrés sur la création de cartographies sémantiques qui puissent ensuite être utilisées pour améliorer le SLAM. Plusieurs stratégies sont possibles, comme par exemple la segmentation d'une cartographie 3D en utilisant des réseaux de neurones ou bien la segmentation d'image 2D qui doivent ensuite être fusionnées pour créer une cartographie consistente. Nous nous sommes plus particulièrement intéressés à la deuxième partie, plus cohérente avec les contraintes d'exécution temps réel du SLAM. Un des premiers papiers publiés pour créer une cartographie sémantique utilisant des images segmentées par réseaux de neurones est [McCormac et al., 2017] qui utilise un SLAM dense basé sur des images RGB-D. Dans ce papier un réseau de convolution estime la probabilité que les pixels d'une image appartiennent à un ensemble de classe. Ces probabilités sont ensuite fusionnées lorsqu'un point 3D possède plusieurs observations 2D. Plus précisément la fusion se base sur l'inférence bayésienne à partir des densités de probabilité. Cette fusion leur permet d'obtenir une carte plus consistente que chacune des segmentations 2D. Diverses approches ont également été proposées pour améliorer les résultats ou les adapter à d'autres types de SLAM [X. Li and Belaroussi, 2016; C. Yu et al., 2018].

Utilisation de la sémantique pour la relocalisation

Le problème de relocalisation consiste a estimer la pose de la caméra lorsque le suivi de la caméra est perdu. Les algorithmes se basent généralement sur une cartographie pré-existente ainsi qu'un ensemble d'images décrivant la scène. C'est un problème particulièrement complexe car il peut y avoir des changements visuels importants entre l'image courante et les images capturées lors de création de la cartographie (e.g différences de luminosité jour-nuit, de saison, etc.). Plusieurs approches ont proposé d'utiliser l'invariance apportée par les réseaux de segmentation pour répondre à ce problème. On peut par exemple citer [Toft et al., 2017] qui à partir d'une image et d'une carte sémantique optimise une erreur de reprojection sémantique. Cette erreur est calculée comme la distance 2D entre la position d'un point reprojeté et la zone de même classe sémantique la plus proche. Ainsi lorsqu' un point est reprojeté dans la bonne classe sémantique, son erreur est nulle. Les expérience montrent que cette approche est moins précise mais plus robuste que l'état de l'art. [Larsson, Stenborg, Toft, et al., 2019] propose de l'améliorer en créant une segmentation plus granulaire de l'image. [Arandjelović and Zisserman, 2014] se base sur l'idée que deux keypoints appareillés doivent avoir la même classe sémantique pour réduire le nombre de mauvais matchs et ainsi améliorer et accélérer le processus de relocalisation. D'autres algorithmes encore utilisent la sémantique pour se concentrer sur des zones stables dans le temps, telles que les bâtiments ou la route.

Utilisation de la sémantique pour la gestion des objets dynamiques

Les objets dynamiques representent un réel problème pour le SLAM classique. En effet ce dernier se base sur l'hypothèse du monde statique qui considère que le monde est un unique objet rigide. Dès lors qu'un objet bouge dans la scène la précision de l'estimation de la pose de la caméra peut être dégradée, de même que la qualité de la carte. Ceci limite fortement les cas d'usage du SLAM dans des scénarios réels. Pour s'affranchir de cette hypothèse certains systèmes utilisent la sémantique pour masquer les objets potentiellement dynamiques. C'est par exemple le cas de [Bescos et al., 2018]. L'inconvénient de ce genre d'approche est que certains objets potentiellement dynamiques peuvent être en réalité statique (e;g. une voiture garée), les masquer entraîne une perte d'information et éventuellement une dégradation de la qualité des estimations. Certaines approches utilisent donc les parties a priori statiques de l'image pour fournir une première estimation de la pose de la caméra. Cette dernière est ensuite utilisée pour détecter si les zones

potentiellement dynamiques le sont ou non et les inclure dans le SLAM dans ce dernier cas. On peut par exemple citer [Cui and Ma, 2019; C. Yu et al., 2018] qui estiment une matrice fondamentale en utilisant les points statiques et vérifient la dynamicité des points restants en calculant leur distance à la ligne épipolaire.
D'autre approches cherchent à utiliser les points correspondants aux objets potentiellement dynamiques afin de les suivre et d'estimer leur trajectoire. C'est ce qui est fait par [Bescos et al., 2021], qui applique également une optimisation d'ajustement de faisceaux pour raffiner toutes les poses de caméras ainsi que la pose de tous les objets potentiellement mobiles.

SLAM basé objets

Certains algorithmes proposent d'utiliser les objets dans la scène comme des ancres, servant à améliorer la pose de la caméra et à obtenir une cartographie de plus haut niveau. On peut distinguer trois catégories d'approches. Tout d'abord celles qui se basent sur des détecteurs d'objets spécifiques et requièrent donc que des objets particuliers soient présents dans la scène [Civera et al., 2011; Fioraio and Di Stefano, 2013; Salas-Moreno et al., 2013]. Ces travaux estiment la pose d'objets qu'ils utilisent ensuite dans le SLAM afin de contraindre la pose de la caméra. Un exemple de contrainte est la consistance de la pose de l'objet sur l'ensemble des images. Deuxièmement, des papiers proposent d'utiliser des détecteurs d'objets génériques, qui prédisent simplement une boite englobante autour de chaque objet. L'objet est ensuite représenté avec géométrie siplifiée, comme une quadrique [Gaudillière et al., 2019; Hosseinzadeh et al., 2018; Nicholson et al., 2018] ou des boites 3D [S. Yang and Scherer, 2019a]. Enfin, une génération très récente d'approches se base sur des détecteurs génériques mais utilise des réseaux de neurones pour reconstruire chaque objet avec ses caractéristiques géométriques spécifiques. DSP-SLAM [J. Wang et al., 2021] optimise le code latent d'un réseau de neurone afin que la forme qu'il génère corresponde aux points 3D de l'objet dans le SLAM. Ceci leur permet d'une part d'obtenir une représentation géométrique complète et proche de la réalité mais également d'estimer la pose des objets, qui vient contraindre la pose de la caméra.

Contributions

Dans ce manuscrit nous proposons de dépasser les limites du SLAM classique, qui repésente le monde en utilisant un unique nuage de point homogène. Pour cela nous

représentons la scène en utilisant un graphe dont les noeuds correspondent aux objets de la scène et les arêtes à des liaisons entre les objets. Les noeuds possèdent des propriétés intrinsèques, telles que leur pose dans le monde, leur texture, leur géométrie, qui sont en partie dépendantes de leur classe sémantique (par exemple la plupart des voitures ont des géométries similaires). Les liaisons entre les noeuds peuvent être multiples et dépendent de la classe des noeuds. Par exemple entre une voiture et la route il ya une contrainte de type *support* : la voiture doit reposer sur la route. Partant de cette nouvelle représentation pour le SLAM nous proposons les contributions suivantes:

- Un système d'estimation de pose objet basé sur de l'apprentissage profond, pouvant être intégré dans un SLAM basé objet. Cet algorithme, appelé L6DNet extrait des patchs d'une image RGB-D qu'il classifie afin de ne conserver que les patchs représentant l'objet d'intérêt. Chaque patch est ensuite utilisé par un réseau de neurones afi de prédire un ensemble de points 3D à la surface de l'objet. Les points 3D sont ensuite aggrégés de manière robuste et associés à des points définis a priori pour calculer la pose relative de l'objet à la caméra. Notre approche hybride obtient des résultats similaires à l'état de l'art en terme de précision de pose. En revanche elle nécessite moins de ressources, tant en terme de capacité de calcul que de quantité de données. Il est donc plus facile d'entraîner notre approche afin de l'intégrer dans un SLAM. Elle nécessite cependant un entraînement sur des objets spécifiques, ce qui peut limiter les cas d'usage du SLAM basé objet. Cet algorithme a été présenté dans les papiers: "**L6DNet: Light 6 DoF Network for Robust and Precise Object Pose Estimation with Small Datasets** par Mathieu Gonzalez, Amine Kacete, Albert Murienne et Eric Marchand, publié dans **IEEE Robotics and Automation Letters (RA-L)**, 2020 ainsi que dans **IEEE International Conference on Robotics and Automation (ICRA)**, 2020. Il est cité dans ce manuscrit: [Gonzalez, Kacete, et al., 2021].

- Pour dépasser la limite imposée par L6DNet nous proposons un SLAM plus générique, capable de créer une cartographie sémantique pouvant contenir des objets non connus a priori. Pour cela nous utilisons un réseau de segmentation panoptique qui segmente les images 2D. Nous fusionnons ensuite dans le SLAM les images segmentées pour obtenir une cartographie sémantique.Cette cartographie est ensuite utilisée pour créer des groupes de points, appelés clusters, correspondant de façon unique aux objets dans la scène. Ceci est un premier pas vers la représenta-

tion du graphe de scène. De plus, nous mettons en place un a priori géométrique sur la classe de certains objets. Cette information nous permet de contraindre la carographie afin qu'elle soit cohérente avec l'information sémantique. Nous nous intéressons en particulier aux classes qui peuvent être représentées grâce à des plans. Avec cette approche nous montrons que nous pouvons créer une cartographie de haut niveau, contenant des structures et cohérente avec la scène. Nous montrons également ques contraintes additionnelles nous permettent d'améliorer la cartographie et l'estimation de pose de la caméra comparée aux autres systèmes de SLAM. Cette approche, bien que générique ne permet pas de gérer les objets dynamiques dans la scène, ce qui peut limiter ses cas d'usage. Cet algorithme a été présenté dans le papier: "**S^{3}LAM: Strcutured SceneSLAM**", par Mathieu Gonzalez, Eric Marchand, Amine Kacete et Jérôme Royan, publié dans **IEEE International Conference on Intelligent Robots and Systems (IROS) 2022** ainsi que dans la conférence nationale **Groupe de Recherche et d'Etudes de Traitement du Signal et des Images (GRETSI) 2022** sous le nom **S^{3}LAM: SLAM à Scène Structurée**. Il est cité dans ce manuscrit comme : [Gonzalez et al., 2022a].

- Afin de dépasser la limite de S^3*LAM* concernant les scènes dynamiques, nous proposons un nouveau SLAM, TwistSLAM capable d'estimer la pose de la caméra dans des environnements dynamique et d'estimer la trajectoire de tous les objets dynamiques dans la scène. Pour cela nous nous basons sur S^{3}LAM et utilisons les clusters a priori statiques (tels que la route, les bâtiments etc.) afin d'estimer la pose de la caméra de façon robuste. Enuite, pour tous les objets potentiellement dynamiques (tels que les voitures, les piétons, etc.) nous estimons leur l'évolution de leur pose par rapport à la caméra en minisant l'erreur de reprojection de leurs points 3D. De plus nous proposons d'utiliser les structures de la scène afin de contraindre le mouvement des objets. Nous appliquons des contraintes mécaniques à la vitesse des objets, comme une moélisation des arêtes du graphe de scène. Par exemple une voiture est en liaison plan avec la route elle ne possède donc que 3 degrés de liberté. Nous utilisons donc un opéraeur de projection pour projeter la vitesse de la voiture afin qu'elle soit cohérente par rapport au plan estimé de la route. Toutes les poses caméras, les poses objets et les points 3D sont ensuites optimisés dans un algorithme d'ajustement de faisceau. Nous montrons que notre approche permet

d'améliorer à la fois l'estimation de pose de la caméra ainsi que l'estimation de pose des objets comparé à l'état de l'art. Cette approche a été présentée dans le papier : **TwistSLAM: Constrained SLAM in Dynamic Environment**, par Mathieu Gonzalez, Eric Marchand, Amine Kacete et Jérôme Royan, publié dans **IEEE Robotics and Automation Letters (RA-L)**, 2022 ainsi que dans **IEEE International Conference on Intelligent Robots and Systems (IROS) 2022**. Il est cité dans ce manuscrit comme : [Gonzalez et al., 2022b].

Conclusion

Dans ce manuscrit nous avons tout d'abord présenté les outils fondamentaux nécessaires à la compréhension de nos travaux. Ensuite nous avons reconstitué le fonctionnement d'un SLAM classique et présenté un état de l'art étendu sur l'usage de la sémantique dans le SLAM. Puis nous avons proposé d'appliquer la structure connue du graphe de scène au SLAM afin de créer une représentation plus haut niveau pour la cartographie, qui ne soit plus homogène mais constituée de différents objets possédant des propriétés et reliés entre eux par des contraintes. Nous soutenons le fait que cette nouvelle représentation haut niveau peut permettre des nouvelles applications en réalité augmentée et navigation autonome. De plus nous avons développé plusieurs algorithmes permettant d'intégrer une information sémantique dans un SLAM, implémentant en pratique cette représentation. Nous avons montré sur plusieurs expériences que nos contributions permettaient d'améliorer le SLAM à plusieurs niveaux et sur plusieurs jeux de données par rapport à l'état de l'art ainsi que l'estimation de pose des objets, remplissant ainsi les objectifs fixés.

List of Figures

List of Tables

Notations

General rules	
Lower case x	Scalars, except for components of 3D points.
Bold lower case $\mathbf{x}$	Vectors, except for 3D points to differentiate them from 2D image points.
Bold upper case $\mathbf{X}$	Matrices and 3D points.

Mathematics	
$\mathbb{R}^n$	Set of real numbers of dimension n
$\mathcal{M}_{n\times m}$	Set of real matrices of dimension $n \times m$
SE(3)	Special Euclidean group of dimension 3.
$\mathfrak{se}(3)$	Lie algebra associated with SE(3).
SO(3)	Special Orthogonal group of dimension 3.
$\mathfrak{so}(3)$	Lie algebra associated with SO(3).
$\mathbf{I}_n$	Identity matrix of dimension n
$\mathbf{0}_{n\times m}$	Null matrix of dimension $n \times m$
$\mathbf{S}^\top$	Transpose of $\mathbf{S}$
$\mathbf{S}^{-1}$	Inverse of $\mathbf{S}$
$\mathbf{A}^+$	Moore-Penrose pseudo-inverse
$[\mathbf{u}]_\times$	Skew matrix of $\mathbf{u} \in \mathbb{R}^3$
$[\boldsymbol{\xi}]^\vee$	Vee operator from $\mathbb{R}^6$ to $\mathfrak{se}(3)$
$[\boldsymbol{\xi}]^\wedge$	Wedge operator from $\mathfrak{se}(3)$ to $\mathbb{R}^6$
$det(\mathbf{A})$	Determinant of $\mathbf{A}$
$\exp_{\mathfrak{so}}(.)$	Exponential map from $\mathfrak{so}(3)$ to SO(3)
$\exp_{\mathfrak{se}}(.)$ or $\exp(.)$	Exponential map from $\mathfrak{se}(3)$ to SE(3)
$\log_{\mathrm{SO}}(.)$	Logarithm map from SO(3) to $\mathfrak{so}(3)$
$\log_{\mathrm{SE}}(.)$ or $\log(.)$	Logarithm map from SE(3) to $\mathfrak{se}(3)$
$\otimes$	Kronecker product of two matrices
$\|.\|_{\mathbf{A}}$	Mahalanobis norm with matrix $\mathbf{A}$

Geometry	
w, c, o	Indices representing respectively the world, the camera and an object.
$\mathbb{E}^3$	Euclidean space of dimension 3.
$\mathcal{F}_w$	Coordinate frame of w.
${}^w\bar{\mathbf{X}}$	3D point expressed in $\mathcal{F}_w$ in homogeneous coordinates.
${}^w\mathbf{X}$	3D point expressed in the coordinate frame w.
${}^w\mathbf{T}_o$	Pose matrix transforming a 3D point from $\mathcal{F}_o$ to $\mathcal{F}_w$.
${}^w\mathbf{R}_o = rot({}^w\mathbf{T}_o)$	Rotation matrix of ${}^w\mathbf{T}_o$.
${}^w\mathbf{t}_o = trans({}^w\mathbf{T}_o)$	Translation vector of ${}^w\mathbf{T}_o$.
${}^w\boldsymbol{\xi}_o$	Twist of object o expressed in $\mathcal{F}_w$.
$\boldsymbol{\omega}$	Rotational component of a twist.
$\mathbf{v}$	Translational component of a twist.
${}^w\mathbf{V}_o$	Adjoint map from $\mathcal{F}_o$ to $\mathcal{F}_w$.
$\mathbf{x}$	2D point in image space.
$\bar{\mathbf{x}}$	2D point in homogeneous coordinates.
π	Projection function from 3D space to image space in pixel coordinates.
$\mathbf{K}$	Intrinsic matrix of the camera.
$\mathbf{C}$	Camera center in $\mathbb{R}^3$.
f	Focal length of the camera in metric units.
f_x, f_y	Focal length of the camera in pixels.
c_x, c_y	x and y coordinates of the camera principal point in pixels.

INTRODUCTION

Context

Visual self localization is a natural capacity for most living beings, allowing them to explore the unknown world as well as to navigate in a familiar environment. This capacity is so natural for us that we may have trouble realizing how powerful it is. Whether it is rainy or sunny, whether the environment is crowded or empty, whether none or many parts of the environment have changed since we last saw it, we are still able to understand where we are. For the past few decades there has been a growing demand from the industry to embed an algorithm with such capacity. Indeed it is seen as one of the keystones of multiple domains such as autonomous robots, self-driving vehicles and augmented reality. Multiple sensors and modalities can be used, such as Inertial Measurement Units (IMUs), LiDARs, GPS, each one with their pros and cons. However cameras hold a special place in the kingdom of sensors since they are low cost, consume little energy, need little space, depict well their environment and are already present in many devices such as smartphones. In this thesis we will focus mostly on vision based approaches, however it seems natural that the most promising solutions combine different modalities (e.g. an IMU and a camera) to counteract their weaknesses. This is why in the final chapter we will integrate LiDAR data into a stereo-based SLAM system to improve it.

Autonomous agents

The goal of an autonomous agent (robot or vehicle) is to navigate safely in its environment without any human intervention to accomplish tasks, such as *"Travelling from Zaragoza to Bonn while passing by Rennes and Nancy"* for a self-driving car. Those tasks are currently highly trendy to facilitate our lives or decrease the number of casualties on the roads that can be caused by drivers' inattention. Hence most of cars manufacturers such as Toyota with its self driving e-Palette shuttles at Tokyo's Olympic games invest funds to develop increasingly autonomous vehicles.

Some of those systems require an estimation of the agent location and a map of the environment to understand how to move within it. They also require an estimation of the

Figure 1 – Illustration of a self-driving car. The car needs to detect entities and agents in the scene (road lanes, traffic signs, cars) to safely navigate. Image courtesy of Adobe europarl.europa.eu.

pose of the agent relative to other agents in the scene (e.g. other cars or pedestrians) to safely interact with them, as it is illustrated in figure 1.

Augmented Reality

The goal of Augmented Reality (AR) is to overlay virtual objects on images from the real world. Doing so the users perceive those objects as part of the environment, seamlessly providing them with additional information. An example of AR application is for industrial manufacturing: the sight of an operator in a factory can be augmented with instructions to follow, directly attached to the environment he is working on. This could allow him to better follow a procedure without having to stop to read a guide. Doing so, the distance between the operator and the task decreases, which facilitates the task, accelerates it and reduces error rates.

Virtual objects can be attached to specific parts of the scene in which the operator moves. To correctly project them on a device such as a smartphone or AR glasses, the pose of the device relative to the scene must be estimated.

In some applications the AR content must be attached to objects that can move within the scene. In those cases knowing the pose of the device relative to the scene is not enough, the pose of the object relative to the device must also be known. An example of such an AR application can be seen in figure 2.

Figure 2 – Example of augmented reality application for the industry, where virtual content is attached to an object. Image courtesy of ptc-solutions.de.

Object pose estimation and Simultaneous Localization and Mapping

For both AR and autonomous navigation the 6 degrees of freedom (DoF) pose of some objects must be estimated, along with the pose of the camera. On the one hand, 6-DoF object pose estimation is solved using a single image of the object of interest. Various solutions exist for example by rigidly attaching markers to the object or by using a data driven algorithm to directly predict the pose of the object relative to the camera given a single image. On the other hand, camera pose estimation can be obtained through Simultaneous Localization And Mapping (SLAM). The goal of SLAM is to estimate the pose of a camera moving in space, while simultaneously building a map of the environment. This second goal is highly dependent on the first one and vice-versa. Indeed on the one hand a map of the environment is required to accurately estimate the camera pose. On the other hand a good estimation of the camera location is needed to build a precise map of the environment. This explains why the SLAM is often qualified to be a *chicken-and-egg* problem. The problem of camera pose estimation is also highly related to object pose estimation. Indeed if an object is static in the scene, its pose with respect to the camera is the inverse of the pose of the camera with respect to the scene (up to a multiplication by a static transformation corresponding to the pose of the object in the scene). Camera pose estimation and object pose estimation can thus be seen as dual problems of one an other, that can also benefit from each other. This, added to the fact that many domains require to solve both justifies the idea of building an object pose estimation system, nested in a

SLAM algorithm.

Limits and challenges of SLAM

Visual SLAM has been studied for a couple decades and was sometimes considered to be solved. This mainly depends on the environment in which the camera evolves and on the expected accuracy and robustness. For example a robot evolving on a 2D plane in a controlled environment can be precisely localized. The same can not be said however for a self-driving car in a crowded pedestrian street under the rain at night. In [Cadena et al., 2016] the authors list challenges that should be adressed to make SLAM reliable in most scenarios. We list here some of them that can benefit from semantic information, which is the knowledge of which types of objects are present in the scene.

Making relocalization more robust.

The relocalization is a very specific step in the SLAM pipeline, its goal is to estimate the pose of the camera given a 3D map. However it is crucial to make the SLAM robust to tracking losses. It is challenging as the map can be created and observed at very distant moments. This impacts the relocalization as visual information depicting a scene at night is drastically different when the image is taken during the day. Those changes are also important from one season to another, making long term data association challenging. Semantic information can help improve relocalization in those cases. Problems can also arise when dealing with large viewpoint variations. In those cases, using objects as high level landmarks to compensate low level keypoints that can be wrongly matched may be beneficial.

Handling dynamic objects.

Changes in the scene may also come from dynamic objects. Indeed most SLAM systems assume that the world is static. In the majority of cases this assumption is not valid which can hurt camera pose estimation accuracy and corrupt the map, creating phantom 3D points. We can easily understand this problem by imagining ourselves in a train station, beside a stopped train that slowly accelerates. This can give the feeling that we are going in the opposite direction as the visual features from the train are moving away.

The same problem arises in SLAM when we do not know which features belong to static or dynamic objects. In the case where camera pose estimation is accurate, the

SLAM system can triangulate multiple times the same 3D point belonging to the object that moves in space, creating a trace of the object in the map. Some SLAM systems are able to mask out dynamic objects, however doing so information that may be necessary for some applications is lost. For example an AR application may need to track an object in the scene to make virtual content seamlessly interract with it.

Creating meaningful maps.

Making the SLAM build a high level semantic map is also a challenging problem. In classical SLAM systems the map can be dense or sparse but is always purely geometric and stores some low level appearance information through features descriptors. This can be problematic for some applications that need to use this map. For example in the case of augmented reality, an application may need virtual content to be overlayed on a specific part of the scene, such as a table in a kitchen. Hence the map needs additional information, corresponding to the nature of objects in the scene. Building a map, with semantic information associated to 3D points allows to reduce the gap between the SLAM internal representation and the real world. But we can also build an even higher level map, by estimating object poses to use them as landmarks, estimating their geometry, their mass and other physical properties that can be necessary for robots to safely interact with the world.

Contributions

This thesis focuses on the introduction of semantic information to improve camera pose estimation. As we saw, classical SLAM systems assume that the world is a single rigid body, represented with a single point cloud. Such a low level representation is far from the reality. Hence, in our work we propose to consider that the map of a SLAM system should be composed of sets of points, called clusters that uniquely correspond to objects within the scene. Those clusters in the map can be seen as vertices of a graph. They possess an internal set of properties that mainly depend on their semantic class. The edges correspond to links or relationships between the clusters. Examples of internal properties can be: the geometry of the object (e.g. all cars have similar geometries), its pose, wether it is dynamic or static, its texture, its size (e.g. humans are on average 171 cm tall with a standard deviation of 6 cm), etc. The links between clusters can represent constrains between the objects such as support constrains, relative pose constrains (e.g.

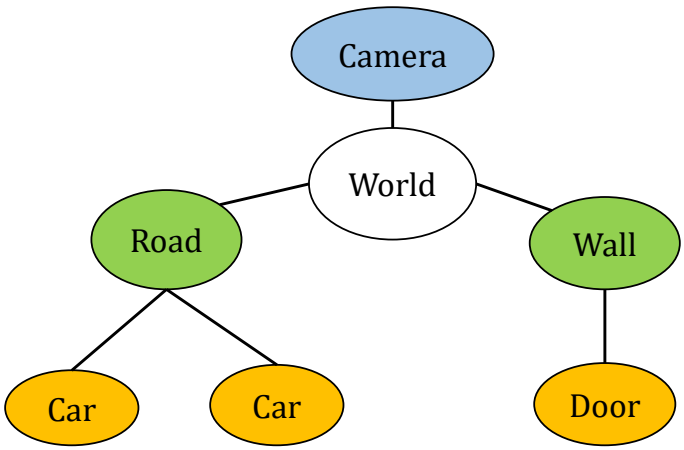

Figure 3 – Example of semantic scene graph: clusters corresponding to objects possess an internal structure and edges that link them to one another.

two cars close to each other are likely to be oriented similarly), collision constrains, etc. An example of scene graph can be seen in figure 3.

The main goal during this thesis was to implement a SLAM system with the capacity to:

- Create a semantic map that contains information about objects in the scene, using segmented stereo or monocular images.
- Take advantage of the additional semantic information to improve the accuracy of camera pose estimation and mapping, particularly in challenging cases such as dynamic scenes.
- Estimate the pose of objects moving in the scene.
- Be general enough to be used in any environment.

To answer those goals we developped several solutions. Our contributions can be summarized as follows:

- In the light of our scene graph representation we first show how to compute the relative pose between a camera and an object using a single image. This approach can then directly be integrated in an object based SLAM. It is built around our network, L6DNet, from our paper **L6DNet: Light 6 DoF Network for Robust and Precise Object Pose Estimation with Small Datasets**. In this work we show that a small network, trained with little data and time can outperform recent large networks trained with much more resources. Our strategy is hybrid and patch

based: patches are extracted from the image and classified by a first network to keep only object patches. Then a second neural network predicts for each patch a set of vectors between the patch 3D center and a set of a priori chosen 3D points on the object surface. By adding the position of the patch 3D center to each vector we obtain an estimation of the 3D position of points for each patch. This produces clusters of points that we robustly aggregate to estimate their centroid which are then associated to the a priori chosen points to estimate the object pose. This patch and offset based approach allows us to rapidly train a small neural network requiring little data and resources. The hybrid approach also facilitates its convergence by removing the pecularities of rotations in the network. Our system can infere objects poses in real-time on a single GPU, meaning that it could be integrated in object based SLAM systems to improve camera pose estimation, using objects as high level landmarks. However this approach requires to train a CNN on specific objects which limits its applicability. We address this limit in our next contribution: S^{3}LAM.
This work was published in **IEEE Robotics and Automation Letters (RA-L) vol. 6 (2), pp. 2914-2921 and in the IEEE International Conference on Robotics and Automation (ICRA) 2020.** [Gonzalez, Kacete, et al., 2021].

- Thus, we go from single image to multiple images pose estimation by defining our cluster based SLAM called S^{3}LAM, from our paper **S^{3}LAM: Structured Scene SLAM**. In this work we create clusters of points in the map corresponding to objects in the scene. This changes from the view of classical SLAM in which the map is a single rigid body represented with a single point cloud. It allows us to create a higher level semantic map in which objects possess intrinsic properties, such as their geometry. To do so, we compute the panoptic segmentation of 2D images. Semantic information and object ids are fused to create a semantic map in which points are clustered, each cluster representing a single object. For some a priori defined clusters we can then robustly fit a plane, thus obtaining higher level information about the scene structure. By modifying the bundle adjustment formulation we constrain object points to respect the estimated geometry of their corresponding cluster. Doing so, we improve the accuracy of both the map and the estimated camera poses. With this approach we push the limits of our previous paper L6DNet by allowing our SLAM system to estimate the pose of generic pla-

nar objects without the need of training a specific object detector. However this approach is still limited in dynamic scenes as it unable to track moving objects. We address this limit in the following chapter, TwistSLAM.
This work was published in the **IEEE International Conference on Intelligent Robots and Systems (IROS) 2022** and in the french conference **Groupe de Recherche et d'Etudes de Traitement du Signal et des Images (GRETSI) 2022** under the name **S^{3}LAM: SLAM à Scène Structurée**. [Gonzalez, Marchand, et al., 2021], [Gonzalez et al., 2022a].

- Third, we show that semantic information can allow us to robustly estimate camera pose in dynamic scenes. We tightly track both the camera and all the objects moving in the environment in a single stereo SLAM system. First, we use the static parts of the scene to robustly track the camera. Then, by minimizing the reprojection error of 3D object points we track potentially moving objects. Finally we tightly refine all objects and camera poses along with 3D point positions in a single bundle adjustment. We use the map structure created in S^{3}LAM to constrain the movement of objects to be coherent with the rest of the scene. For example the trajectory of moving cars is optimized with respect to the structure of the road. To do so we define mechanical links between moving objects and the map. Each mechanical link is associated to a projection operator which is applied on dynamic object twists corresponding to the object velocity. The pose of the object is then updated with the projected twist and is thus coherent with the structure of the map. Doing so we can use dynamic object points as shared observations between different camera poses which improves camera pose estimation. We also propose a new bundle adjustment formulation to tightly refine all camera and object poses and all 3D point positions. With this approach we solve the problem of dynamic objects present in S^{3}LAM. However we are still unable to estimate the canonical pose of objects nor to recover the geometry of generic objects. We address this limit in our final contribution.
  This work is published in: **TwistSLAM: Constrained SLAM in Dynamic Environment** accepted at **IEEE RA-L vol. 7 (3) pp. 6846 - 6853 and in IROS 2022.** It is cited in this manuscipt as [Gonzalez et al., 2022b]

- Our final contribution, that we denote **TwistSLAM++** is an improvement over

TwistSLAM. Our goal is to improve the accuracy of object pose estimation, while estimating the canonical pose of each object. To do so we propose to inject LiDAR scans into our dynamic vision SLAM system. We feed scans to a 3D object detector that estimate a set of object 3D bounding boxes. We associate bounding boxes to tracked cluster, allowing us to have access to the canonical pose of each object. Furthermore we register consecutive LiDAR scans with an ICP algorithm to obtain a new estimation of objects transformations. This estimation and the canonical pose estimation from the object detector are fused with our stereo-based tracking in a new bundle adjustment. Finally, using LiDAR object points and a deep neural network we optimize a latent code that represents the shape of the object. Then we constrain object map points to lie on the surface of the estimated geometry in the BA, improving the map. We show on the Kitti dataset that this approach improves the accuracy of object tracking.

- This work was submitted in a conference. It is cited in this manuscript as [Gonzalez et al., 2022c].

Each of those contributions are also accompagnied by an explanatory video that shows additional exemples. The link of each video is given at the beginning of each chapter. Furthermore, a strong focus was put on the state of the art reported in this manuscript, which is why it will be submitted as a journal paper.

Thesis outline

Chapter 1 introduces the fundamentals required to understand this thesis. We begin by presenting the mathematics of 3D motions, used to transform points from world to camera coordinates. Points in camera coordinates can then be projected in images using projective geometry. Following we explain how 2D primitives can be extracted from images and described. Then we introduce the fundamentals of mathematical optimization in non-robust and robust cases. Chapter 1 ends with a wide presentation of semantic information, from object detection to panoptic segmentation.

Chapter 2 shows how to build a SLAM pipeline. It begins with the bundle adjustment and structure from motion. Then camera tracking and mapping are parallelized to obtain a real time solution. Next we add the initialization step followed by the relocalization and loop closure blocks to obtain a classical state of the art SLAM pipeline. Afterwards we present some limits of classical SLAM and finally show how semantic information has

been used in state of the art to overcome those limits. We first show how to build a semantic map. Then we present how semantic information can be used to improve camera relocalization. We study the use of semantic information for dynamic SLAM. Finally we present object based SLAM systems that use objects as high level landmarks.

Chapter 3 presents our object pose estimation network **L6DNet** that can predict the relative pose between the camera and an object in the scene using a single image. We show in a series of experiments that our network produces state-of-the-art results while requiring much less resources than other networks.

Chapter 4 introduces our cluster based SLAM **S^{3}LAM** that builds a map of clusters corresponding to objects in the scene. We show that semantic information can be used to create a high level semantic map. We also show in experiments on public real world datasets that this additional information can be used to obtain a more accurate map which improves camera pose estimation compared to the state-of-the-art.

Chapter 5 details our dynamic SLAM system **TwistSLAM** that can robustly estimate the pose of the camera in dynamic scenes and track all moving objects. The trajectory of dynamic objects is constrained using the map structure. This allows us to improve camera pose estimation in dynamic scenes as well as object pose estimation.

Chapter 6 presents the improvement over chapter 5, with our SLAM system **TwistSLAM++** that injects LiDAR scans into our stereo-based SLAM to improve the accuracy of object tracking and estimate ther canonical pose.

Chapter 7 concludes the thesis by summarizing their contributions and gives perspectives on future directions to our research.

Chapter 1

Fundamentals

Contents

1.1 Introduction

The goal of this chapter is to present fundamental notions that are required to understand this dissertation. The section 1.2 introduces mathematical tools used to describe the 3D transformations and velocities of rigid bodies. Following, section 1.3 describes the mathematics of projective geometry, used to model the formation of a 2D image from a 3D scene. Section 1.4 presents the fundamentals of feature points and descriptors. Then, section 1.5 introduces optimization tools that are classically used in computer vision. Section 1.6 describes robust approaches applied in computer vision to make estimation algorithms less sensitive to noise and errors. Finally, section 1.7 presents what is semantic information as well as how it can be obtained using deep learning based approaches.

1.2 Mathematical foundations of 3D motion

As we are interested in estimating the pose and movement of objects or cameras in our three-dimensional (3D) world we should first understand how to represent the position

and orientation of objects. The goal of this section is to give the mathematical formalism used to represent 3D transformations which are called euclidean transformations. For a more in-depth description of 3D transformations, we refer the reader to textbooks such as: [Blanco, 2010; Y. Ma et al., 2004].

1.2.1 Rigid body motion

The 3D euclidean space $\mathbb{E}^3$ is a vector space associated with a scalar product. It contains points that we will represent using their 3D coordinates $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$. We can write those coordinates as a function of time t, $\mathbf{X}(t)$, allowing the point to move in space. A rigid body can be defined as a set of points which distances are constant through time. To describe the movement of a 3D rigid body we can attach to it an oriented frame $\mathcal{F}_a$. Indeed as the object is rigid, all body points ${}^a\mathbf{X}$ can be represented as a linear combination of the frame vectors with fixed coefficients. Hence to describe the position of each point it is sufficient to describe the location of the frame. This frame can be defined with respect to another frame $\mathcal{F}_b$. We can define the transformation between one frame to another by first rotating its points using a rotation matrix ${}^b\mathbf{R}_a$ and then translating it using a translation vector ${}^b\mathbf{t}_a$:

$$ {}^b\mathbf{X} = {}^b\mathbf{R}_a {}^a\mathbf{X} + {}^b\mathbf{t}_a \tag{1.1} $$

The notation with superscripts and subscripts allows us to easily know on which coordinate frame the transformation is applied. The translation vector is a vector of $\mathbb{R}^3$. The rotation matrix ${}^b\mathbf{R}_a$ is a member of the special orthogonal group SO(3) defined as:

$$ \mathrm{SO}(3) = \{\mathbf{R} \in \mathcal{M}_{3\times 3} \quad | \quad \mathbf{R}^\top\mathbf{R} = \mathbf{I}_3, \quad det(\mathbf{R}) = 1\} \tag{1.2} $$

which corresponds to the set of rotations in three dimensions. Rotation matrices are thus represented using 9 components but have only 3 degrees of freedom due to the orthogonality and direct orientation constraints.

The transformation of a 3D point can be written in a linear way using homogeneous coordinates:

$$ {}^b\bar{\mathbf{X}} = \begin{pmatrix} {}^b\mathbf{X} \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda X \\ \lambda Y \\ \lambda Z \\ \lambda \end{pmatrix} \quad \lambda \in \mathbb{R}_+ \tag{1.3} $$

In homogeneous coordinates, points are embedded in a higher dimensional space in which colinear points are considered equivalent. This allows us to write the euclidean transformation (1.1) using a single homogeneous matrix:

$$ {}^{b}\bar{\mathbf{X}} = {}^{b}\mathbf{T}_{a}\, {}^{a}\bar{\mathbf{X}} \tag{1.4} $$

where the matrix ${}^{b}\mathbf{T}_{a}$ is called the pose matrix of $\mathcal{F}_a$ relative to $\mathcal{F}_b$ and can be written:

$$ {}^{b}\mathbf{T}_{a} = \begin{pmatrix} {}^{b}\mathbf{R}_{a} & {}^{b}\mathbf{t}_{a} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{1.5} $$

The group that contains all euclidean transformations, which correspond to the displacement of rigid objects, is called the special euclidean group. It is denoted SE(3) and can be defined as:

$$ \mathrm{SE}(3) = \{\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \quad | \quad \mathbf{R} \in \mathrm{SO}(3), \mathbf{t} \in \mathbb{R}^3\} \tag{1.6} $$

The product of two matrices of SE(3) is a transformation matrix. It allows us to decompose a transformation between two coordinate frames $\mathcal{F}_a$ and $\mathcal{F}_c$ as the transformation from $\mathcal{F}_a$ to $\mathcal{F}_b$ followed by the transformation from $\mathcal{F}_b$ to $\mathcal{F}_c$:

$$ {}^{c}\mathbf{T}_{a} = {}^{c}\mathbf{T}_{b}\, {}^{b}\mathbf{T}_{a} \tag{1.7} $$

This can represent for example the transformation from an object coordinate frame to a camera coordinate frame by using the world coordinate frame as a bridge between both. A point in $\mathcal{F}_a$ can thus be written in $\mathcal{F}_c$:

$$ {}^{c}\bar{\mathbf{X}} = {}^{c}\mathbf{T}_{a}\, {}^{a}\bar{\mathbf{X}} = {}^{c}\mathbf{T}_{b}\, {}^{b}\mathbf{T}_{a}\, {}^{a}\bar{\mathbf{X}} \tag{1.8} $$

The inverse of a transformation matrix is also a transformation matrix such that:

$$ {}^{b}\mathbf{T}_{a}\, {}^{b}\mathbf{T}_{a}{}^{-1} = {}^{b}\mathbf{T}_{a}{}^{-1}\, {}^{b}\mathbf{T}_{a} = \mathbf{I}_4 \tag{1.9} $$

We will denote ${}^{a}\mathbf{T}_{b} = {}^{b}\mathbf{T}_{a}{}^{-1}$ as it represents the transformation from the second coordinate frame to the first one. Using the fact that rotation matrices are orthonormal, the inverse

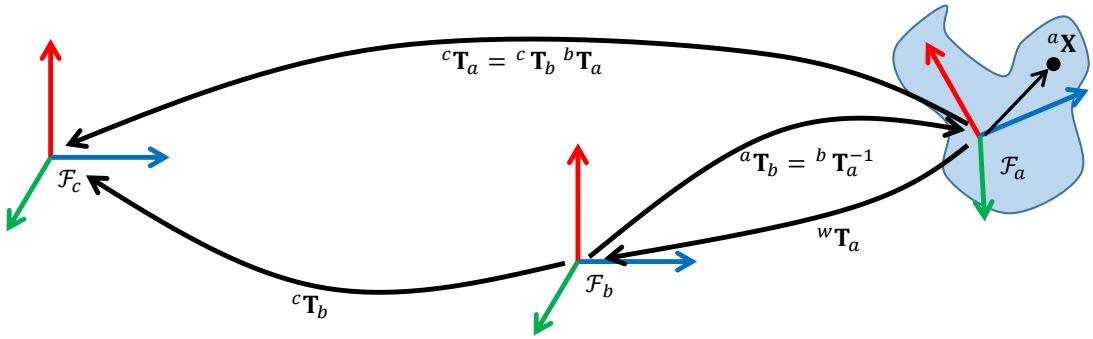

Figure 1.1 – Example of 3D transformations between the coordinate frames $\mathcal{F}_a$, $\mathcal{F}_b$ and $\mathcal{F}_c$.

of a transformation matrix can be simply computed as:

$$ {}^{b}\mathbf{T}_a{}^{-1} = \begin{pmatrix} {}^{b}\mathbf{R}_a{}^{\top} & -{}^{b}\mathbf{R}_a{}^{\top}\,{}^{b}\mathbf{t}_a \\ 0 & 1 \end{pmatrix} \tag{1.10} $$

We illustrate those transformations between the object, the camera and the world coordinate frames in figure 1.1.

1.2.2 Exponential and logarithm map for rotations

In the previous section we introduced rotation matrices to represent rotations in 3D space. In this section we are going to present a minimal representation for the rotational velocity of coordinate frames. We saw in the previous section that the following constraint holds at all times:

$$ {}^{b}\mathbf{R}_c\,{}^{b}\mathbf{R}_c{}^{\top} = \mathbf{I}_3 \tag{1.11} $$

By considering the rotation as a function of time ${}^{b}\mathbf{R}_c(t)$ and derivating the equation with respect to time we obtain:

$$ \frac{\partial {}^{b}\mathbf{R}_c(t)}{\partial t}\,{}^{b}\mathbf{R}_c(t)^{\top} + {}^{b}\mathbf{R}_c(t)\frac{\partial {}^{b}\mathbf{R}_c(t)^{\top}}{\partial t} = \mathbf{0}_{3\times 3} \tag{1.12} $$

This shows that the matrix $\frac{\partial {}^{b}\mathbf{R}_c(t)}{\partial t}\,{}^{b}\mathbf{R}_c(t)^{\top}$ is a skew-symmetric matrix and thus, using a vector expressed in $\mathcal{F}_b$, ${}^{b}\boldsymbol{\omega}(t) \in \mathbb{R}^3$ that represents the rotational velocity of $\mathcal{F}_c$, that it

can be written:

$$\frac{\partial^b \mathbf{R}_c(t)}{\partial t} {}^b\mathbf{R}_c(t)^\top = \begin{pmatrix} 0 & -\omega_z(t) & \omega_y(t) \\ \omega_z(t) & 0 & -\omega_x(t) \\ -\omega_y(t) & \omega_x(t) & 0 \end{pmatrix} = [{}^b\boldsymbol{\omega}(t)]_\times \in \mathfrak{so}(3) \tag{1.13}$$

where $\mathfrak{so}(3)$ is the set of all skew-symmetric matrices of size 3: $\mathfrak{so}(3) = \{\mathbf{A} \in \mathcal{M}_{3\times3} | \mathbf{A} = -\mathbf{A}^\top\}$ and $[.]_\times$ is the skew-symmetric operator which transforms a 3D vector into a skew-symmetric matrix. This equation is a differential equation that can be re-ordered as:

$$\frac{\partial^b \mathbf{R}_c(t)}{\partial t} = [{}^b\boldsymbol{\omega}(t)]_\times {}^b\mathbf{R}_c(t) \tag{1.14}$$

By considering that $\boldsymbol{\omega}(t)$ is constant with respect to time t, this equation can be solved as:

$${}^b\mathbf{R}_c(t) = \exp_{\mathfrak{so}}(t[{}^b\boldsymbol{\omega}]_\times) {}^b\mathbf{R}_c(0) \tag{1.15}$$

Where the exponential of a matrix is given by its Taylor approximation:

$$\exp_{\mathfrak{so}}([{}^b\boldsymbol{\omega}]_\times t) = \sum_{n=0}^{\infty} \frac{(t^b[\boldsymbol{\omega}]_\times)^n}{n!} \tag{1.16}$$

By re-arranging the terms in the Taylor formula we can obtain the Rodrigues' formula:

$$\exp_{\mathfrak{so}}([{}^b\boldsymbol{\omega}]_\times) = \mathbf{I}_3 + \frac{[{}^b\boldsymbol{\omega}]_\times}{||{}^b\boldsymbol{\omega}||} \sin(||{}^b\boldsymbol{\omega}||) + \frac{[{}^b\boldsymbol{\omega}]_\times^2}{||{}^b\boldsymbol{\omega}||^2}(1 - \cos(||{}^b\boldsymbol{\omega}||)) \tag{1.17}$$

This function, called the exponential map, maps elements of the Lie algebra $\mathfrak{so}(3)$ to elements of its corresponding Lie group SO(3). As the representation in $\mathfrak{so}(3)$ is minimal, the Lie algebra can be represented as a hyper plane tangent to the manifold of the Lie group SO(3) that is itself inside the set of 3×3 matrices $\mathcal{M}_{3\times3}$. As illustrated in the following equation, we can use an element in $\mathfrak{so}(3)$ to move around a rotation matrix in SO(3). This is often preferred to directly using a rotation matrix because the rotation matrix has to respect orthogonality constraints. The exponential map of $\mathfrak{so}(3)$ represents the rotation between $\mathcal{F}_c$ and the coordinate frame $\mathcal{F}_{c'}$ obtained by applying the rotational velocity $\boldsymbol{\omega}$ to $\mathcal{F}_c$.

$${}^{c'}\mathbf{R}_b = \exp_{\mathfrak{so}}([{}^b\boldsymbol{\omega}]_\times) {}^c\mathbf{R}_b = {}^{c'}\mathbf{R}_c {}^c\mathbf{R}_b \tag{1.18}$$

The inverse of the exponential map is the logarithm map from SO(3) to $\mathfrak{so}(3)$ and can be defined as:

$$\log_{\mathrm{SO}}({}^{c'}\mathbf{R}_c) = \frac{1}{2\,\mathrm{sinc}(\theta)}({}^{c'}\mathbf{R}_c - {}^{c'}\mathbf{R}_c{}^{\top}) \tag{1.19}$$

where $\theta \in (0, \pi)$ can be computed using $\cos(\theta) = \frac{1}{2}(tr({}^{c'}\mathbf{R}_c) - 1)$ and $\sin(\theta) = \frac{1}{2}\sqrt{(3 - tr({}^{c'}\mathbf{R}_c))(1 + tr({}^{c'}\mathbf{R}_c))}$. The log map yields a skew symmetric matrix which can be transformed into a vector of $\mathbb{R}^3$ or which can directly be obtained using the following formula:

$$\boldsymbol{\omega} = \frac{\theta}{2\sin(\theta)}(\mathbf{R}_{32} - \mathbf{R}_{23}, \mathbf{R}_{13} - \mathbf{R}_{31}, \mathbf{R}_{21} - \mathbf{R}_{12})^{\top} \tag{1.20}$$

where $\mathbf{R} = {}^{c'}\mathbf{R}_c$. For a more in-depth explanation we refer the reader to [Blanco, 2010].

1.2.3 Exponential and logarithm map for poses

Similarly to rotations, we can define a Lie algebra for the space of 3D poses SE(3) that we denote $\mathfrak{se}(3)$:

$$\mathfrak{se}(3) = \left\{ [\boldsymbol{\xi}]^{\vee} = \begin{pmatrix} [\boldsymbol{\omega}]_{\times} & \mathbf{v} \\ \mathbf{0}_{1\times 3} & 0 \end{pmatrix} \quad | \quad \boldsymbol{\omega} \in \mathbb{R}^3, \quad \mathbf{v} \in \mathbb{R}^3 \right\} \tag{1.21}$$

where $\boldsymbol{\omega}$ represents the rotational velocity and $\mathbf{v}$ is the translational velocity of a coordinate frame. The operator $[.]^{\vee}$ maps a vector, called a twist, $\boldsymbol{\xi} \in \mathbb{R}^6$ to a matrix of $\mathfrak{se}(3)$. Its inverse is the operator $[.]^{\wedge}$ such that $[[\boldsymbol{\xi}]^{\vee}]^{\wedge} = \boldsymbol{\xi}$. Hence, the twist $\boldsymbol{\xi} \in \mathbb{R}^6$ is written:

$$\boldsymbol{\xi} = \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{pmatrix} \tag{1.22}$$

Similarly to SO(3) we can define a mapping from $\mathfrak{se}(3)$ to SE(3) that transforms twists into poses:

$${}^{b}\mathbf{T}_{a'} = \exp_{\mathfrak{se}}({}^{b}\boldsymbol{\xi})\,{}^{b}\mathbf{T}_a = {}^{b}\mathbf{T}_a \exp_{\mathfrak{se}}({}^{a}\boldsymbol{\xi}) = {}^{b}\mathbf{T}_a\,{}^{a}\mathbf{T}_{a'} \tag{1.23}$$

where the twist ${}^{b}\boldsymbol{\xi}$ is expressed in the $\mathcal{F}_b$ coordinate frame while ${}^{a}\boldsymbol{\xi}$ is expressed in $\mathcal{F}_a$. The new coordinate frame $\mathcal{F}_{a'}$ corresponds to the displacement of $\mathcal{F}_a$ with rotational and translational velocity corresponding to the twist.

This equation transforms the pose ${}^{b}\mathbf{T}_a$ in the tangent hyperplane $\mathfrak{se}(3)$ as illustrated in figure 1.2. As we can see, the choice of the coordinate frame for the twist depends on

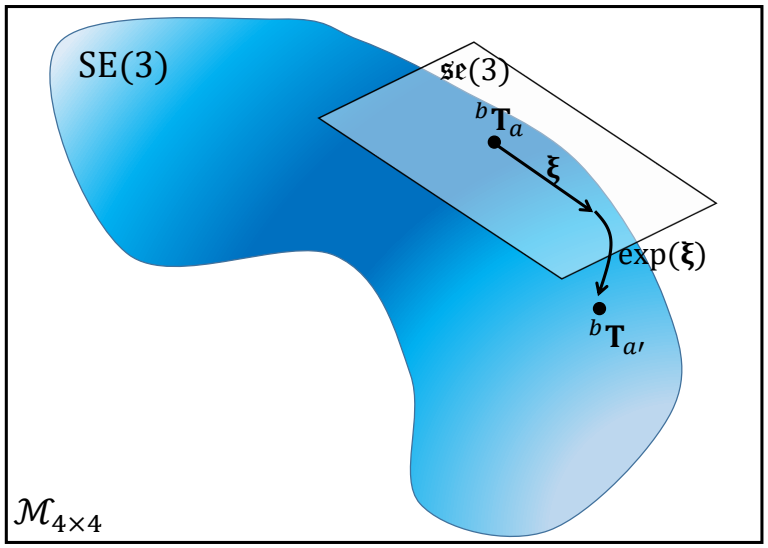

Figure 1.2 – Illustration of the manifold SE(3) embedded in the space of 4×4 matrices with its associated Lie algebra $\mathfrak{se}(3)$.

the order of multiplication. Similarly to points, we can change the coordinate frame in which twists are expressed. To do so it is possible to define an operator called the adjoint map such that:

$$ {}^b\boldsymbol{\xi} = {}^b\mathbf{V}_a \, {}^a\boldsymbol{\xi} \tag{1.24} $$

This operator maps a twist expressed in $\mathcal{F}_a$ to a twist expressed in $\mathcal{F}_b$ and can be computed using the relative pose between both coordinate frames using the following formula:

$$ {}^b\mathbf{V}_a = \begin{pmatrix} 0 & {}^b\mathbf{R}_a \\ {}^b\mathbf{R}_a & [{}^b\mathbf{t}_a]_\times \, {}^b\mathbf{R}_a \end{pmatrix} \tag{1.25} $$

The exponential map from $\mathfrak{se}(3)$ to SE(3) is given by the following formula:

$$ \exp_{\mathfrak{se}}(\boldsymbol{\xi}) = \begin{pmatrix} \exp_{\mathfrak{so}}(\boldsymbol{\omega}) & \mathbf{V}\mathbf{v} \\ 0 & 1 \end{pmatrix} \tag{1.26} $$

where $\mathbf{V} = \mathbf{I}_3 + \frac{1-\cos(\theta)}{\theta^2}[\boldsymbol{\omega}]_\times + \frac{\theta-\sin(\theta)}{\theta^3}[\boldsymbol{\omega}]_\times^2$, with $\theta = ||\boldsymbol{\omega}||$ is the rotation angle induced by $\boldsymbol{\omega}$. The logarithm map from SE(3) to $\mathfrak{se}(3)$ can be computed as:

$$ \log_{\mathrm{SE}}(\mathbf{T}) = \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \frac{\theta}{2\sin(\theta)}(\mathbf{R}_{32} - \mathbf{R}_{23}, \mathbf{R}_{13} - \mathbf{R}_{31}, \mathbf{R}_{21} - \mathbf{R}_{12})^\top \\ \mathbf{V}^{-1}\mathbf{t} \end{pmatrix} \tag{1.27} $$

where $\mathbf{t}$ is the translational part of $\mathbf{T}$.

We can also define a pseudo-exponential map and a pseudo-logarithm map [Blanco, 2010] that are simpler to compute by ignoring the rotation impact on the translation. The main advantage of those transformations is that translations and rotations are separated, which leads to simpler derivatives. Those functions can be defined as:

$$\exp_{\mathfrak{se}}(\boldsymbol{\xi}) = \begin{pmatrix} \exp_{\mathfrak{so}}(\boldsymbol{\omega}) & \mathbf{v} \\ 0 & 1 \end{pmatrix} \tag{1.28}$$

$$\log_{\mathrm{SE}}(\mathbf{T}) = \begin{pmatrix} \frac{\theta}{2\sin(\theta)}(\mathbf{R}_{32} - \mathbf{R}_{23}, \mathbf{R}_{13} - \mathbf{R}_{31}, \mathbf{R}_{21} - \mathbf{R}_{12})^\top \\ \mathbf{t} \end{pmatrix} \tag{1.29}$$

For those functions, as the translational part does not depend on the rotational part, their Jacobians contain additional 0 entries in place of the derivative of the matrix $\mathbf{V}$.
We saw in this section how to transform points from one coordinate frame to another, for example from the object coordinate frame to the camera coordinate frame by using the world coordinate frame as a bridge and how to compute the rotational and translational velocities of coordinate frames. We will explain in the next section how to project those 3D points into 2D image points once they expressed in the camera coordinate frame.

1.3 Projective geometry

Most cameras are composed of an optical system that concentrates the light incoming from the scene onto a photosensitive sensor that will measure and save it. The goal of this section is to present projection models that simulate the way a camera works by transforming 3D points into 2D points. We define two projection models: first for monocular cameras and then for stereo cameras. For a more in-depth description of projective geometry, the reader can refer to textbooks such as: [Forsyth and Ponce, 2011; R. Hartley and Zisserman, 2003; Y. Ma et al., 2004; Szeliski, 2010].

1.3.1 Monocular projection models

There are multiple ways to model a camera, depending among other things on the type of camera used. The most simple one is called the pinhole projection model. This model corresponds to a simplification of a camera for which the lens diameter is zero. For this model, illustrated in figure 1.3, we need to define two entities: the camera center $\mathbf{C}$

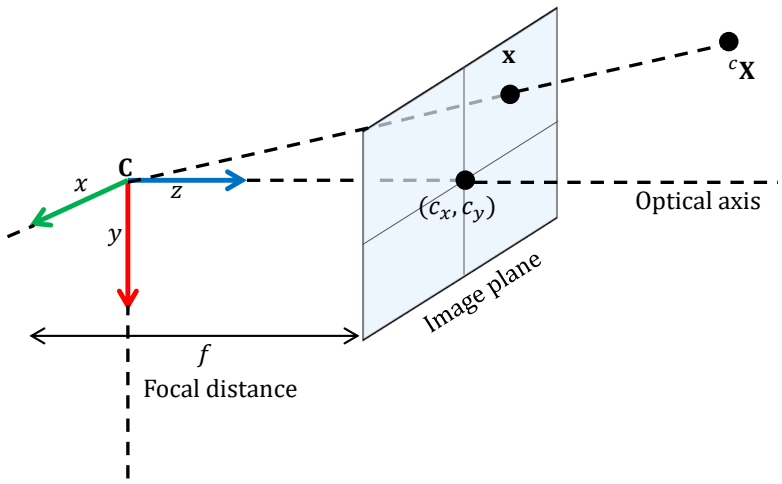

Figure 1.3 – The pin-hole camera model.

and the image plane, located at a distance f of the center. f is the focal length of the camera.

With this model every 3D point is mapped to the image plane by computing the intersection between the plane and a ray connecting the 3D point and the camera center. The projection function of a point ${}^c\mathbf{X} = (X, Y, Z)^\top$ can be computed analytically using Thales theorem:

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{fX}{Z} \\ \frac{fX}{Z} \end{pmatrix} \tag{1.30}$$

Points are then sampled to pixels and expressed relatively to the image top left, the projection function that we denote $\pi({}^c\mathbf{X})$ thus becomes:

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \pi({}^c\mathbf{X}) = \begin{pmatrix} \frac{f_x X}{Z} + c_x \\ \frac{f_y X}{Z} + c_y \end{pmatrix} \tag{1.31}$$

where the parameters (c_x, c_y) are the coordinates of the camera principal point in pixels. f_x and f_y can be computed as:

$$f_x = \frac{f}{l_x} \qquad f_y = \frac{f}{l_y} \tag{1.32}$$

where (l_x, l_y) is the size of a pixel on the sensor. Those parameters are often stocked in

the camera intrinsics matrix $\mathbf{K}$ defined as:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{1.33}$$

Using this matrix, a projection matrix and homogeneous coordinates we can write:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K\Pi} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1.34}$$

where the projection matrix $\mathbf{\Pi}$ is defined by:

$$\mathbf{\Pi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.35}$$

The pin-hole projection model thus depends on 4 parameters. However there exist more complex projection models that can take into account deformations in the image. Indeed when using cameras with thick lenses straight lines in 3D space are curved in image space. This is the case for example for fish eye cameras that have a large field of view. To take this effect into account, one can either model the distortion to undistord the images or use a more complex projection model, such as the unified model [Geyer and Daniilidis, 2000].

The projection function can be inverted to triangulate 3D points from 2D pixels if their depth is known. This function is denoted:

$${}^c\mathbf{X} = \pi^{-1}(\mathbf{x}, Z) = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} (x - c_x)\frac{Z}{f_x} \\ (y - c_y)\frac{Z}{f_y} \\ Z \end{pmatrix} \tag{1.36}$$

To summarize, a 3D point in the world coordinate frame can be projected in the image space by first transforming it to the camera coordinate frame and by using the projection

function to obtain its coordinates in pixels.

$$\bar{\mathbf{x}} = \mathbf{K}\mathbf{\Pi}^{\mathbf{c}}\mathbf{T}_{\mathbf{a}}{}^{\mathbf{a}}\bar{\mathbf{X}} \tag{1.37}$$

We denote this full transformation without homogeneous coordinates as:

$$\mathbf{x} = p({}^{c}\mathbf{T}_{a}, {}^{a}\mathbf{X}) \tag{1.38}$$

1.3.2 Stereo projection models

This transformation corresponds to monocular cameras but can also be adapted to stereo cameras. A stereo camera is composed of a pair of monocular cameras with coordinate frames $\mathcal{F}_{c_1}$ and $\mathcal{F}_{c_2}$ with relative pose ${}^{c_1}\mathbf{T}_{c_2}$. This setting allows to estimate the depth of the scene by triangulating pixels matched in both frames. Stereo images can be rectified by projecting them on a plane to make the epipolar lines horizontal. To do so, an homography is computed for each camera, so that the pose between transformed cameras c_1' and c_2' becomes:

$${}^{c_1'}\mathbf{T}_{c_2'} = \begin{pmatrix} 1 & 0 & 0 & b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.39}$$

where b is the baseline between the transformed cameras [1]. This eases the matching process as corresponding keypoints lie on the same horizontal line. In the case of a rectified stereo camera with a baseline b the projection becomes:

$$\begin{pmatrix} x_l \\ y_l \\ x_r \end{pmatrix} = \begin{pmatrix} \frac{f_x X}{Z} + c_x \\ \frac{f_x X}{Z} + c_x \\ \frac{f_x X - f_x b}{Z} + c_x \end{pmatrix} \tag{1.40}$$

where x_l, y_l are the coordinates of the point in the left camera and x_r is the horizontal coordinate in the right camera. The vertical coordinate is not computed as it is equal to y_l thanks to rectification.

Similarly to the monocular case we denote the reprojection function $\pi({}^{c}\mathbf{X})$. The func-

1. This model is the one used in ORB-SLAM2. It can always be applied to a stereo camera after rectifying the images.

tion can be inverted:

$$\pi(\mathbf{x}) = \begin{pmatrix} b\frac{x_l - c_x}{x_l - x_r} \\ b\frac{f_x}{f_y}\frac{y_l - c_y}{x_l - x_r} \\ bf_x\frac{1}{x_l - x_r} \end{pmatrix} \tag{1.41}$$

We can also define a function that transforms point coordinates to camera coordinates and then project them as we did in the monocular case. We also denote this function:

$$\mathbf{x} = p({}^c\mathbf{T}_a, {}^a\mathbf{X}) \tag{1.42}$$

In the following section we will present feature points, that are actual measures of the position of 2D points in images.

1.4 Image primitives

In this section we present how to extract and describe 2D primitives in images that correspond to points in 3D space.

1.4.1 Feature points

Pixels in an image can not be described using solely their own value. Indeed, as it takes only integer values between 0 and 255, it is shared by many other pixels and is thus not discriminative. Hence when analyzing a point in an image we will take into account a small window around that point. Doing so we give a more specific signature to the pixel.

Different points in an image do not all bring the same amount of information. For example multiple points on a white wall are very similar, same goes for points along a line. On the contrary points on highly textured area are very specific. It can be shown that to correctly localize a point, its local gradient should be strong in 2 different directions. Those points are called corner points or keypoints and multiple algorithms have been designed to extract them in an image. Among those algorithms are the Harris corner detector [Harris, Stephens, et al., 1988] that computes the local curvature of the image, SIFT [Lowe, 1999], SURF [Bay et al., 2006] that compute respectively a difference and a laplacian of gaussians at multiple scales.

However those approaches are computationally expensive, which is why the FAST keypoint extractor [Rosten and Drummond, 2006] was developed. This algorithm is straightforward: for all pixels in the image verify if an arc around the pixel is significantly lighter

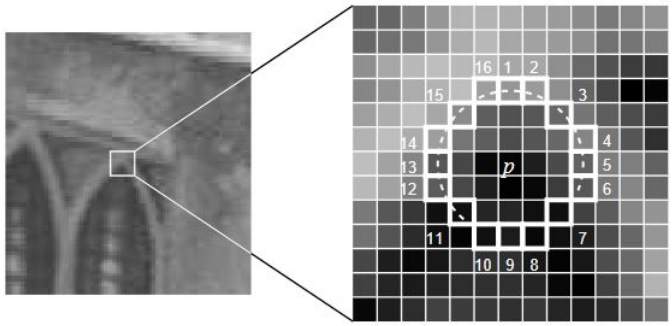

Figure 1.4 – Example of FAST test on a pixel, image from [Rosten and Drummond, 2006]

or darker than the central pixel. If it is the case, the pixel corresponds to a keypoint. This can create areas with a high density of points, which is why Non Maximum Suppression (NMS) can be used to keep only the keypoints with the highest scores. An example of a FAST test on a pixel is shown figure 1.4. ORB [Rublee et al., 2011] proposes to modify FAST keypoints by computing their local orientation to make them invariant to rotations. To make the keypoints scale invariant, they are also computed at different levels in an image pyramid which contains down sampled versions of the original image.

1.4.2 Points descriptors

Once keypoints have been extracted from an image they should be associated with a vector that describes its surrounding visual information and acts like a signature. Such vector is called a descriptor and it would ideally be unique and invariant with respect to changes such as viewpoint or illumination. Such strong properties are required so that two descriptors depicting the same 3D point seen in different images are uniquely matched together. In reality descriptors are not invariant and are prone to be falsely paired, which is why they should be handled by robust approaches.

The most straightforward descriptor is a vectorized version of the image patch around the keypoint. In that case the difference between two descriptors can be obtained using the well known normalized cross-correlation. To create an invariant descriptor, SIFT proposed to compute a histogram of gradients for each of the 16 areas around the keypoint. Concatenating the histograms yields a vector of size 128 which represents the texture around the keypoint. SIFT descriptors can then be matched using the classical euclidean distance between vectors. The SURF descriptor is similar and proposes to compute for

each area its response to the convolution with the Haar wavelet.

The BRIEF [Calonder et al., 2010] descriptor has been designed to be extracted and matched rapidly. It is represented using a binary vector of size 256 obtained by comparing pairs of pixels in the patch. Descriptors can then be compared using the Hamming distance which can be computed efficiently. The BRIEF descriptor has been improved by [Rublee et al., 2011] which proposes to compute the local orientation of the patch and build a steered version of the BRIEF descriptor to make it rotational invariant.

More recently, with the rise of deep learning based methods [K. He et al., 2016; Krizhevsky et al., 2012; Simonyan and Zisserman, 2014] some approaches try to train Convolutional Neural Networks (CNN) that can estimate a set of keypoint locations with associated descriptors. Their goal is to harness the invariance of networks when trained with large amounts of data to build descriptors and keypoints that are consistent across views. Superpoint [DeTone et al., 2018] proposes a self supervised approach that predicts both keypoints locations and descriptors. To do so they train a network on a simple synthetic dataset. Then, they use this network to predict keypoints locations for real warped images that are then aggregated to obtain stable keypoint locations which are used to refine the base network. Doing so they obtain obtain comparable results to SIFT [Lowe, 1999]. Such approaches can prove to be beneficial for computer vision problems such as SLAM. For example [Tang et al., 2018; 2019] learn a binary descriptor to be consistent across views. Other approaches for deep learnt detectors can be found in [J. Ma et al., 2021].

1.5 Optimization

The goal of this subsection is to present optimization methods classically used in computer vision. Optimization is a field of applied mathematics that seeks to find the parameters that minimize a cost function. In computer vision we often need to estimate parameters given measurements. One way to do so is by using a measurement model that can generate idealized measurements given parameters. By comparing the generated measurements and the true ones, in what is called a cost function, and minimizing this function, we can estimate the parameters that yielded the observations. We will cover two types of minimization techniques: linear and non linear least-squares. For a more in depth coverage of optimization techniques we refer the reader to: [Nocedal and Wright, 1999].

1.5.1 Linear least-squares minimization

Given a set of n measurements $\mathbf{b} \in \mathbb{R}^n$ and a measurement model g, such as the pin-hole camera model, the goal of least-squares minimization is to find the parameters $\boldsymbol{\beta} \in \mathbb{R}^m$ of g that minimize the following cost function:

$$E(\boldsymbol{\beta}) = \frac{1}{2}||g(\boldsymbol{\beta}) - \mathbf{b}||^2 = \frac{1}{2}\sum_{i=0}^{n}||g(\boldsymbol{\beta})_i - b_i||^2 = \frac{1}{2}\sum_{i=0}^{n}||r_i(\boldsymbol{\beta})||^2 \tag{1.43}$$

where the r_i are called the residuals. By optimizing this function we seek to find the parameters of the function that yield the closest values from the measurements. When the function g is linear with respect to the parameters we can write the cost function as:

$$E(\boldsymbol{\beta}) = \frac{1}{2}||\mathbf{A}\boldsymbol{\beta} - \mathbf{b}||^2 \tag{1.44}$$

As this is a strictly convex function, the minimum can be found by canceling its gradient, using the normal equations:

$$\nabla E(\boldsymbol{\beta}) = 0 \Leftrightarrow \mathbf{A}^\top(\mathbf{A}\boldsymbol{\beta} - \mathbf{b}) = 0 \tag{1.45}$$

$$\Leftrightarrow \mathbf{A}^\top\mathbf{A}\boldsymbol{\beta} = \mathbf{A}^\top\mathbf{b} \tag{1.46}$$

$$\Leftrightarrow \boldsymbol{\beta} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b} \tag{1.47}$$

$$\Leftrightarrow \boldsymbol{\beta} = \mathbf{A}^+\mathbf{b} \tag{1.48}$$

In practice the normal equations are not solved by inverting the matrix $\mathbf{A}^\top\mathbf{A}$ as it can be expensive to compute with a complexity of $O(n^3)$. Furthermore if $\mathbf{A}$ is ill conditioned, making the inversion process highly sensitive to small errors. Hence the normal equations are rather solved by computing a factorization of $\mathbf{A}$ such as the Cholesky, QR decomposition or directly from $\mathbf{A}$ using the SVD.

1.5.2 Non linear least-squares minimization

When the function g is non linear, the optimization is more complex as it can not be solved in a single operation. A way to solve it is iteratively by linearizing the cost function at the current estimate, solving the normal equations to find a better local minimum and using that local solution to repeat the operation until it converges.

The first order Taylor expansion of g is:

$$g(\boldsymbol{\beta} + \boldsymbol{\delta}) \simeq g(\boldsymbol{\beta}) + \mathbf{J}\boldsymbol{\delta} \tag{1.49}$$

where $\mathbf{J}$ is the Jacobian matrix of g defined by:

$$\mathbf{J}_{i,j} = \frac{\partial g(\boldsymbol{\beta})_i}{\partial \boldsymbol{\beta}_j} \tag{1.50}$$

Using this approximation we can find the increment $\hat{\boldsymbol{\delta}}$ which minimizes:

$$\hat{\boldsymbol{\delta}} = \arg\min_{\boldsymbol{\delta}} \frac{1}{2}||g(\boldsymbol{\beta}) + \mathbf{J}\boldsymbol{\delta} - \mathbf{b}||^2 \tag{1.51}$$

This function is linear in $\boldsymbol{\delta}$, its minimum can thus be computed using linear least squares results. We thus have:

$$\hat{\boldsymbol{\delta}} = -(\mathbf{J}^\top \mathbf{J})^{-1} \mathbf{J}^\top r(\boldsymbol{\beta}) \tag{1.52}$$

where $r(\boldsymbol{\beta}) = g(\boldsymbol{\beta}) - \mathbf{b}$ corresponds to the residuals. The optimal increment is then used to update the current estimate to obtain a new one: $\boldsymbol{\beta}' = \boldsymbol{\beta} + \hat{\boldsymbol{\delta}}$. Linearization is then performed at the new estimate to find a new optimal increment. This process is repeated until convergence. This algorithm is called the Gauss-Newton algorithm and it can be summarized as follows:

Gauss-Newton algorithm

Starting with a good solution $\boldsymbol{\beta}^{(0)}$, iterate:

1. Compute the Jacobian $\mathbf{J}$ and residuals at $\boldsymbol{\beta}^{(s)}$
2. Compute the optimal increment: $\hat{\boldsymbol{\delta}} = -(\mathbf{J}^\top \mathbf{J})^{-1} \mathbf{J}^\top r(\boldsymbol{\beta}^{(s)})$
3. Update the current estimate $\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} + \hat{\boldsymbol{\delta}}$

Iterations can be stopped when exceeding a threshold or when $||\boldsymbol{\beta}^{(s+1)} - \boldsymbol{\beta}^{(s)}|| \leq \epsilon$

The Gauss-Newton algorithm can also be obtained by canceling the derivative of the cost function using the iterative Newton approach. However this approach requires to compute the Hessian of the cost function, which can be approximated by $\mathbf{J}^\top \mathbf{J}$ by ignoring second order derivatives.

We show in figure 1.5 an illustration of the Gauss-Newton algorithm: the function is linearized around $\boldsymbol{\beta}^{(s)}$ and a linear least squares system is solved. This amounts to fitting

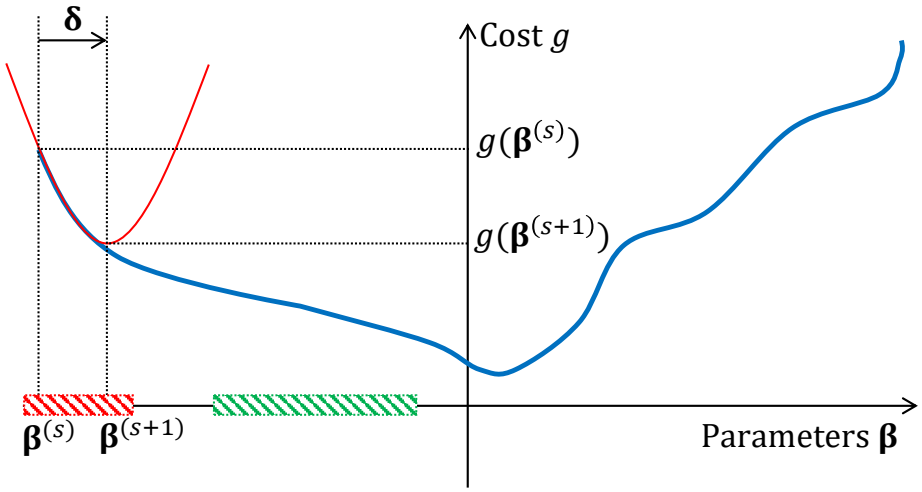

Figure 1.5 – Example of a Gauss-Newton iteration.

and minimizing a local parabola. As we can see this approximation works well when the function is locally quadratic (in the red hatched area). However when the function is locally linear (in the green hatched area) the convergence is very slow. A solution to this problem would be to use gradient descend, which is a first order minimization approach. Both the Gauss-Newton algorithm and gradient descent can thus be merged to obtain the Levenberg-Marquardt algorithm by changing the computation of the increment:

$$\hat{\boldsymbol{\delta}} = -(\mathbf{J}^\top \mathbf{J} + \lambda \mathbf{I}_n)^{-1} \mathbf{J}^\top r(\boldsymbol{\beta}) \tag{1.53}$$

where $\lambda \geq 0$ is used to balance both algorithms. When $\lambda = 0$ it corresponds to the classical Gauss-Newton step, while when $\lambda \longrightarrow \infty$ the left part of the update step can be neglected which corresponds to a gradient descent step. There are several ways to set the λ hyperparameter, the general idea is to decrease it when finding better solutions and increase it otherwise. This modification of the Gauss-Newton algorithm can also be related to the well known Tikhonov regularization. Its goal is to take the matrix $\mathbf{J}^\top \mathbf{J}$ away from the set of singular matrices, i.e. matrices with null eigen values, in order to improve its conditioning.

1.5.3 Non linear least-squares minimization on a manifold

A cost function can be defined and optimized on a manifold, other than $\mathbb{R}^n$. For example, when estimating the pose of a camera we optimize over SE(3). To do so we can not directly solve the normal equations introduced in the previous part. Indeed, the increment is not constrained and we would have no guarantee that the updated estimate would lie on the manifold. A solution to bypass this problem is thus to optimize the increment in the unconstrained space of the Lie algebra and then use the exponential map to update the current estimate on the manifold:

Gauss-Newton algorithm on the Lie group SE(3)
Starting with a good solution $\boldsymbol{\beta}^{(0)} \in \text{SE}(3)$, iterate:

1. Compute the Jacobian $\mathbf{J}$ and residuals at $\boldsymbol{\beta}^{(s)}$
2. Compute the optimal increment: $\hat{\boldsymbol{\delta}} = -(\mathbf{J}^\top \mathbf{J})^{-1} \mathbf{J}^\top r(\boldsymbol{\beta}^{(s)})$
3. Update the current estimate $\boldsymbol{\beta}^{(s+1)} = \exp(\hat{\boldsymbol{\delta}})\boldsymbol{\beta}^{(s)}$

Iterations can be stopped when exceeding a threshold or when $||\boldsymbol{\beta}^{(s+1)} - \boldsymbol{\beta}^{(s)}|| \leq \epsilon$

This however requires a slight modification of the Jacobian:

$$\mathbf{J}_{i,j} = \left. \frac{\partial g(\exp(\boldsymbol{\delta})\boldsymbol{\beta})_i}{\partial \boldsymbol{\delta}_j} \right|_{\boldsymbol{\delta}=0} \tag{1.54}$$

More information about optimization on manifolds can be found in [C. Hertzberger, 2008; Grisetti et al., 2011]

1.6 Robust estimations

The data on which optimization algorithms are applied comes either directly from measurements or from other algorithms that processed measurements. Among those measurements there can be spurious information due to sensors malfunction or algorithmic errors, such as wrong keypoints matches for example. Those measurements should be dealt with by either discarding or down-weighting them. The goal of this section is to present methods that can do so automatically. We will first present the RANSAC algorithm and then robust M-estimators. For a more in depth coverage of robust estimation techniques we refer the reader to: [Huber, 2011; Malis and Marchand, 2006].

1.6.1 RANSAC

The goal of the algorithm RANSAC (RANdom SAmple Consensus) [Fischler and Bolles, 1981] is to estimate the parameters of a model using data that contains gross outliers. It is often used in computer vision to filter out wrong matches between feature points. This algorithm works in an iterative fashion: first a minimal set of points is randomly selected and used to fit the model then the model is used to verify which remaining points are inliers or outliers. We then repeat this process to find the model with the biggest set of inlier points, which is then used to fit the final model.

The main idea behind this algorithm is that the outliers come from a random noise and should not be supported by any model. This algorithm has a hard threshold with respect to the outliers, meaning that the outliers are not down weighted but rather completely discarded. This approach can be computationally expensive as the number of iterations required to fit the correct model can be large, depending on the model degrees of freedom and the ratio of outliers.

RANSAC has known many improvements to make it more robust, increase its speed, remove the need for a hard threshold, for example with GC-RANSAC, MAGSAC and MAGSAC++ [Barath and Matas, 2018; Barath et al., 2019; 2020].

1.6.2 Robust M-estimators

An M-estimator corresponds to the parameter that is obtained when optimizing a cost function. In the case of a least-square cost function the potential outliers that do not fit well the model will have a high impact on the cost. Thus, even if they are few they may bias the result of the optimization. To deal with this problem, a robust cost function can be designed by down weighting the contribution of outliers. A well known example of such cost functions is the Huber cost function:

$$\rho(x) = \begin{cases} \frac{1}{2}x^2 & \text{if} \quad |x| \leq c \\ c(|x| - \frac{c}{2}) & \text{else.} \end{cases} \tag{1.55}$$

where $c = 1.345\hat{\sigma}$ and $\hat{\sigma}$ is a robust estimator of the standard deviation of the residuals which is obtained using the Median Absolute Deviation (MAD):

$$\hat{\sigma} = 1.48\text{MAD} = 1.48\text{median}(|x - \text{median}(x)|) \tag{1.56}$$

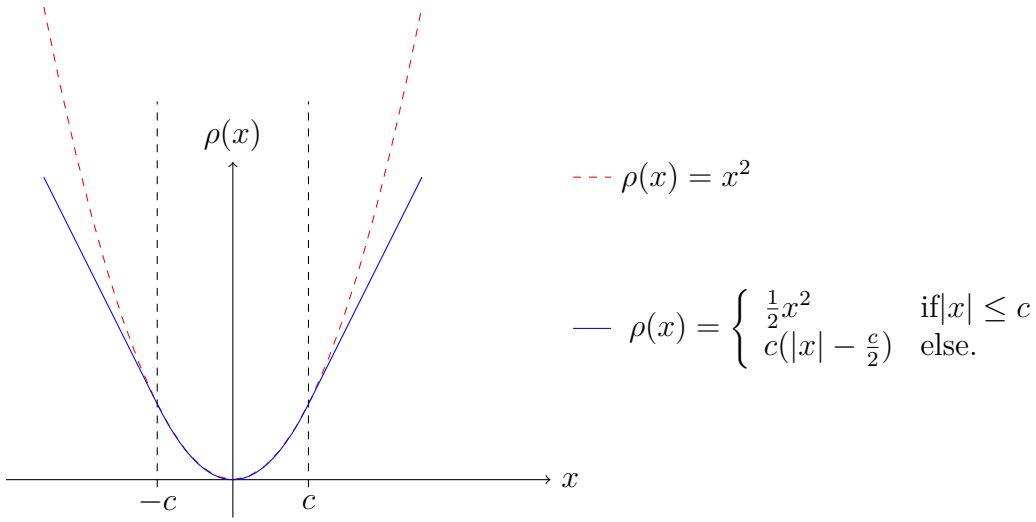

Figure 1.6 – The Huber cost function (blue) compared to the quadratic cost function (red)

As we can see in figure 1.6 the function is quadratic on a limited part of its domain and linear everywhere else. This function can be applied to residuals such as in the following equation:

$$E(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=0}^{n} \rho(||r_i(\boldsymbol{\beta})||) \tag{1.57}$$

In that case residuals larger than c, that are most likely outliers are treated linearly and thus have a lesser impact on the cost function than if they were treated quadratically. Contrary to RANSAC, this robust cost function does not remove outliers but rather takes them into account with a lower weight. To take into account this function in the optimization process, the update step in the normal equation must be modified with a diagonal weight matrix $\mathbf{D}$ computed from $\rho(||r(\boldsymbol{\beta})||)$:

$$\hat{\delta} = -(\mathbf{DJ})^{+}\mathbf{D}r(\boldsymbol{\beta}) \tag{1.58}$$

1.7 Semantic information

The term *semantic* comes from the ancient greek *semanticos* which designated something that carries meaning. In our case we call semantic information the knowledge of the class or category of objects, such as *laptop, road, car, table, etc.*. Semantic information in images was for some times a really challenging information to obtain. However with

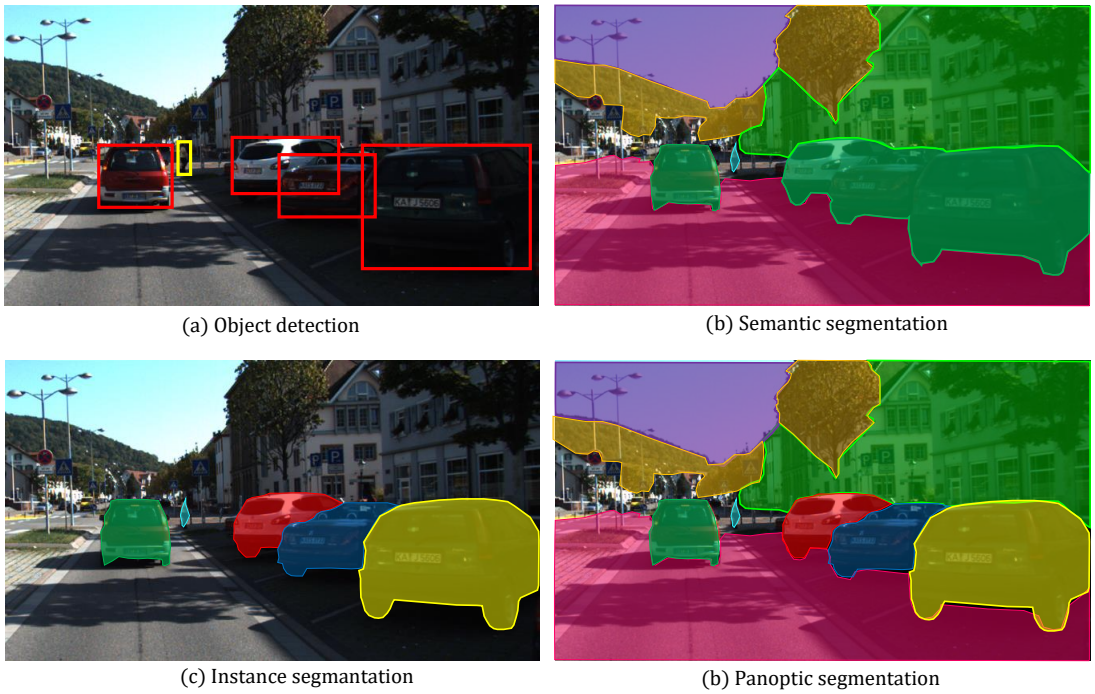

Figure 1.7 – Examples of (a) object detection, (b) semantic segmentation, (c) instance segmentation and (d) panoptic segmentation on an image from the KITTI dataset.

the rise of deep learning after AlexNet [Krizhevsky et al., 2012] performance in the ImageNet [Russakovsky et al., 2015] dataset, deep Convolutional Neural Networks (CNNs) have been successfully trained to answer the problem of semantic information estimation.

In this section we will introduce the fundamentals of deep learning followed by four categories of subproblems which goal is to obtain semantic information: object detection, instance segmentation, semantic segmentation and panoptic segmentation that are illustrated in figure 1.7. The goal here is not to do an in depth survey of deep learning based semantic approaches but rather to give the reader an idea of what semantic information is and how we can obtain it. In-depth surveys can be found in [Jiao et al., 2019; W. Liu et al., 2019; Minaee et al., 2021; Zaidi et al., 2022]. It should be noted that we focus here on segmentation applied on a single image, however the problem is modified when considering videos input as there is a temporal continuity.

1.7.1 Deep learning fundamentals

The idea behind deep learning, which was popularized by [Krizhevsky et al., 2012], is to stack simple computational layers to approximate more complex functions. Each layer contains a set of parameters that can be modified by the network to minimize a loss function during an expensive training step. Once the training is done the network should be able to generalize on new data samples.

Convolutional Neural Networks (CNN) are one of the most popular type of deep networks. They are usually made of two parts: the convolutional layers and the fully connected layers. Convolutional layers contain filters that are applied to images to extract specific features. Those layers are stacked and each one extracts features from data that was treated by the previous layer. Convolutional layers can be associated with pooling layers which reduce the amount of data, making the approach more computationally efficient.

Each fully connected layer is a matrix multiplication with the input data, followed by a non linear function (generally the ReLU function). Those layers usually take as input the features extracted by the convolutional layers and transform them to match the variable that we seek to predict.

There has been a lot of work to find efficient neural architectures, that can be as deep as possible without requiring too many parameters. Most popular ones include VGG [Simonyan and Zisserman, 2014], Inception [Szegedy et al., 2015], ResNet [K. He et al., 2016], ResNeXt [Xie et al., 2017] and EfficientNet [Tan and Le, 2019]. More recently transformers architectures [Vaswani et al., 2017] have been adapted for image processing [Dosovitskiy et al., 2020; Z. Liu et al., 2021], yielding impressive results for many vision tasks. However a recent approach [Z. Liu et al., 2022] seems to obtain better results with a well designed convolutional approach.

1.7.2 Object detection

The goal of object detection is, given an RGB image, to estimate the location and class of all objects visible in the image. The location of the object is represented using a bounding box, that is a 2D box that tightly encapsulates the object. Thus, networks that solve object detection predict for a single image a set of 2D box coordinates with object classes.

The first approach that solved object detection using deep learning was R-CNN [Gir-

shick et al., 2015]. They proposed to detect regions of interest in images using selective search, extract image features using a CNN from those regions and use an SVM and linear regression trained on the features to estimate the class of the region and refine the region location. This approach was accurate but slow and was quickly improved by Fast R-CNN [Girshick, 2015] and Faster R-CNN [Ren et al., 2015] who built upon it. Fast R-CNN speeds up inference time by extracting image features at once and then by pooling the region of interest within the feature maps. Faster R-CNN goes beyond this approach by using a region proposal network which also uses the extracted features maps to replace selective search.

These approaches are called region proposal based techniques as they first need to get regions of interest and then apply a classifier on the regions to get the object class. [Redmon et al., 2016] proposes a new approach called YOLO by predicting at once all the bounding boxes and classes using a single CNN. By reducing the number of steps in the pipeline they obtain state of the art results in real-time. SSD [W. Liu et al., 2016] goes further by predicting multiple boxes per object at different scales to have better fitting boxes. However those approaches are limited by the large amount of negative or background classes that pollute the loss function, compared to the few positive examples. To solve this problem RetinaNet [Lin et al., 2017] modifies the loss function to give more weight to hard positive examples. This allows them to predict a much higher number of boxes without risking class imbalance. More recently DETR [Carion et al., 2020] use transformers for object detection. They propose to simplify object detection by directly predicting a single box per object instead of having multiple anchors or proposals that then need to be filtered. To do so they extract features from images using a CNN. The features are then handled by a transformer encoder and decoder to finally predict bounding boxes.

1.7.3 Semantic segmentation

The goal of semantic segmentation is to predict an object class for each pixel in the image. This means that the prediction is dense as it is pixelwise. This also means that if there are multiple objects with the same category in an image they will all be segmented the same way. Fully Convolutional Networks [Long et al., 2015] is one of the first work that segment images using deep learning. To do so they propose to remove the fully connected layer in classification architectures to obtain a 2D heatmap per class. The heatmap is then upsampled to match the original image resolution. To improve the sharpness of the segmentation they fuse heatmaps obtained at different level in the network.

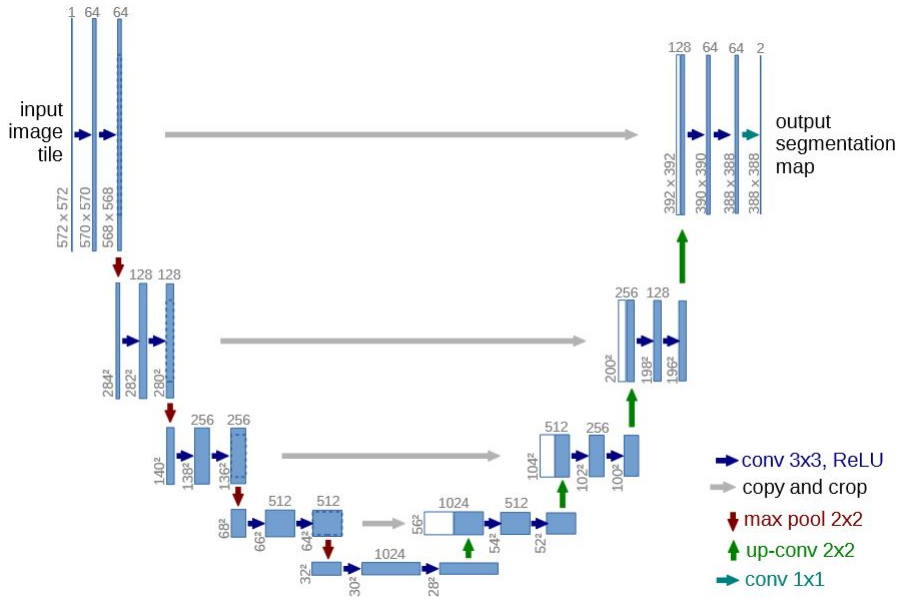

Figure 1.8 – The U-Net [Ronneberger et al., 2015] architecture. Features are extracted at multiple scales and fused to aggregate information at different levels. Image from [Ronneberger et al., 2015].

U-Net [Ronneberger et al., 2015] proposes a symmetric architecture, shaped like a "U" which extracts image features at different resolutions and fuse them all, improving the base idea of [Long et al., 2015]. We show their architecture in figure 1.8.

DeepLab [L.-C. Chen, Papandreou, Kokkinos, et al., 2017] uses atrous convolutions which correspond to large filters containing zeros at regular intervals. Those convolutions allow them to keep an image of a reasonable resolution while having a larger receptive field, and thus aggregating more context information. Those convolutions are applied in parallel with multiple filter sizes in the atrous spatial pyramid pooling layer, which aggregates information at different scales. This layer produces a heatmap which is then upsampled and refined using a conditional random field, allowing to obtain more precise object boundaries. This architecture has been improved in its following versions [L.-C. Chen, Papandreou, Schroff, and Adam, 2017; L.-C. Chen et al., 2018]

1.7.4 Instance segmentation

Instance segmentation relates to both object detection and semantic segmentation. Its goal is to segment all objects in the scene while assigning a unique id to each object.

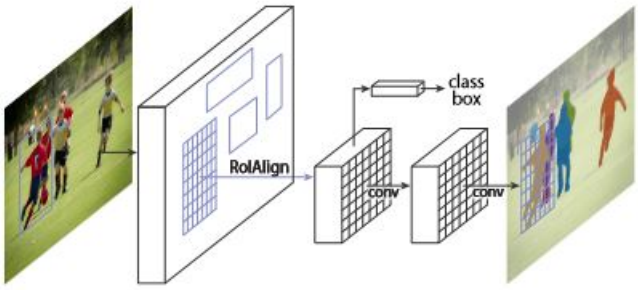

Figure 1.9 – Architecture of Mask-RCNN from [K. He et al., 2017]

Note however that instance segmentation only segments countable objects (called *things*) contrary to semantic segmentation that also segment uncountable objects (called *stuff*). Note also that networks trained for instance segmentation also output object detection, usually with a better precision than networks that tackle only object detection. The most famous instance segmentation network is Mask R-CNN [K. He et al., 2017]. This system is built upon Faster R-CNN [Ren et al., 2015] and proposes to segment an object in each region found by the region proposal network. To do so they apply a fully convolutional network on the feature maps aligned with the region of interest. This network predicts masks on each detected bounding box. Its architecture is shown in figure 1.9.

This approach runs at less than 10 fps and is thus not real time. YOLACT [Bolya et al., 2019] proposes an architecture for instance segmentation at more than 30 fps with an accuracy close to state of the art. To do so they transform the 2 stages strategy of Mask-RCNN by predicting for a single image a set of prototype masks and a set of coefficients. The segmentation of the image is given by the linear combination of masks with coefficients. This approach is then improved in YOLACT++ [Bolya et al., 2020].

1.7.5 Panoptic segmentation

Panoptic segmentation tries to solve both instance and semantic segmentation. Its goal is to assign a class to each pixel in the image while separating different instances of objects from a same class. The simplest way to solve this problem is to use both a semantic segmentation network and an instance segmentation network and combine their results [Kirillov et al., 2019a]. However this approach can produce overlapping masks and can be improved by using a single network trained end-to-end for panoptic segmentation. Panoptic Feature Pyramid Networks [Kirillov et al., 2019b] adopt this strategy by ex-

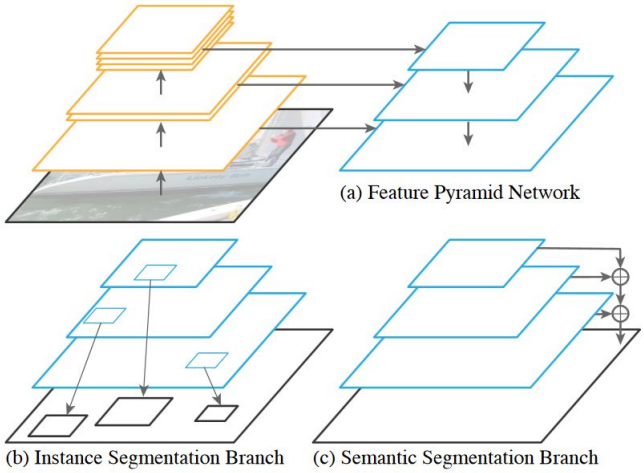

Figure 1.10 – Architecture of Panoptic Feature Pyramid Networks by [Kirillov et al., 2019b]

tracting feature maps at multiple scales. Those are then used in parallel by two branches, the first one being Mask-RCNN [K. He et al., 2017] which yields instance segmentation. The second branch yields semantic segmentation by fusing and up-sampling feature maps at different scales. The architecture of their approach is visible figure 1.10. More recently MaskFormer [B. Cheng et al., 2021] proposes to unify semantic and instance segmentation. To do so they predict a set of binary masks associated with labels indifferently for both *stuff* and *things*. They show that this unified approach improves both panoptic and semantic segmentation. They improved their network in its next iteration Mask2Former [B. Cheng et al., 2022];

1.8 Conclusion

In this chapter we introduced the fundamental concepts on which we based our work. The first section introduced euclidean transformations in 3D space, that are used to transform points from one coordinate frame to another, as well as twists, that represent the rotational and translational velocity of objects. We showed how twists could be used as a minimal unconstrained representation and how to integrate them using the exponential mapping. Then in section 2 we presented the measurement model used to represent a camera, which allows us to project a 3D point in image space. In section 3 we presented image primitives, and more particularly 2D keypoints, with their associated descriptors

that are used to represent them in an unique way. Following, in section 4 we introduced optimization techniques for linear and non linear least-squares that can be used in computer vision to fit a parametric model to observations. Then in section 5 we presented robust techniques that can be used to handle outliers. Finally in section 6 we showed the different types of semantic information and how to obtain them using deep learning based approaches.

Chapter 2

State of the Art of Classical and Semantic SLAM

Contents

2.1 Introduction

The goal of this chapter is to present previous works related to ours. It is divided into two parts. In the first one we introduce classical SLAM, beginning with its origins from Structure from Motion (SfM) and the Bundle Adjustment (BA). Following we show how it can be solved in real time by parallelizing tracking and mapping. We then incrementally build a full SLAM pipeline around this core by adding the initialization step, the relocalization and the loop closure. However we will not cover those last blocks in depth as we focus on the tracking and mapping blocks in this manuscript.

In the second part we present how semantic information can be introduced into SLAM. We begin with an introduction on the limits of classical SLAM systems, which justify the use of semantic information. Then we show how to create a consistent semantic map using multiple segmented images. Afterwards we present how semantic information can be used to improve the problem of relocalization, particularly in challenging cases such as long term relocalization or with important viewpoint variations. The penultimate section shows how semantic information can be used to make SLAM more robust in dynamic scenes, either by ignoring dynamic objects or by tracking them. Finally we present object based SLAM systems which use objects detected in the scene as high level landmarks to

improve camera pose estimation.

2.2 Classical SLAM

2.2.1 Introduction

SLAM is a fundamental problem in robotics and augmented reality. Its goal is to estimate the pose of a sensor moving in a scene while simultaneously creating a map of the environment. Multiple types of sensors can be used, such as Inertial Measurement Units (IMU), LiDARs or event cameras. In our case we focus on visual SLAM that uses only RGB cameras, either monocular or stereo. Cameras are very interesting, as they are low cost and require little power and space. However they present an additional complexity, being a projective sensor to understand the 3D world. To denote the pose of the camera we will use pose matrices in SE(3) at discrete timestamps, that we denote ${}^{c_i}\mathbf{T}_w$ for the i^{th} timestamp t_i and which represents the transformation between the fixed world coordinate frame $\mathcal{F}_w$ and the moving camera coordinate frame $\mathcal{F}_{c_i}$. The map will be represented using a set of 3D points expressed in $\mathcal{F}_w$ with j indices $\{{}^w\mathbf{X}_j\}$. Finally the 2D keypoint corresponding to the observation of ${}^w\mathbf{X}_j$ by the i^{th} camera is denoted ${}^i\mathbf{x}_j$.

2.2.2 Structure from Motion and the Bundle Adjustment

Structure from Motion

Structure from Motion is a problem closely related to SLAM as its goal is, using a set of unordered images taken from multiple points of view and at different moments, to estimate the 3D geometry of the scene and the poses of the cameras [Dellaert et al., 2000; Schonberger and Frahm, 2016]. The main two differences with SLAM are first, the lack of continuity, as images are unordered and can be captured at different moments. Second is the lack of computational constraints. Indeed, while SfM can be processed offline, SLAM must run in real time. However the solution of both problems is based on the minimization of the reprojection error.

Reprojection error

Keypoints extracted from images correspond to the 2D observation through the camera of 3D points in the scene. If the camera pose and 3D point position is known and

the camera model is perfect, then the points projected in the image should precisely correspond to keypoints:

$$ {}^{i}\mathbf{x}_j = p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j) \tag{2.1} $$

However as the estimations are not perfect and the measurement process is in reality noisy we write:

$$ {}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j) = \mathbf{e}_{i,j} \tag{2.2} $$

where $\mathbf{e}_{i,j} \in \mathbb{R}^2$ is the realization of a random variable that follows a given probability distribution, usually a 2-dimensional normal distribution with a mean of 0 and an unknown variance. This variable is called the reprojection error. To solve the problems of SfM and SLAM, the points positions and camera poses can be optimized so that their reprojection is aligned with 2D observations. Such optimization can be done by the bundle adjustment.

The bundle adjustment

The keystone of all recent SfM pipelines is the bundle adjustment [Grisetti et al., 2010; Triggs et al., 1999]. The goal of BA is to optimize the camera poses and the 3D map. To do so it maximizes the likelihood of variables given measurements taken from images, so that the variables best fit the measurements. Maximizing the likelihood of variables is equivalent to minimizing the reprojection error of 3D points in images. Thus, the BA minimizes the following cost function:

$$ E(\{{}^{c_i}\mathbf{T}_w\}, \{{}^{w}\mathbf{X}_j\}) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} ||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j)||^2 \tag{2.3} $$

$$ = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} ||\mathbf{e}_{i,j}||^2 = ||\mathbf{e}||^2 \tag{2.4} $$

where $\mathbf{e}_{i,j}$ is the reprojection error of point j in image i, $\mathbf{e} = (\mathbf{e}_{0,0}, \ddots, \mathbf{e}_{N,M})^\top$ is the vector of residuals, N is the number of images, M is the number of 3D points, $\{{}^{c_i}\mathbf{T}_w\}$ is the set of all camera poses and $\{{}^{w}\mathbf{X}_j\}$ is the set of all points positions. We show in the figure 2.1 an example of scene with 3 cameras and 2 points. As we can see the points are not well reprojected in the second image, the reprojection error (in cyan) is important. To decrease this error we can either move the second camera or move the points. However moving the points may require to also move the first and third camera to keep their reprojection errors low. The goal of BA is to optimize all those elements at once as they are all tightly

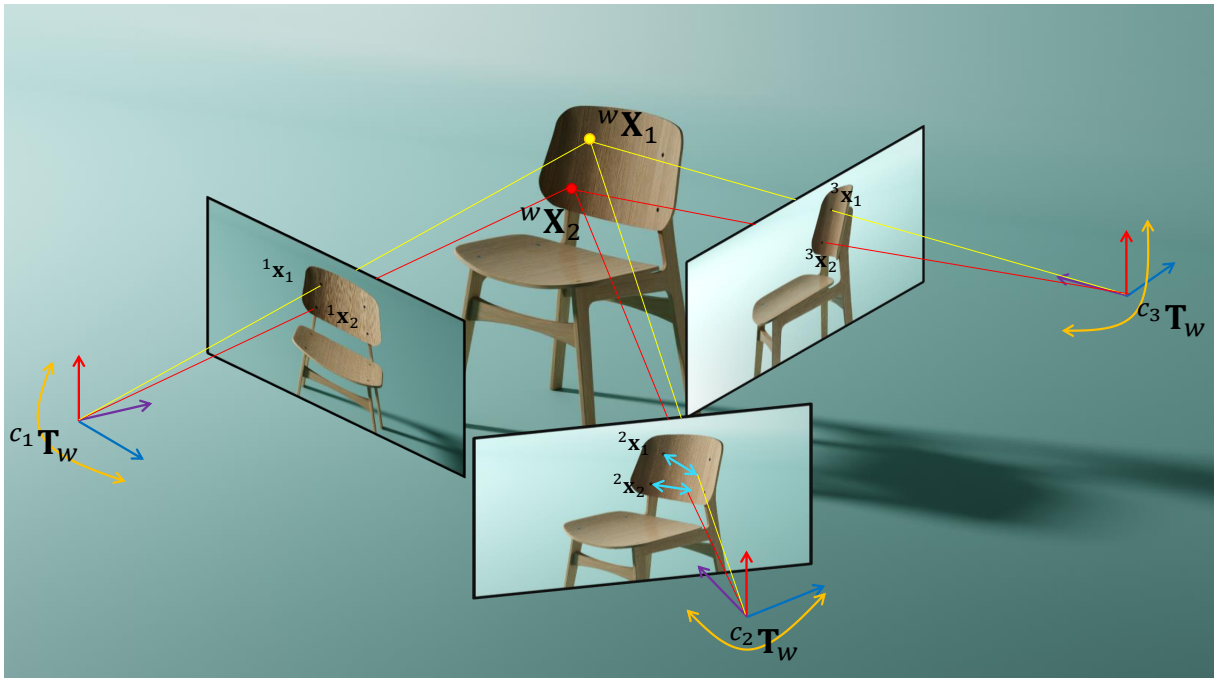

Figure 2.1 – Illustration of a scene with 3 cameras and 2 points. The reprojection errors of the points in the second camera are shown in cyan.

linked.

The cost function (2.3) is often represented using a graph, in which a vertex represents a variable to be optimized and an edge between vertices represents the associated error. An example of a simple graph with 3 poses and 5 points can be seen in figure 2.2.

Note that the BA cost function is sometimes written:

$$E(\{{}^{c_i}\mathbf{T}_w\}, \{{}^{w}\mathbf{X}_j\}) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} ||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j)||^2_{\Sigma^{-1}_{i,j}} \tag{2.5}$$

where $||.||_\Sigma$ is the Mahalanobis norm defined as:

$$||\mathbf{X}||_\Sigma = \sqrt{\mathbf{X}^\top \Sigma \mathbf{X}} \tag{2.6}$$

This allows to weight each residual individually. The matrix $\Sigma_{i,j}$ is often set to be an estimate of the covariance of the corresponding residual, thus giving more weight to residuals with low variance.

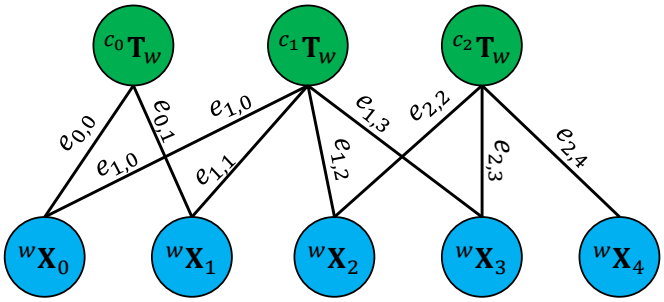

Figure 2.2 – Example of graph representing a Bundle Adjustment cost function with 3 poses and 5 points.

Solving the Bundle Adjustment

The Bundle Adjustment can be solved using the Levenberg-Marquardt algorithm that we introduced in the previous chapter. However we can see that the size of the corresponding Jacobian is $2L \times (6N + 3M)$ where L is the number of observed keypoints, which is at most NM when all points are visible in all images. The size of the corresponding Hessian is $(6N + 3M) \times (6N + 3M)$. Solving the normal equations can thus rapidly become untractable when the number of points and poses grows large. To alleviate this problem we can make use of the sparsity of the Jacobian. Indeed, the Jacobian contains the derivative of the reprojection error with respect to poses and points. As each reprojection error is impacted only by its own point and camera, its derivative with respect to all other points and cameras is null. This explains the specific sparse structure of the Jacobian that we can see in equation 2.7. Furthermore as points are often seen from a limited number of images, they have no reprojection error in many images and thus a null derivative, which

also adds zero entries in the matrix.

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \mathbf{e}_{0,0}}{\partial^{c_0}\mathbf{T}_w} & \frac{\partial \mathbf{e}_{0,0}}{\partial^{c_1}\mathbf{T}_w} & \cdots & \frac{\partial \mathbf{e}_{0,0}}{\partial^{c_N}\mathbf{T}_w} & \frac{\partial \mathbf{e}_{0,0}}{\partial^{w}\mathbf{X}_0} & \frac{\partial \mathbf{e}_{0,0}}{\partial^{w}\mathbf{X}_1} & \cdots & \frac{\partial \mathbf{e}_{0,0}}{\partial^{w}\mathbf{X}_M} \\ \frac{\partial \mathbf{e}_{1,0}}{\partial^{c_0}\mathbf{T}_w} & \frac{\partial \mathbf{e}_{1,0}}{\partial^{c_1}\mathbf{T}_w} & \cdots & \frac{\partial \mathbf{e}_{1,0}}{\partial^{c_N}\mathbf{T}_w} & \frac{\partial \mathbf{e}_{1,0}}{\partial^{w}\mathbf{X}_0} & \frac{\partial \mathbf{e}_{1,0}}{\partial^{w}\mathbf{X}_1} & \cdots & \frac{\partial \mathbf{e}_{1,0}}{\partial^{w}\mathbf{X}_M} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{e}_{N,M}}{\partial^{c_0}\mathbf{T}_w} & \frac{\partial \mathbf{e}_{N,M}}{\partial^{c_1}\mathbf{T}_w} & \cdots & \frac{\partial \mathbf{e}_{N,M}}{\partial^{c_N}\mathbf{T}_w} & \frac{\partial \mathbf{e}_{N,M}}{\partial^{w}\mathbf{X}_0} & \frac{\partial \mathbf{e}_{N,M}}{\partial^{w}\mathbf{X}_1} & \cdots & \frac{\partial \mathbf{e}_{N,M}}{\partial^{w}\mathbf{X}_M} \end{pmatrix} \tag{2.7}$$

$$= \begin{pmatrix} \frac{\partial \mathbf{e}_{0,0}}{\partial^{c_0}\mathbf{T}_w} & 0 & \cdots & 0 & \frac{\partial \mathbf{e}_{0,0}}{\partial^{w}\mathbf{X}_0} & 0 & \cdots & 0 \\ 0 & \frac{\partial \mathbf{e}_{1,0}}{\partial^{c_1}\mathbf{T}_w} & \cdots & 0 & \frac{\partial \mathbf{e}_{1,0}}{\partial^{w}\mathbf{X}_0} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{\partial \mathbf{e}_{N,M}}{\partial^{c_N}\mathbf{T}_w} & 0 & 0 & \cdots & \frac{\partial \mathbf{e}_{N,M}}{\partial^{w}\mathbf{X}_M} \end{pmatrix} \tag{2.8}$$

The components of the Jacobian can be computed using the chain rule, first for derivatives with respect to the point position:

$$\frac{\partial \mathbf{e}_{i,j}}{\partial^{w}\mathbf{X}_j} = \frac{\partial \pi({}^{i}\mathbf{x}_j)}{\partial^{i}\mathbf{x}_j} \frac{\partial ({}^{c_i}\mathbf{T}_w \, {}^{w}\bar{\mathbf{X}}_j)}{\partial^{w}\mathbf{X}_j} \tag{2.9}$$

$$= \frac{1}{{}^{c}Z_j} \begin{pmatrix} f_x & 0 & -f_x \frac{{}^{c}X_j}{{}^{c}Z_j} \\ 0 & f_y & -f_y \frac{{}^{c}Y_j}{{}^{c}Z_j} \end{pmatrix} {}^{c_i}\mathbf{R}_w \tag{2.10}$$

as the transformation ${}^{c_i}\mathbf{T}_w \, {}^{w}\bar{\mathbf{X}}_j$ can be written as ${}^{c_i}\mathbf{R}_w \, {}^{w}\mathbf{X}_j + {}^{c_i}\mathbf{t}_w$, which is linear in ${}^{w}\mathbf{X}_j$. And then for the derivatives with respect to the camera pose:

$$\frac{\partial \mathbf{e}_{i,j}}{\partial^{c_i}\mathbf{T}_w} = \frac{\partial \pi({}^{i}\mathbf{x}_j)}{\partial^{i}\mathbf{x}_j} \frac{\partial ({}^{c_i}\mathbf{T}_w \, {}^{w}\bar{\mathbf{X}}_j)}{\partial^{c_i}\mathbf{T}_w} \tag{2.11}$$

$$= \frac{1}{{}^{c}Z_j} \begin{pmatrix} f_x & 0 & -f_x \frac{{}^{c}X_j}{{}^{c}Z_j} \\ 0 & f_y & -f_y \frac{{}^{c}Y_j}{{}^{c}Z_j} \end{pmatrix} \begin{pmatrix} \mathbf{I}_3 & -[{}^{c}\mathbf{X}_j]_\times \end{pmatrix} \tag{2.12}$$

as explained in the previous chapter, the derivative with respect to a 6 DoF pose is computed on the Lie algebra $\mathfrak{se}(3)$:

$$\left.\frac{\partial(\exp_{\mathfrak{se}}(\boldsymbol{\xi})^{c_i}\mathbf{T}_w{}^{w}\bar{\mathbf{X}}_j)}{\partial\boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=0} = \left.\frac{\partial(\mathbf{Q}^{w}\bar{\mathbf{X}}_j)}{\partial\mathbf{Q}}\right|_{\mathbf{Q}=\exp_{\mathfrak{se}}(\boldsymbol{\xi})^{c_i}\mathbf{T}_w={}^{c_i}\mathbf{T}_w} \left.\frac{\partial\mathbf{Q}^{c_i}\mathbf{T}_w}{\partial\mathbf{Q}}\right|_{\mathbf{Q}=\exp_{\mathfrak{se}}(\boldsymbol{\xi})=\mathbf{I}_4} \left.\frac{\exp_{\mathfrak{se}}(\boldsymbol{\xi})}{\boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=0} \tag{2.13}$$

$$= \left({}^{w}\bar{\mathbf{X}}_j{}^{\top}\otimes\mathbf{I}_3\right)\left({}^{c_i}\mathbf{T}_w{}^{\top}\otimes\mathbf{I}_3\right)\begin{pmatrix}\mathbf{0}_{3\times3} & -[\mathbf{e}_1]_\times\\ \mathbf{0}_{3\times3} & -[\mathbf{e}_2]_\times\\ \mathbf{0}_{3\times3} & -[\mathbf{e}_3]_\times\\ \mathbf{I}_3 & \mathbf{0}_{3\times3}\end{pmatrix} \tag{2.14}$$

where $\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\}$ is the canonical base of $\mathbb{R}^3$ and $\otimes$ is the Kronecker product of two matrices. This equation, after development is equal to $(\mathbf{I}_3, -[{}^{c}\mathbf{X}_j]_\times)$.

Those derivatives correspond to the monocular case. For the stereo case, the derivative of the projection operator must be replaced by the following 3×3 matrix:

$$\frac{1}{{}^{c}Z_j}\frac{\partial\pi({}^{i}\mathbf{x}_j)}{\partial{}^{i}\mathbf{x}_j} = \begin{pmatrix} f_x & 0 & -f_x\frac{{}^{c}X_j}{{}^{c}Z_j}\\ 0 & f_y & -f_y\frac{{}^{c}Y_j}{{}^{c}Z_j}\\ f_x & 0 & -f_x\frac{({}^{c}X_j-b)}{{}^{c}Z_j}\end{pmatrix} \tag{2.15}$$

where b is the stereo baseline. More details about the derivative of the exponential map and the derivatives of matrix products can be found in [Blanco, 2010].

As we can see the Jacobian is highly sparse. This is because points and poses only have an impact on their own reprojection error and each 3D point is only seen by a limited number of cameras. This creates a specific structure that corresponds to the adjacency matrix of the associated graph. The Hessian, that is approximated as $\mathbf{H}=\mathbf{J}^{\top}\mathbf{J}$ inherits from this specific structure. We show in figure 2.3 an example of Jacobian and Hessian.

The Hessian can be written using blocks:

$$\mathbf{H} = \begin{pmatrix}\mathbf{H_{PP}} & \mathbf{H_{PC}}\\ \mathbf{H_{PC}}^{\top} & \mathbf{H_{CC}}\end{pmatrix} \tag{2.16}$$

where $\mathbf{H_{CC}}$ is a block diagonal matrix which represents the second order derivatives of camera poses, $\mathbf{H_{PP}}$ is a block diagonal matrix which represents the second order deriva-

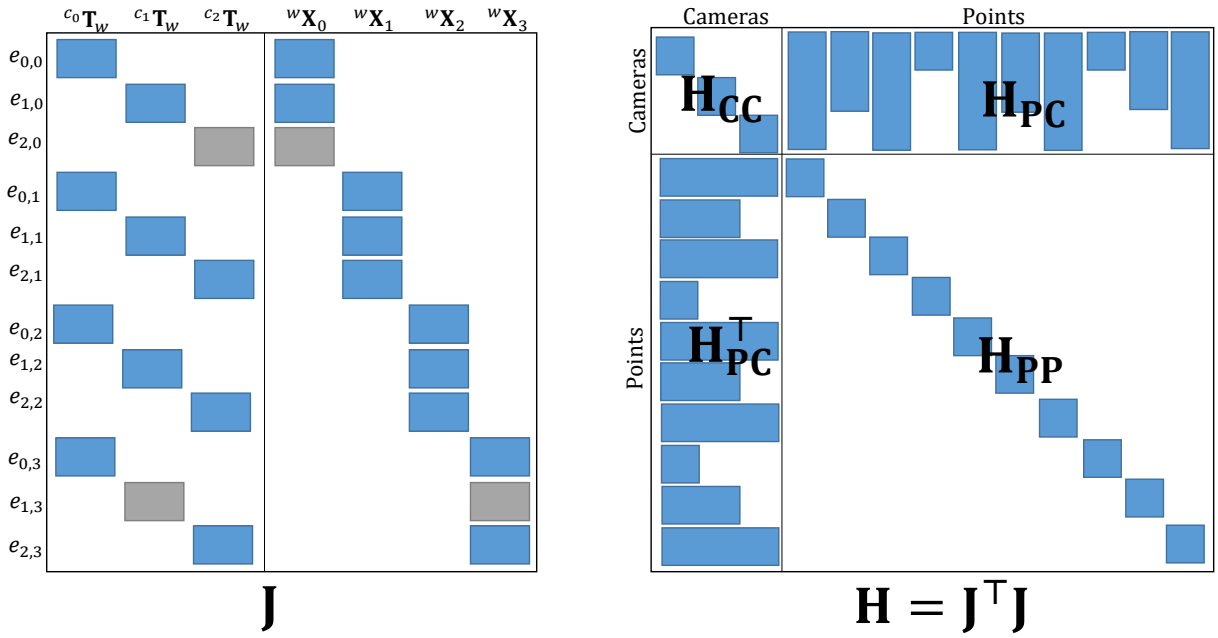

Figure 2.3 – Left: example of Jacobian structure with 3 camera poses and 4 points. Right: example of Hessian structure with 3 camera poses and 10 points.

tives of point positions and $\mathbf{H_{PC}}$ represents the crossed derivatives of camera poses and point positions. It should be noted that the matrix $\mathbf{H_{PP}}$ is in practice much larger than $\mathbf{H_{CC}}$ as there are orders of magnitude more 3D points ($\approx 10^4$ to 10^5 for medium scale scenes) than camera poses ($\approx 10^2$). Hence the Hessian is essentially block diagonal with two dense blocks for crossed derivatives. To obtain the optimal increment we must solve the normal equation:

$$\mathbf{J}^\top\mathbf{J}\boldsymbol{\delta} = \mathbf{J}^\top\mathbf{e} \tag{2.17}$$

By separating cameras and points, it can be written:

$$\begin{pmatrix} \mathbf{H_{CC}} & \mathbf{H_{PC}} \\ \mathbf{H_{PC}}^\top & \mathbf{H_{PP}} \end{pmatrix} \begin{pmatrix} \delta_\mathbf{C} \\ \delta_\mathbf{P} \end{pmatrix} = \begin{pmatrix} \mathbf{b_C} \\ \mathbf{b_P} \end{pmatrix} \tag{2.18}$$

where $(\delta_\mathbf{C}, \delta_\mathbf{P})$ is the incremental update in terms of cameras poses and point position and $\mathbf{b} = \mathbf{J}^\top\mathbf{e}$. Using the Schur complement of $\mathbf{H_{PP}}$ that we denote $\bar{\mathbf{H}}_\mathbf{PP}$ and introducing a new variable $\bar{\mathbf{b}}_\mathbf{C}$ we can write:

$$\bar{\mathbf{H}}_\mathbf{PP} = \mathbf{H_{CC}} - \mathbf{H_{PC}}\mathbf{H_{PP}}^{-1}\mathbf{H_{PC}}^\top \qquad \bar{\mathbf{b}}_\mathbf{C} = \mathbf{b_C} - \mathbf{H_{PC}}\mathbf{H_{PP}}^{-1}\mathbf{b_P}^\top \tag{2.19}$$

This substitution allows to compute $\delta_{\mathbf{C}}$ using a smaller system than the original one:

$$\begin{pmatrix} \bar{\mathbf{H}}_{\mathbf{CC}} & \mathbf{0} \\ {\mathbf{H}_{\mathbf{PP}}}^{-1}{\mathbf{H}_{\mathbf{PC}}}^{\top} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \delta_{\mathbf{C}} \\ \delta_{\mathbf{P}} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{b}}_{\mathbf{C}} \\ {\mathbf{H}_{\mathbf{PP}}}^{-1}\mathbf{b}_{\mathbf{P}} \end{pmatrix} \tag{2.20}$$

This system can be obtained by left multiplying the equation 2.18 successively by the following matrices:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & {\mathbf{H}_{\mathbf{PP}}}^{-1} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbf{I} & -\mathbf{H}_{\mathbf{PC}} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{2.21}$$

Such system can be solved using Cholesky factorization for example which has a complexity growing with the cube of the number of cameras. It is thus much faster than solving the system that includes 3D points. Then, the point positions can be computed efficiently using back substitution:

$$\delta_{\mathbf{P}} = {\mathbf{H}_{\mathbf{PP}}}^{-1}\mathbf{b}_{\mathbf{P}} - {\mathbf{H}_{\mathbf{PP}}}^{-1}{\mathbf{H}_{\mathbf{PC}}}^{\top}\delta_{\mathbf{C}} \tag{2.22}$$

More details regarding the bundle adjustment and how to solve it can be found in [Engels et al., 2006; Triggs et al., 1999]. Repeating those steps allow us to obtain an accurate estimate of point positions and camera poses. However the bundle adjustment is not solved using only raw image data, a pre-processing step must be applied to extract characteristic points.

Front-end and back-end

As we saw, keypoints must first be extracted from images before solving the BA. Those steps are often referred to as the front end and the back end. The goal of the front-end is to preprocess images by cleansing them, performing for example histogram equalization to reduce over and under exposition [Pizer et al., 1987; 1990]. The front-end then extracts keypoints from images which are matched with previously seen keypoints. Finally the matched points are processed by the Bundle Adjustment that yields optimized point positions and camera poses. The goal of the front-end is also to compute reasonably good estimate to serve as initialization for the BA. Indeed, the cost function is non-convex and the optimization step by step may get stuck on a local minima.

Direct and indirect methods

It should be noted however that not all SLAM approaches use keypoints. Some approaches directly use image intensity information. They have the advantages of discarding less information and of being able to work even with featureless images. To do so they minimize the following cost function, called the photometric error:

$$E({}^{c_{i+1}}\mathbf{T}_w) = \sum_{\mathbf{x}_j \in \Omega} ||\mathbf{I}_i(\mathbf{x}_j) - \mathbf{I}_{i+1}(p({}^{c_{i+1}}\mathbf{T}_w \pi(\mathbf{x}_j)^{-1}))||^2 \tag{2.23}$$

where $\mathbf{I}_i$ is the i^{th} image and Ω is a subset of image pixels. The goal of this equation is to find the relative pose of consecutive cameras by aligning their images. However the main disadvantage of this approach is that it relies on the assumption that the brightness of a point is constant. This assumption is obviously false when considering reflective surfaces or sudden illumination changes. In those cases indirect approaches have the upper hand as extracted keypoints are supposed to be invariant towards those changes. One of the most well known direct SLAM is LSD-SLAM [Engel et al., 2014] that used edge points to track the camera and build the map. Furthermore tracking was performed by optimizing both the pose and the scale factor of the scene, allowing the system to handle scale variation. This approach however is not suitable for the bundle adjustment. Indeed as points are semi-densely extracted, the map points are correlated and the corresponding Hessian loses its sparsity. Trajectory optimization is thus done using pose graph optimization. Some visual odometry systems such as SVO [Forster et al., 2014] and DSO [Engel et al., 2017] also apply direct approaches to track the camera. In this manuscript we will focus on indirect approaches.

Pipeline

At this point the "SLAM" pipeline, visible in figure 2.4 is straightforward: given a set of images we extract and match keypoints in the frond end. Then we solve the Bundle Adjustment which yields camera poses and 3D points positions. We show how to modify this pipeline in the following section to obtain a real-time SLAM system.

2.2.3 From filters to keyframes

To create a SLAM system, the BA must be modified, indeed its computational complexity is too high to allow real-time operations. In practice optimizing it can take from

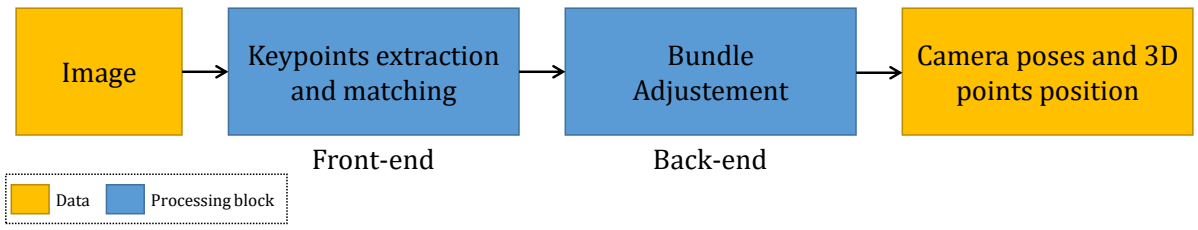

Figure 2.4 – Illustration of front and back end in a SLAM system.

0.1 second to tens of seconds for very large graphs. Two systems can thus be implemented: filter based or keyframe based SLAM systems [Engels et al., 2006].

Filter based SLAM systems

MonoSLAM [Davison et al., 2007] is the first monocular SLAM system that solves the bundle adjustment in real time. To do so they model the current camera pose and points positions as a single high dimensional gaussian variable, called the map state. They use an Extended Kalman Filter (EKF) to predict the map state using a constant velocity model for the camera. Then they update the map state with measurements from the Shi and Tomasi detector [J. Shi et al., 1994]. This kind of approach, called filter based, could work in real-time but was limited to small scenes. Indeed at each time step the whole map state and its covariance matrix must be computed. As the size of the covariance matrix grows quadratically with the number of points, MonoSLAM is limited to use only 100 feature points which does not allow to cover a space larger than a desk. Furthermore EKF approaches require the system to be approximately linear, which is not the case when considering the projection function.

Parallel Tracking and Mapping

To solve the large scale issue of filter based SLAM systems, PTAM [Klein and Murray, 2007] proposes to split tracking and mapping. Although the camera pose should be estimated in real-time and can change rapidly, the geometry of the scene is rather static and mainly used to solve camera tracking for augmented reality. Hence they solve only camera pose estimation on a thread, which is updated in real-time while on another separated thread they can triangulate new map points and optimize the camera trajectory and map geometry at a much lower rate. This allows them to apply an expensive bundle adjustment on a high number of points.

Furthermore as tracking and mapping are separated, "*Tracking is no longer probabilistically slaved to the map-making procedure*" [Klein and Murray, 2007] like it was for MonoSLAM and robust approaches can be applied to minimize the impact of mapping errors on the tracking thread.

Hence in the tracking thread of PTAM, the i^{th} camera pose is obtained by minimizing the reprojection error of fixed 3D map points:

$$E({}^{c_i}\mathbf{T}_w) = \sum_{j=0}^{M-1} ||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j)||^2 \tag{2.24}$$

As the camera moves smoothly in the scene, the previous camera pose ${}^{c_{i-1}}\mathbf{T}_w$ can be used to initialize the pose estimation and for keypoints matching. Indeed by assuming a constant velocity model, the previous pose can be updated and used to project map points into the image. Keypoints can then be searched in areas centered around the projected points. This allows to decrease both the number of descriptors comparison as well as the probability of wrong matches. A similar approach was also initially proposed by [Mouragnon et al., 2006].

Keyframes and triangulation

Frames captured by a camera with a frequency of 30 Hz contain redundant information. In filter based SLAM systems they must nonetheless be all processed similarly. By decoupling tracking and mapping, PTAM can select a representative sample of frames to optimize them in the bundle adjustment instead of optimizing all frames. Those frames are called keyframes and act as anchors for other frames. In PTAM keyframes are chosen to cover space in the 3D scene: if a frame is far enough from the previous keyframe it is chosen to become a new keyframe.

The triangulation of two matched keypoints in the i^{th} and the k^{th} images, ${}^{i}\mathbf{x}_j$ and ${}^{k}\mathbf{x}_j$ can be performed by computing the intersection between the rays $(\mathbf{C_i}, {}^{i}\mathbf{x}_j)$ and $(\mathbf{C_k}, {}^{k}\mathbf{x}_j)$, where $\mathbf{C_i}$ and $\mathbf{C_k}$ are the cameras 3D centers. However due to noise, rays do not usually intersect. Thus the point 3D position can be computed as the closest point to both rays using the midpoint method [R. I. Hartley and Sturm, 1997]. Keyframes are used by the mapping thread to create new map points which are triangulated using multiple views.

In opposition to filter based SLAM systems or SfM, this kind of approach is referred to as keyframe based SLAM system.

ORB-SLAM [Mur-Artal et al., 2015] is a recent monocular SLAM system inspired from

PTAM that also uses a multi-threaded architecture. ORB [Rublee et al., 2011] keypoints are extracted and matched very efficiently by the front end. The keypoints are used to estimate the current frame pose by matching them with the last frame or the nearest keyframe and with the map. The current frame can then be converted into a new keyframe. This happens when too few keypoints are tracked, meaning that we enter an area that is not mapped and thus that we need a new keyframe to triangulate new map points. Finally keyframes and map points are optimized within the bundle adjustment. ORB-SLAM uses a covisibility graph to measure the visual proximity between keyframes. This allows them to optimize only a small set of keyframes that share a high covisibility with the current one, reducing the computational burden of the bundle adjustment. ORB-SLAM was later improved with ORB-SLAM 2 [Mur-Artal and Tardós, 2017] that can use stereo and RGB-D cameras to handle pure rotations and estimate a metric map.

A comparison of filter based SLAM that marginalizes the previous camera poses and keyframes based SLAM that sparsify the camera trajectory was proposed by [Strasdat et al., 2012]. They show that generally the accuracy of SLAM systems depends more on the number of visible points than on the number of keyframes, as long as there are enough keyframes to cover the trajectory. As the complexity of keyframe based SLAM systems is linear in the number of points while the complexity of filter based SLAM is cubic in the number of points, the choice to obtain the less computationally expensive system is straightforward.

The two main threads of a basic SLAM system

We show in figure 2.5 an example of a simple keyframe based SLAM system. In the tracking thread keypoints are extracted from images and used for camera pose estimation. The current image can then be chosen to be a keyframe, which is handled by the mapping thread. It creates new map points and performs the bundle adjustment to refine the poses and point positions.

2.2.4 Initialization

In the previous sections we demonstrated how to estimate the pose of a new frame using 3D map points and how to triangulate new 3D points knowing the poses of the camera. Those approaches work well however a problem arises when neither the camera pose nor the point position are known, which happens when the algorithm start. This is

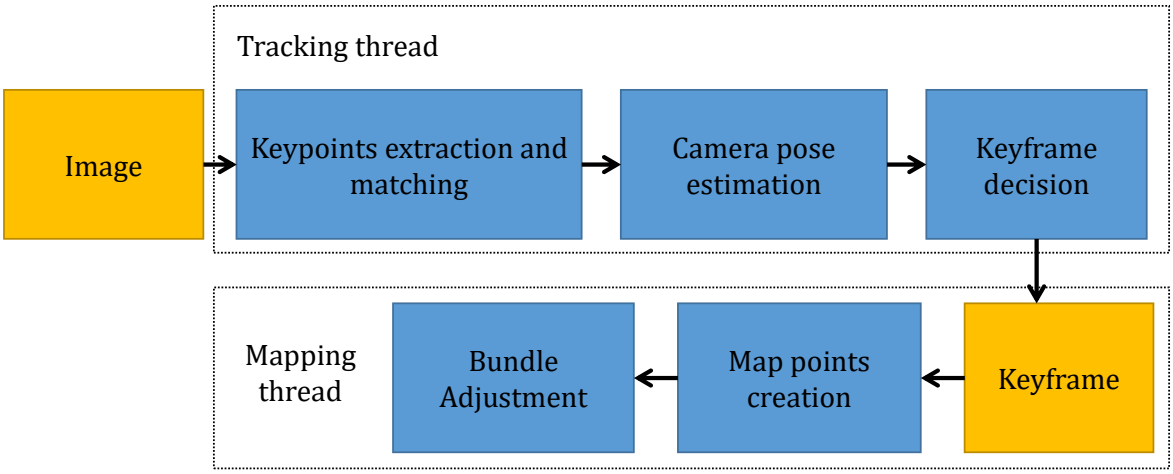

Figure 2.5 – Pipeline of a simple Keyframe based SLAM system.

often referred to as a "chicken and egg problem" as we need camera pose to triangulate new points but also need points to estimate new poses. To solve this problem, an initialization or bootstrap step is required. In the RGB-D and stereo cases the initialization step is trivial as map points can directly be triangulated using a single view. In the monocular case, ORB-SLAM systems [Mur-Artal and Tardós, 2017; Mur-Artal et al., 2015] solve the problem by applying principles from 2-views geometry [R. Hartley and Zisserman, 2003; Y. Ma et al., 2004]. Using matches between 2 images they estimate both a fundamental matrix $\mathbf{F}$ and an homography $\mathbf{H}$ that link matches according to the following equations:

$$ {}^{0}\mathbf{x}_j{}^{\top}\,\mathbf{F}\,{}^{1}\mathbf{x}_j = 0 \quad \text{or} \quad {}^{0}\mathbf{x}_j = \mathbf{H}\,{}^{1}\mathbf{x}_j \tag{2.25} $$

where ${}^{0}\mathbf{x}_j{}^{\top}$ and ${}^{1}\mathbf{x}_j{}^{\top}$ are corresponding keypoints in the first and second images. The homography matrix links points in a planar scene or for pure rotations while the fundamental matrix works for the general case. The transformation that is supported by the most inliers is then chosen and the relative pose between both cameras is extracted from it. Knowing the relative pose allows the algorithm to triangulate the matches, which starts the map. A BA is then applied to refine the map and the relative pose. The first camera pose is often chosen to be the world coordinate frame for the whole map.

2.2.5 Relocalization and loop closure

We present here the processes of relocalization and loop closure that are not necessary for SLAM but are commonly used to make it more robust and precise. Our goal is not to thoroughly present those algorithms and more details can be found in [Duong, 2019; Lowry et al., 2015; Masone and Caputo, 2021] for relocalization.

Relocalization

Camera pose estimation in the tracking thread can fail to converge towards a good solution. Multiple effects can cause a failure. For example rapid camera movements can create motion blur, limiting the amount of keypoints that can be extracted from the image. If the camera moves too fast it can also move to a yet unmapped region of the scene, making tracking unfeasible. In those cases tracking is said to be *lost* as the current pose relative to previous poses is unknown. To solve this problem SLAM systems need to be able to estimate the current camera pose using only the map (comprising map points and keyframe poses). This process is called the relocalization. In classical SLAM systems it is solved by first finding a set of keyframes that are visually close from the current frame (called the query). The hypothesis here is that visually close frames should also be close in 3D space. This problem is solved using image retrieval techniques. For example in ORB-SLAM2 [Mur-Artal and Tardós, 2017], images are represented using bag of visual words [Gálvez-López and Tardos, 2012; Sivic and Zisserman, 2003]. A vocabulary made of clustered descriptors is first created offline using a set of generic images. Then while the SLAM system runs, keyframes are used to compute an histogram of visual words. Each histogram corresponds to a vector that describes the image. The query frame can thus be associated to visually close images by finding its nearest neighbors in the histogram space. Given those close candidate keyframes, the best one is chosen by performing geometrical checks. Keypoints are matched between the query and each of the candidate and a geometrical model (homography or fundamental matrix) is robustly fitted. The candidate that is supported by the most points is chosen as the reference keyframe. It is then used to associate 3D map points to keypoints and estimate the query pose by minimizing the reprojection error. The system then returns to its classical state by tracking the camera. Approaches based on local handcrafted descriptors are often limited by the lack of invariance of descriptors. Thus they can fail when facing important luminosity and viewpoint changes. Recently deep learning based approaches have been developed to be more robust.

For example NetVLAD [Arandjelovic et al., 2016] inspired from VLAD [Jégou et al., 2010] extracts D dimensional image features using a CNN. Features are then softly assigned to one of K learned clusters. Finally the residuals between features and each cluster centroid are averaged to obtain K, D dimensional descriptors that can bee seen as a $K \times D$ VLAD descriptor. Other approaches such as [X. Wang et al., 2022] directly use the output of a CNN to build stable descriptors.

Loop closure

As we saw earlier, the SLAM pipeline mainly alternates between tracking and mapping. The pose of the camera is estimated using the visible map and then used to triangulate new map points. As the pose of the camera is not perfectly estimated, the position of 3D points is noisy, which is then reflected on the future poses and points and so on. This effect is called the drift as estimation noise accumulates and it is inherent to all odometry systems. However this error can be estimated and minimized by revisiting a place that was previously mapped. Such place is called a loop and the correction process is the loop closure, often referred to as *long-term data association*. The first task of loop closure is to detect a loop. To do so approaches similar to relocalization based on image retrieval can be used, yielding loop hypothesis. Those hypothesis are then verified using geometric approaches to filter out false positives. Hypothesis verification is done by robustly computing 7 DoF similarity transformations from 3D-3D matches between the current frame and the candidates. Similarly to relocalization, the transformation supported by the most inliers is selected. It is then applied on the current frame pose and its neighbors to align it with the loop and fuse the map points. This correction is followed by pose graph optimization which moves each keyframe pose to distribute the error along the whole trajectory. Finally in ORB-SLAM 2 a global bundle adjustment is performed with all the map points and keyframes. Those steps are computationally expensive, taking up to several seconds to finish, which is why a whole CPU thread is dedicated to it. Hence the SLAM system can correct drift in the trajectory, as illustrated in the figure 2.6.

The complete SLAM pipeline

We show in the figure 2.7 a SLAM pipeline including initialization, relocalization and loop closure, typical for state of the art systems such as [Mur-Artal and Tardós, 2017].

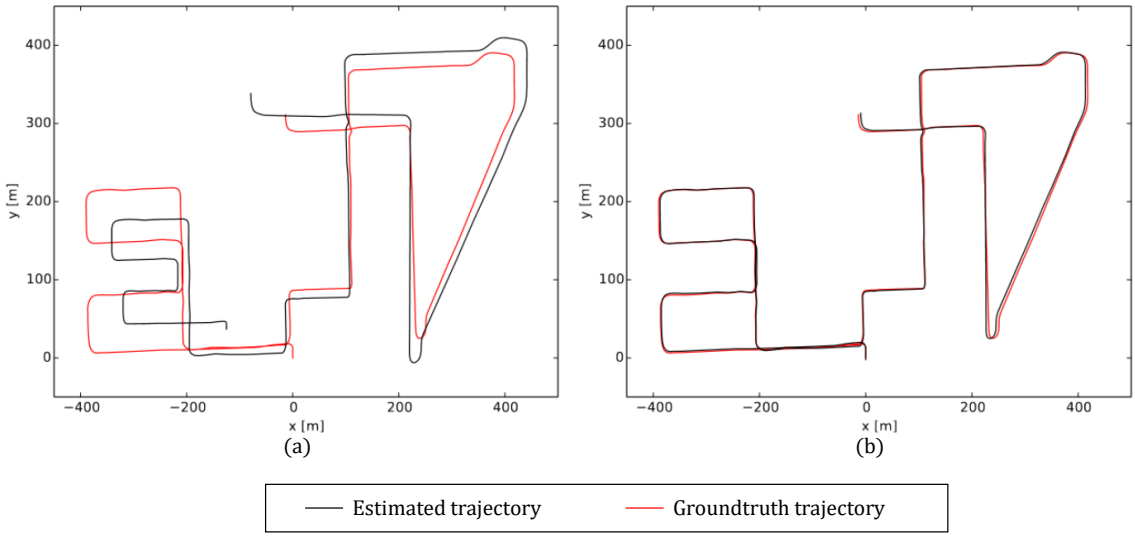

Figure 2.6 – Trajectory from the KITTI dataset (seen from above) in a monocular setting with scale drift (a) and stereo setting with no scale drift (b). Image from ORB-SLAM 2 [Mur-Artal and Tardós, 2017]

.

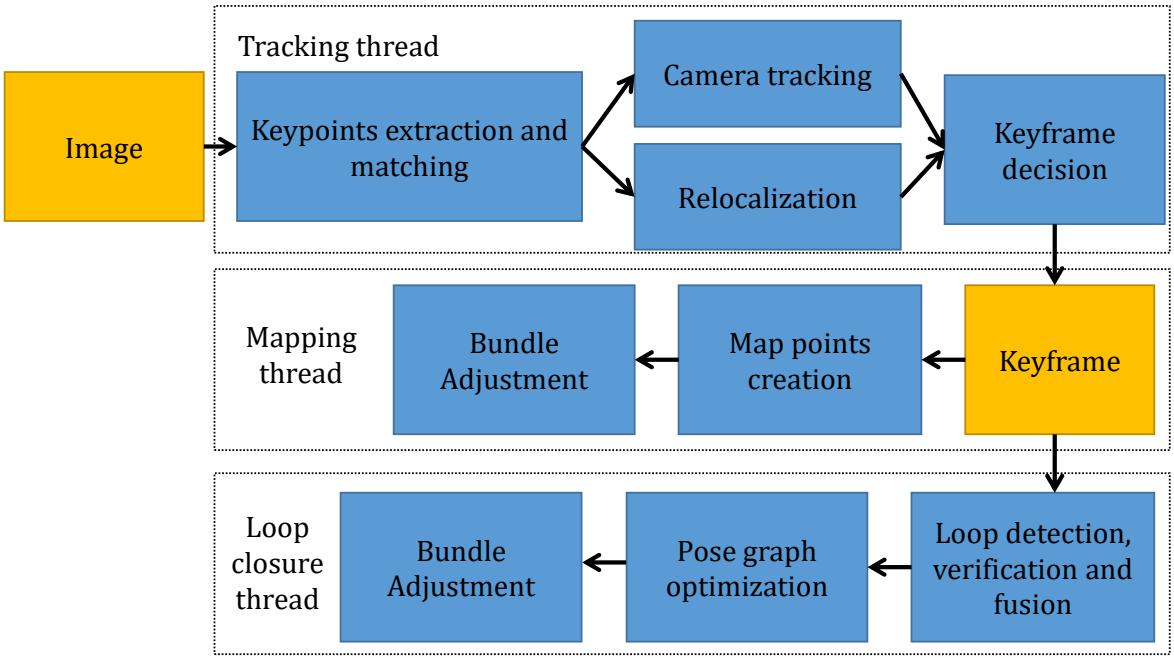

Figure 2.7 – Pipeline of a classical SLAM system with three threads for tracking, mapping and loop closure.

2.2.6 On the evaluation of SLAM systems

To compare the strengths and weaknesses of different SLAM systems it is primary to compare the accuracy of their estimations. To do so public datasets with ground truth must be used and evaluation metrics must be defined. In this manuscript we will mainly use three datasets for SLAM:

- The TUM RGB-D dataset [J. Sturm et al., 2012] which contains a set of short indoor sequences filmed with an RGB-D Kinect camera at 30 Hz. The sequences depict a typical desk space and may contain moving people. Camera trajectory ground truth is obtained using a motion capture system that estimates the 3D position of reflective markers attached to the camera. This kind of system is highly precise with a frame to frame relative error of 1mm and 0.5°.
- The Kitti odometry dataset [Geiger et al., 2012; 2013] which contains many outdoors sequences of variable length (from hundreds of meters to kilometers). This dataset was created using two stereo cameras (RGB and grayscale at 10 Hz) with a large baseline (about 54 cm), mounted on a car. The sequences depict a variety of urban scenes, from crowded pedestrian streets, to highways, or residential neighborhoods. Camera trajectory groundtruth is obtained using a GPS and an Inertial Measurement Unit (IMU) containing an accelerometer and a gyroscope. This kind of system is less precise than the previous one with ground truth errors estimated at 5 cm. This dataset also contains LiDAR scans.
- The Kitti tracking dataset which was created like the odometry dataset but also contains the pose of various objects in the scene (cars, trucks, pedestrians, cyclists, etc.). Those were obtained by manually annotating LiDAR scans.

As SLAM algorithms estimate both the trajectory and the geometry of the scene it seems natural to evaluate both. However obtaining the ground truth for the geometry of a scene is cumbersome, sometimes even impossible for large scale, outdoors scenes. Furthermore in the context of augmented reality the precision of camera pose estimation is more important than the precision of geometry. Finally as both estimations are correlated, evaluating only the camera pose error is a good indicator of the map quality.

SLAM systems are mainly evaluated through 2 metrics [Kümmerle et al., 2009; J. Sturm et al., 2012]: the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE). Given a sequence of N estimated poses $\{{}^{c_0}\mathbf{T}_w, \ldots, {}^{c_{N-1}}\mathbf{T}_w\}$ and N ground truth poses $\left\{{}^{c_0^*}\mathbf{T}_w, \ldots, {}^{c_{N-1}^*}\mathbf{T}_w\right\}$. The i^{th} absolute trajectory error between the ground truth and

the estimated pose can be computed as:

$$\mathbf{ATE}_i = {}^{w}\mathbf{T}_{c_i^*} \, {}^{c_i}\mathbf{T}_{w} \tag{2.26}$$

where the ground truth and estimated trajectories need to be aligned using the method of Horn [Horn, 1987] or Umeyama [Umeyama, 1991]. Indeed they do not necessarily share the same coordinate frame. [J. Sturm et al., 2012] proposes to evaluate the translational part of this error:

$$ATE_i^t = ||trans(\mathbf{ATE}_i)||_2 \tag{2.27}$$

It must then be aggregated over the whole trajectory, by computing for example its mean, median or RMSE:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} ATE_i^{t\,2}} \tag{2.28}$$

The ATE measures the global consistency of the trajectories by computing the distances between pairs of trajectories. However it is sensitive to the time where the error occurs [Kümmerle et al., 2009]. An early error on the trajectory has a greater impact on the ATE than a later one. On the other hand the RPE measures the local drift on a part of the trajectory, defined using either spatial or temporal intervals.

$$\mathbf{RPE}_i = ({}^{w}\mathbf{T}_{c_i^*} \, {}^{w}\mathbf{T}_{c_{i+\Delta}^*})^{-1} \, ({}^{w}\mathbf{T}_{c_i} \, {}^{w}\mathbf{T}_{c_{i+\Delta}}) \tag{2.29}$$

where Δ here is the time interval. The RPE is often split into two parts:

$$RPE_i^t = ||trans(\mathbf{RPE}_i)|| \qquad RPE_i^R = ||angle(rot(\mathbf{RPE}_i))|| \tag{2.30}$$

where $angle(\mathbf{R})$ denotes the rotation angle of $\mathbf{R}$. As the RPE depends on the choice of the interval, we can compute it for different Δ and aggregate the results by computing their mean. As for the APE, the results must then be aggregated over the whole trajectory, using the RMSE, the mean or the median.

A note on non-deterministic behaviors.
The most cited SLAM system, ORB-SLAM2 [Mur-Artal and Tardós, 2017] which is used for many comparisons and even built upon to create new algorithms has a non deterministic behavior. This means that running ORB-SLAM2 on the same sequence multiple times gives different results each time. This behavior is mainly due to the fact that it is based on multiple unsynchronized threads. Thus, depending on the CPU load some parts of the code may be executed after others or even not at all. ORB-SLAM2 also uses random processes, such as RANSAC, which is particularly impactful during the initialization. However RANSAC can be deterministic by setting the same seed of the random process.
To enable a fair comparison between different SLAM algorithms, the authors of [Mur-Artal et al., 2015] propose to run each sequence several times and to report the median over all runs. However we argue that for future works it may be more appropriate to report the mean and the standard deviation over multiple runs to be able to correctly compare different algorithms. Furthermore the stability of a SLAM system is rather important to guarantee good performances. Finally, we argue the ground truth accuracy should also be taken into account when comparing algorithms as gains below this threshold can not be deemed significant.

Additional information about the evaluation of SLAM systems can be found in [Z. Zhang and Scaramuzza, 2018]. For all our experiments in this manuscript we will use the well known evo library by Michael Grupp (https://github.com/MichaelGrupp/evo) to compute all the metrics.

2.3 Semantic SLAM

2.3.1 Introduction

We introduced in the previous section a typical state of the art of SLAM systems, such as ORB-SLAM2 [Mur-Artal and Tardós, 2017] that is able to accurately localize a camera in small to large scenes, indoors and outdoors and to create a map of its environment. However it is still limited in challenging cases that include:

Figure 2.8 – Same place, different seasons from [Stenborg et al., 2018]

.

Robust relocalization and loop closure.

Image retrieval systems based on descriptors such as [Gálvez-López and Tardos, 2012] are limited by their lack of invariance. Variability can come from different viewpoints, if two trajectories are perpendicular or have opposite directions for example. It can also come from different timestamps. Indeed two images taken during different seasons may be visually very different and contain few matching descriptors as can be seen in the figure 2.8. Indeed, some objects can also visually change over time, such as cars or vegetation. This makes the process of image retrieval harder.

Dynamic objects handling.

Most classical SLAM systems assume the scene to be rigid as the pose of the camera is computed using all visible map points. Hence when an object moves in the environment, the reprojection of its points will not be coherent with the rest of the scene. This can corrupt camera pose estimation. Furthermore map points attached to a moving object can be triangulated at different positions in space, creating a phantom of the object in the map. This assumption is called the static scene assumption and severely limits the scenarios in which a SLAM system can be used.

Scale drift.

When using a monocular SLAM system the scale of the scene is unknown. Its estimation can thus drift through time. Loop closure can be used to solve this problem with the obvious limitation of requiring a loop, which may not happen regularly. An example of scale drift can be seen in figure 2.6.

Furthermore the map built by a classical SLAM system only contains low level appearance and geometrical information and lacks semantic meaning. This additional information can be useful for some applications [Cadena et al., 2016]. For example mobile robots should understand semantic concepts (such as a cup of coffee) in order to understand the world, safely navigate within it (going to the kitchen without bumping into someone) and interact with it (grabbing the cup of coffee). Augmented reality can as well make use of a semantic map, to enable applications that can virtually interact with the scene to create a more realistic experience, as shown in [Runz et al., 2018]. To summarize, in classical SLAM systems there is a semantic gap between the representation of the world that is rather low level and the real world in which the system evolves. Thus, new higher level representations must be developed.

Moreover a semantic understanding of the world can make the SLAM more accurate and more robust towards complex environments. It may even enable the SLAM in scenarios in which it can not currently work [Cadena et al., 2016].

In the following section we show how semantic information can be injected into SLAM to answer the previously cited problems.

2.3.2 Semantic Mapping

The goal of semantic mapping is to build a map in which the semantic class of entities is known. There are mainly two ways to do semantic mapping depending on the dimension of the input data that is considered. Indeed a semantic map can be created using 2D segmented images of a scene [Hermans et al., 2014; Kundu et al., 2014; X. Li and Belaroussi, 2016; McCormac et al., 2017; Pham, Hua, et al., 2019; Runz et al., 2018; Stückler and

Behnke, 2014; Xiang and Fox, 2017; C. Yu et al., 2018; C. Zhao et al., 2017]. Another approach to obtain a semantic map consists in directly applying a 3D semantic segmentation algorithm to the map (represented by a point cloud, a mesh or voxels for example) [Dai and Nießner, 2018; Dai et al., 2017; Landrieu and Simonovsky, 2018; Y. Li, Bu, et al., 2018; Pham, Nguyen, et al., 2019; Qi, Su, et al., 2017; Qi, Yi, et al., 2017; Tatarchenko et al., 2018; W. Wu et al., 2019]. However those approaches require a denser point cloud than what is produced by sparse SLAM systems such as ORB-SLAM 2. Moreover the segmentation can be computed only once the point cloud is large enough, which goes against the online nature of a SLAM system. For those reasons we will focus on the first type of approaches. It should be noted that some SLAM systems use 3D LiDAR scans as input data. For those approaches CNNs have been developed to segment the point clouds [Aygun et al., 2021; Bogoslavskyi and Stachniss, 2016; Milioto et al., 2019; 2020]. We do not cover those approaches in this chapter as we focus on vision based SLAM.

The simplest way to create a semantic map is to fuse 2D segmented keyframes obtained using a CNN like [K. He et al., 2017; Kirillov et al., 2019a; Redmon et al., 2016] or more recently [Hong et al., 2021; Y. Li et al., 2021] to obtain the probability distribution of 3D points. Surveys about image semantic segmentation can be found in [Garcia-Garcia et al., 2018; Guo et al., 2018; Siam et al., 2017; Thoma, 2016; B. Zhao et al., 2017; H. Zhu et al., 2016]. We focus here on CNN based approaches, however other older approaches, often based on random forests or conditional random fields such as [Hermans et al., 2014; Sengupta et al., 2013; Stückler and Behnke, 2014; Valentin et al., 2013; Vineet et al., 2015] also exist and have proven to be reliable in the past.

SemanticFusion [McCormac et al., 2017] is one of the most cited SLAM that can create a consistent semantic map in real time using CNNs. Their work is based on ElasticFusion [Whelan et al., 2015] which creates a dense map from RGB-D images and yields the camera pose in real time. They use a CNN to segment RGB-D frames, producing a probability map for each pixel. The probabilities are then fused over time using a recursive Bayesian update to obtain the probability distribution of each surfel in the map. A Conditional Random Field (CRF) optimization is then done to refine the segmentation, using depth and appearance information. SemanticFusion shows that fusing information in a SLAM system improves the segmentation accuracy. The improvement brought by the CRF is negligible while the optimization is computationally expensive. They also show that segmenting every frame brings little accuracy while adding computational burden. An example of a dense RGB and semantic map obtained by [McCormac et al., 2017] is

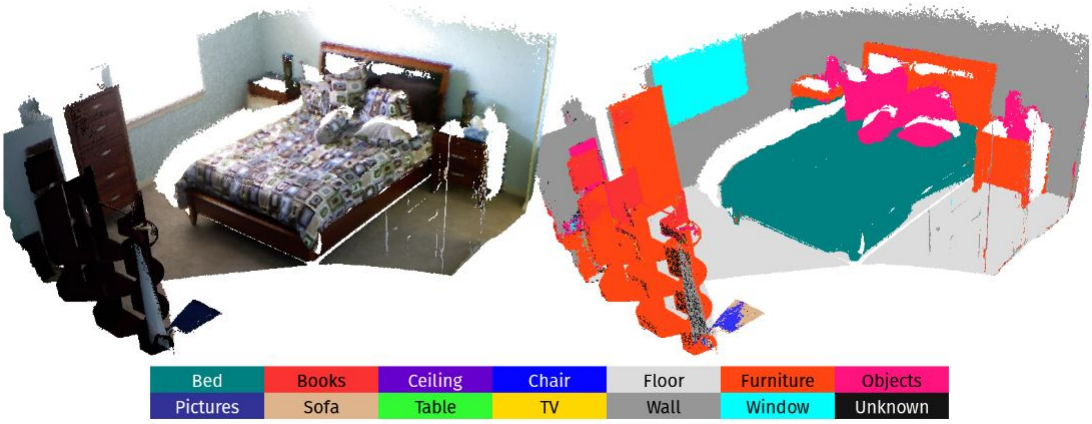

Figure 2.9 – Example of 3D semantic map from [McCormac et al., 2017]

visible figure 2.9. [X. Li and Belaroussi, 2016; C. Yu et al., 2018] are very similar to [McCormac et al., 2017] but [X. Li and Belaroussi, 2016] is built upon a semi-dense SLAM system [Engel et al., 2014] and [C. Yu et al., 2018] uses an octree as map representation.

Recently the framework Kimera [Rosinol et al., 2020] proposed a new open source SLAM system that can build a dense semantic map using the approach of [McCormac et al., 2017].

[Pham, Hua, et al., 2019] has a similar approach, namely, a 2D segmentation using a CNN followed by CRF optimization on the reconstructed semantic map. However they propose a super-voxel cluster approach to create clusters of voxels and apply the CRF on the super voxels, making it lightweight. Furthermore they tackle the problem of instance segmentation, by predicting each object id with the CRF. The higher order constraints of the CRF allow them to improve results over [McCormac et al., 2017].

Those approaches often rely on CRF optimization to refine the segmentation using geometrical information. To avoid this additional post processing step, [C. Zhao et al., 2017] proposes to integrate the CRF constraints in an end to end trainable network to improve genericity and computation time.

[Sünderhauf et al., 2017] uses a 2D object detection network (SSD [W. Liu et al., 2016]) to detect objects in RGB images. The depth image is then over-segmented in an unsupervised manner based on the geometry of objects. Finally segments are associated to detected objects and to obtain semantic segments with a low computational cost. MaskFusion [Runz et al., 2018] uses Mask-RCNN [K. He et al., 2017] to segment objects in RGB frames. The associated depth map is used to refine the segmentation. To do so geometric

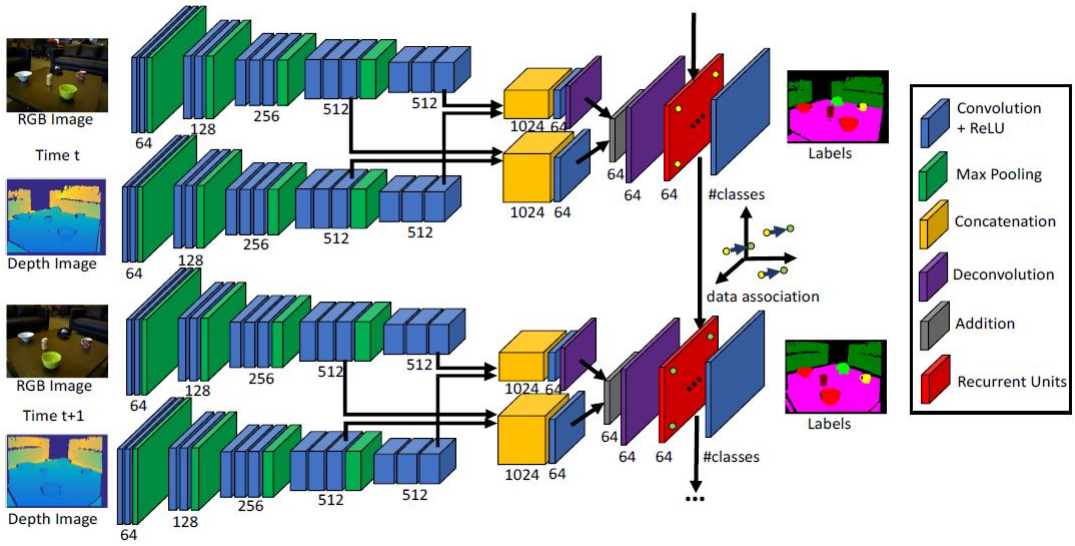

Figure 2.10 – Pipeline of [Xiang and Fox, 2017], features are extracted from RGB and depth images and used with current hidden state for segmentation and hidden state update.

segmentation is performed in real-time on the GPU, grouping close and concave parts of the depth map. This segmentation is then fused with the semantic segmentation, refining object boundaries.

Most of the approaches described above use a simple method to fuse individual segmentations from single frames in a coherent manner (usually by averaging or bayesian fusion). DA-RNN [Xiang and Fox, 2017] proposes a new way to fuse CNN outputs that are highly correlated as cameras navigate smoothly in the scene. They present a new type of recurrent neural networks, called data associated recurrent neural networks (DA-RNN) that are composed of data associated recurrent units. They use a fully convolutional neural network (FCN [Long et al., 2015]) that extracts features from an image and upsample them to generate a per-pixel classification. They introduce a layer made of recurrent units (one per pixel) before the classification step to take into account information from the previous time steps at the corresponding pixel. Doing so they show that they can coherently segment a video stream and obtain significantly better results than the state of the art. Their pipeline can be found in figure 2.10.

2.3.3 Semantic data for relocalization

Classical keypoints descriptors like ORB [Rublee et al., 2011] or SIFT [Lowe, 1999] are not highly invariant towards changes in illumination. Thus we can not use such keypoints to find matches between images taken some time apart. Moreover the static scene assumption may not be true for long term relocalization, for example cars in the scene can move which may result in incorrect data association and wrong pose estimation. Semantic data has been widely used for long term relocalization due to the invariance brought by segmentation networks either using classical keypoint extractors with semantic information [Castaldo et al., 2015; Larsson, Stenborg, Toft, et al., 2019; Lianos et al., 2018; Liao et al., 2020; W.-C. Ma et al., 2019; T. Shi et al., 2019; Singh and Košecká, 2016; Stenborg et al., 2018; Toft et al., 2017; 2018; X. Yu et al., 2018] or using deep learning [Garg et al., 2017; 2018; Mousavian et al., 2015; Naseer et al., 2017; Radwan et al., 2018; Schönberger et al., 2018; Seymour et al., 2019; Taira et al., 2018; Valada et al., 2018; Voodarla et al., 2021] with datasets such as [Larsson, Stenborg, Hammarstrand, et al., 2019; Toft et al., 2020]. We focus here on the use of semantic information for relocalization, however a more complete state of the art can be found in [Duong, 2019; Masone and Caputo, 2021]. [Toft et al., 2017] is one of the first work to use only semantic segmentation for relocalization. They propose to create a semantic map of points and lines and to relocalize the camera by optimizing a semantic reprojection function. This function consists in measuring the 2D distance between a reprojected map point and the closest semantic zone with the same class. A pose is thus considered optimal if all map points are reprojected into their corresponding semantic class. The experiments show that this approach is less accurate than classical keypoints in simple cases with only luminosity changes, however it is more robust in challenging cases with large seasonal changes. [Lianos et al., 2018] works in a similar way but used this approach to improve camera tracking instead of relocalization. Semantic keypoints are used to obtain medium term matches that are visually too different to be found by classical descriptors. Camera pose is then optimized using both classical keypoints and semantic keypoints, which reduces the drift. Their experiments show an improvement of the keyframe pose accuracy and a reduction of the scale drift. [Toft et al., 2018] approach is similar. For each point a set of pose hypothesis is created by swiping over possible poses. For each pose a semantic consistency score is computed by counting the number of points that reproject in their corresponding area. The best score of each point is then used to weight a PnP RANSAC which allows to robustly recover the pose. The pipeline of [Toft et al., 2018] is visible in figure 2.11.

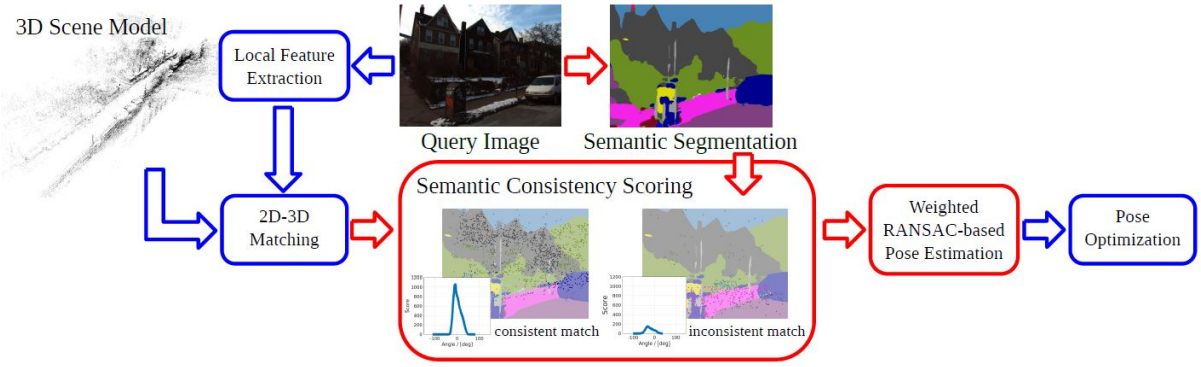

Figure 2.11 – Relocalization pipeline from [Toft et al., 2018].

[Stenborg et al., 2018] proposes a filter based approach for long term relocalization, with high seasonal changes where classical descriptors such as SIFT [Lowe, 1999] fail. Similarly to other approaches they reproject 3D semantic points in segmented images to obtain a score. This score is then used to weight particles in a particle filter and recover the pose.

We can easily see that those works do not really make use of the semantic information but rather of the invariance brought by segmentation networks. [Larsson, Stenborg, Toft, et al., 2019] shows that those networks can be trained in an unsupervised manner to increase the number of classes without semantic meaning. This makes the network more discriminant and the relocalization based on the reprojection of semantic keypoints more precise. [Arandjelović and Zisserman, 2014] and [Kobyshev et al., 2014] are similar works that use semantic information to make local keypoints more discriminant and more robust towards changes in viewpoint. Keypoints are matched only if their local semantic information matches. This can as well accelerate the matching of keypoints as less comparisons need to be made. Additionally [Kobyshev et al., 2014] proposes to predict correct matches using a random forest approach on semantic features, which removes keypoints from unreliable objects such as cars or pedestrians and can further accelerate the matching process. [Castaldo et al., 2015] proposes to solve the problem of aerial-ground localization using an RGB image and a geographic information system (GIS). To do so they present a Semantic Segment Layout that encodes both semantic information and its spatial layout. However those kind of approaches are rather specific and out of the scope of this paper. Some approaches are tailored to specific applications such as vehicle localization [Y. Hou et al., 2017; Liao et al., 2020; W.-C. Ma et al., 2019; Voodarla et al., 2021] which allows them to reduce the size of the map. For example [W.-C. Ma et al.,

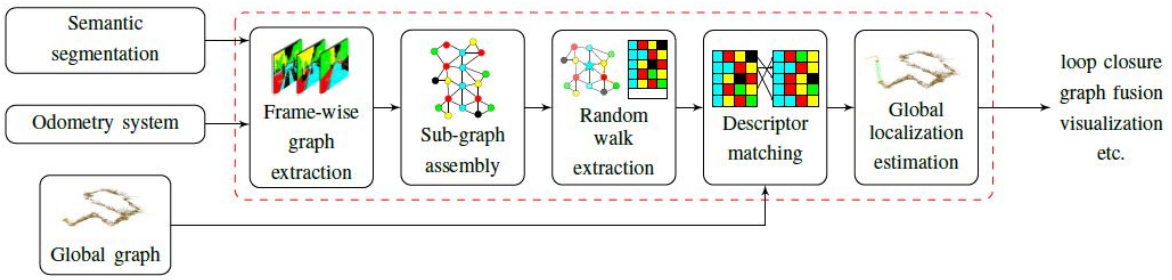

Figure 2.12 – Illustration of the pipeline of [Gawel et al., 2018]

2019] uses lanes and traffic signs, combined with other sensors (such as IMU and GPS) to predict the 3 DoF pose of a car using an histogram filter. [W.-C. Ma et al., 2017] proposes an original approach by using the sun direction and specific semantic cues for vehicles (such as intersections or road type) to localize a car. To estimate the sun direction they use a specific CNN on a single image, which is then used in a bayesian filter to estimate the pose. Semantic information can as well be used to detect discriminant structures within the scene. [Mousavian et al., 2015] uses a semantic segmentation network to detect buildings and creates a Bag Of Visual Words [Gálvez-López and Tardos, 2012] using only man-made structures. This avoids using less discriminative structures such as trees, which furthermore change on the long term. X-View [Gawel et al., 2018] proposes to use semantic information to disambiguate global localization that can suffer from appearance changes and perceptual aliasing. To do so, they build a graph using semantic information. Vertices correspond to the 3D location and class of objects in the scenes while edges encode the 3D distance between pairs of objects. This allows them to obtain a high level representation of scene parts. Matching graphs for relocalization can then be done by computing a descriptor for each vertex corresponding to a set of random walks from this vertex. This approach allows them to efficiently represent the scene given any view point. Their pipeline can be seen in figure 2.12. As we can see in the pipeline the built graph is an efficient high level representation of the visited scene. On the other hand, inspired by the work of [Kendall et al., 2015] some approaches use CNNs to estimate the pose of a camera within a scene. [Radwan et al., 2018; Valada et al., 2018] propose a new CNN architecture to solve at once the problems of localization, odometry and semantic segmentation. Their pipeline, visible in figure 2.13 is composed of three parts. In the first one a couple of residual networks extract features from consecutive images to estimate their relative pose. In the second one only the current image is used to estimate its global pose. In the last one the networks segments the current image. The networks are trained

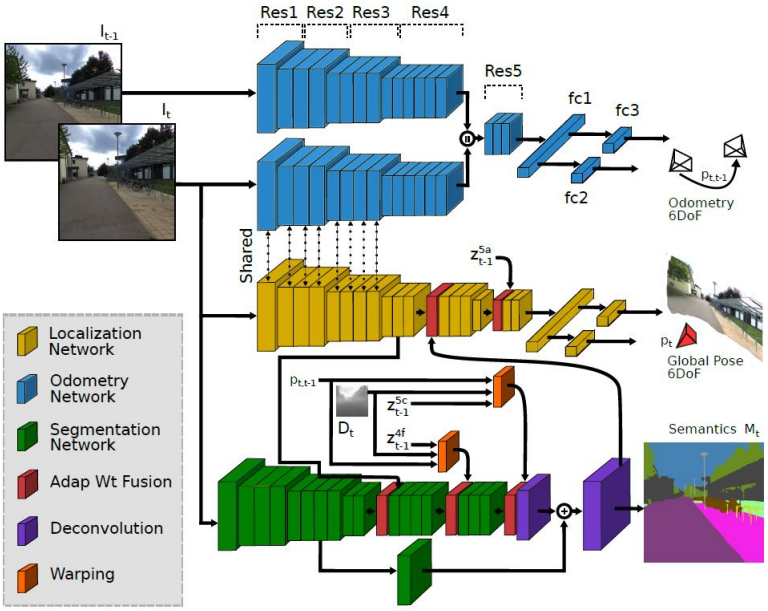

Figure 2.13 – Illustration of the pipeline of [Radwan et al., 2018]

together in a multitask fashion to exploit inter task dependencies. Furthermore [Radwan et al., 2018] encourages the reusing of features by fusing intermediate feature maps from the previous frame into the current frame feature maps. [P. Wang et al., 2018] render a semantic image by splatting points from a previously captured semantic map using a LiDAR. Then a CNN processes both the rendered semantic image and the RGB image from the camera to predict its pose which is smoothed by an LSTM network. The system re-renders the semantic image using the refined pose and an auto-encoder refines the segmentation. Their pipeline can be seen in figure 2.14. Finally some approaches tackle the problem of image retrieval as in [Arandjelovic et al., 2016; Jégou et al., 2010]. Their goal is to find the visually closest images in a database given a query image based on the assumption that visually close images should represent close points of view. The best image is then selected using local descriptors which discards geometrically incoherent solutions. A survey of such approaches for long term relocalization can be found in [Zaffar et al., 2019]. [Schönberger et al., 2018] uses a 3D CNN to encode a semantic map of voxels into a single vector in an embedded space that is then used for image retrieval. Their approach consists in training a Variational Encoder Decoder to complete semantic maps from incomplete semantic information. The network is composed of two parts: first the encoder takes as input the 3D voxel map and generates a latent vector encoding the observation, then the decoder recreates a completed version of the input map using the latent vector.

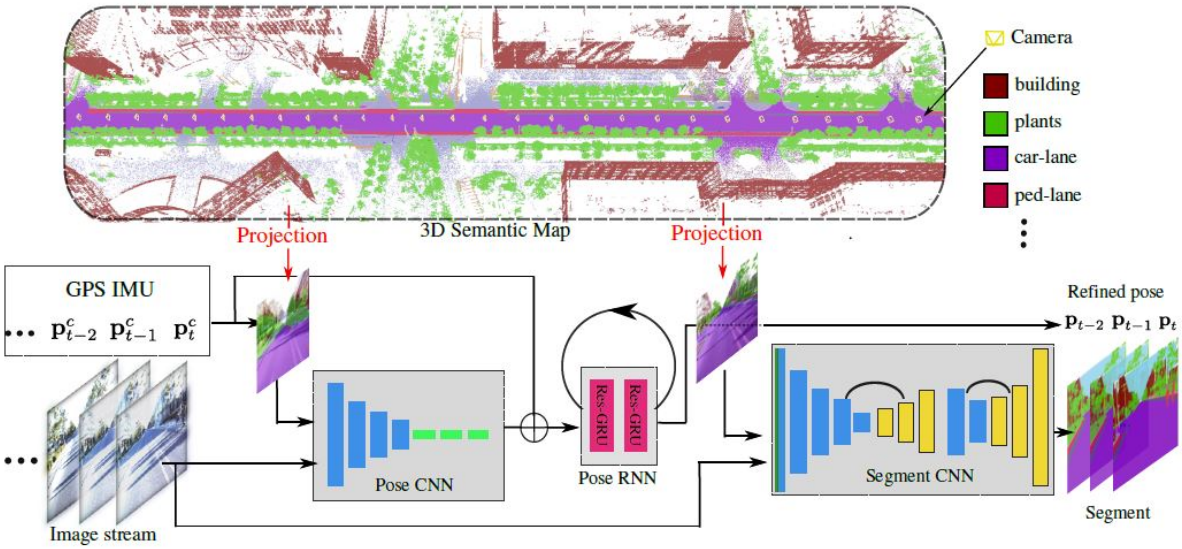

Figure 2.14 – Illustration of the pipeline of [P. Wang et al., 2018]

Once the encoder is trained it can generate latent vectors for local maps and a simple nearest neighbor approach is used to recover the maps closest from the query.

LoST [Garg et al., 2018] proposes a new type of image descriptor tailored for opposite viewpoints relocalization. First, images are segmented using a CNN and the obtained features are used to create a local semantic tensor, inspired from [Jégou et al., 2010] which represents the image. Given a query, a set of closest images are found using their local semantic tensor. Semantic keypoints, corresponding to the highest activation of each feature map are then extracted in the query and retrieved images and used to select the best match.[Garg et al., 2019] is extension of LoST. The same authors [Garg et al., 2017] use place categorization to improve place recognition. Place categorization is a classification problem which consists in predicting a descriptive class (such as kitchen or forest) given an image. Place categorization predictions from a CNN are merged from different times using a hidden Markov model to obtain a temporally coherent prediction. This prediction is used by looking for matches that exhibit the same estimated categories. [Seymour et al., 2019] fuses semantic and appearance features in a CNN. To do so they use $1x1$ convolutions to project features in the same space and add them, thus merging high and mid level representation which brings robustness towards visual changes. The features maps are then spatially pooled to obtain a single descriptive vector. Furthermore they propose to use semantic and appearance information to focus on stable parts of the im-

age. This is done by predicting attention maps which correspond to weight maps that are then multiplied with the features maps, allowing to increase the importance of some parts of the image in the finale vector. VLASE [X. Yu et al., 2018] proposes to modify VLAD [Jégou et al., 2010] with semantic features. Semantic edges, corresponding to pixels associated with class probabilities are extracted from the image and used to build a VLAD descriptor. The closest descriptors using the cosine distance are then retrieved with the corresponding images. [Singh and Košecká, 2016] computes a semantic label descriptor to represent an image. To do so the image is segmented and divided with a grid. In each cell the semantic histogram is computed and their concatenation yields a vector that characterizes the spatial layout of semantic information. This vector can be used for image retrieval, associated to a Bag Of Words approach [Gálvez-López and Tardos, 2012].

Luminosity is not the only thing that changes overtime, some objects move as well, for example cars or pedestrians. Thus keypoints extracted from those objects can not be used for relocalization as they may create wrong matches leading to a deterioration of the pose estimation. We present here some research papers that focus exclusively on the filtering of dynamic objects for relocalization, other dynamic SLAM systems are presented in the following section. Similarly to [Mousavian et al., 2015], [Naseer et al., 2017] uses a CNN to segment buildings from RGB images in order to focus on the most static parts of the image as can be seen figure 2.15. The features extracted by the CNN on stable parts of the image are then combined with features from the whole image to create an image descriptor. As the emphasis is put on stable parts of the image this descriptor is more robust towards changes in the image. [Taira et al., 2019] is based on InLoc [Taira et al., 2018] for indoor visual localization. In this work they use semantic information to mask out highly dynamic objects such as people.

2.3.4 Semantic data for dynamic SLAM

Classical SLAM systems are built on the hypothesis that the world is mainly static. Some small objects that move are treated as outliers by robust cost functions. However as soon as the moving keypoints are not negligible the map gets corrupted and the camera pose estimation deteriorates. Some systems propose to use semantic information to help detecting dynamic objects within the scene [Ballester et al., 2021; Bescos et al., 2018; X. Chen et al., 2019; Fan et al., 2020; Hachiuma et al., 2020; Han and Xi, 2020; Henein et al., 2020; Huang et al., 2019; 2020; Kaneko et al., 2018; P. Li, Qin, et al., 2018; Y. Liu and

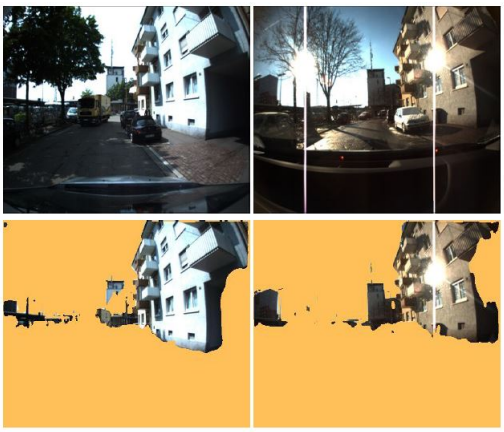

Figure 2.15 – Example of images *(top)* and stable segmentation *(bottom)* from [Naseer et al., 2017]

Miura, 2021; Luiten et al., 2020; J. Peng et al., 2019; Schorghuber et al., 2019; Vincent et al., 2020; Wong et al., 2021; Xiao et al., 2019; H. Xu et al., 2019; S. Yang and Scherer, 2019a; C. Yu et al., 2018; J. Zhang et al., 2020; T. Zhang and Nakamura, 2018]. There are two ways to deal with moving objects: on the one hand moving objects can be detected and filtered out of the image, to use only keypoints extracted from static parts of the image, thus coming back to the classical SLAM formulation [Ballester et al., 2021; Bescos et al., 2018; X. Chen et al., 2019; Fan et al., 2020; Han and Xi, 2020; Kaneko et al., 2018; Y. Liu and Miura, 2021; Luiten et al., 2020; Schorghuber et al., 2019; Vincent et al., 2020; H. Xu et al., 2019; C. Yu et al., 2018; T. Zhang and Nakamura, 2018]. On the other hand moving objects can be tracked and integrated within the SLAM system, which is more complex than simple camera tracking but allows the user to obtain information about moving objects, which can be interesting for autonomous vehicles for example [Bescos et al., 2021; Hachiuma et al., 2020; Henein et al., 2020; Huang et al., 2019; 2020; P. Li, Qin, et al., 2018; Wong et al., 2021; S. Yang and Scherer, 2019a; J. Zhang et al., 2020]. There are many ways to segment the image and detect dynamic parts within it, we focus here on the works that explicitly use semantic information, although some dynamic SLAM systems give very good results without such information [Huang et al., 2019; S. Li and Lee, 2017; Scona et al., 2018; Y. Sun et al., 2017].

Dynamic objects filtering

DS-SLAM [C. Yu et al., 2018] uses semantic information to mask out people in the images as they are likely to be dynamic. Furthermore they perform a moving consistency

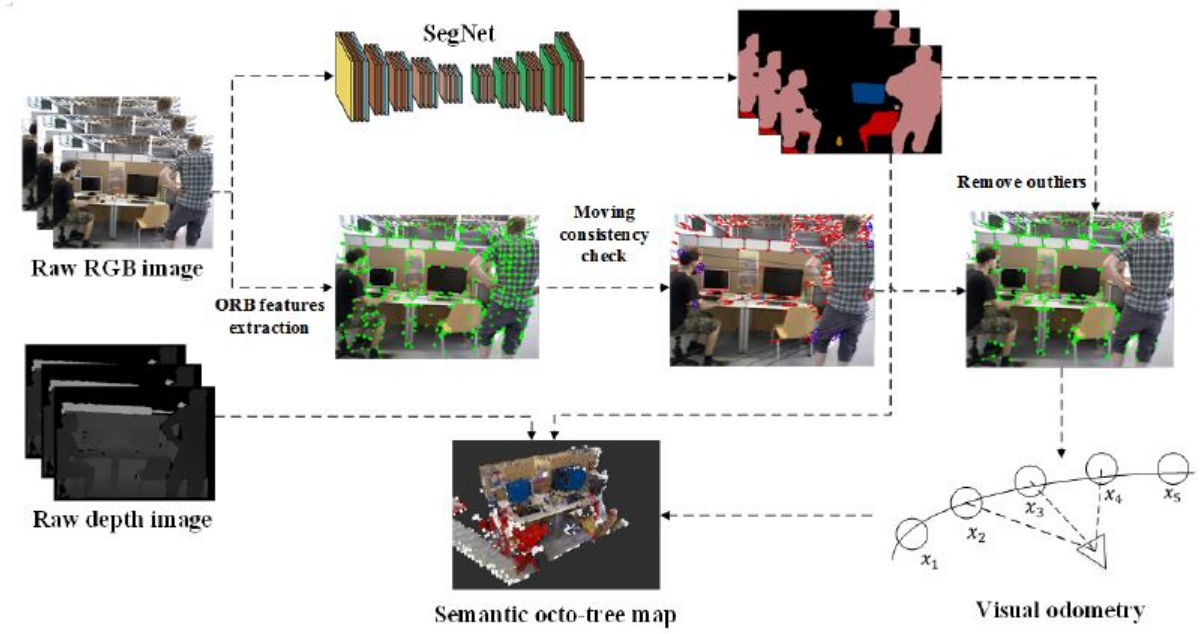

Figure 2.16 – Illustration of the pipeline of the dynamic SLAM [C. Yu et al., 2018]. As we can see keypoints on dynamic people are removed after the geometric check.

check based on epipolar geometry on the keypoints in each segmented area , which allows them to deduce which objects are moving. More precisely the main idea is to compute the 2D distance in image space between each point and its corresponding epipolar line, that is computed with a fundamental matrix estimated in a RANSAC scheme. If the distance is more than some threshold the point is flagged as an outlier. If the number of outliers in a given segmented region is higher than a threshold the region is flagged as dynamic and all points within it are discarded. While this approach works it does not use semantic information but rather the segmentation of the image, moreover it assumes that a fundamental matrix can be accurately estimated and thus that the scene is mostly static. The pipeline of [C. Yu et al., 2018] is visible in figure 2.16. SOF-SLAM [Cui and Ma, 2019] has a similar approach: each frame is segmented and dynamic or potentially dynamic objects are masked out to robustly estimate the fundamental matrix between the previous and current frame. Then the distance between each potentially dynamic keypoint and its corresponding epipolar line is computed to determine whether it is static or dynamic. The approach of [Han and Xi, 2020] is similar, except that points in a priori dynamic objects are removed and an homography matrix is then computed using remaining points which are discarded if their optical flow is larger than a threshold. [Fan et al., 2020] proposes a

comparable approach but an additional cleaning step using depth information is applied to refine object masks. [Bescos et al., 2018] proposes to use an object segmentation network to detect and segment objects in frames. Objects are classified into two categories: static or a priori dynamic (e.g. cars, pedestrians). All a priori dynamic objects are then removed from the image which is used in a SLAM framework. This can lead to a deterioration of the pose estimation as a priori dynamic objects can as well be static (e.g. a parked car). To detect non-segmented objects they can as well check the depth consistency. However this approach does not work for far or small objects. In their experiments [Bescos et al., 2018] show that their approach improves camera pose estimation in highly dynamic scenes. However in static scenes removing keypoints can hurt the pose accuracy. To counter that, SLAMANTIC [Schorghuber et al., 2019] proposes to compute a confidence value for each map point depending on both the semantic class and the number of correct observations. During tracking high confidence points are used to obtain a first pose estimation, then using this estimation the reprojection error of less confident map points is computed to validate or invalidate them. Finally the pose is refined using all static and valid points. The approach of [Brasch et al., 2018] is similar as semantic information is used as a prior for points staticity and the observation of a point during time can make a point static or dynamic. DOT [Ballester et al., 2021] detects truly dynamic objects. Their system segments potentially dynamic objects, the static parts being used for camera tracking. Then for each segmented object the reprojection error is computed densely. If the median of this error is higher than a given threshold then the object has moved and will not be used in the SLAM system. PoseFusion [T. Zhang and Nakamura, 2018] detects humans in scenes to remove them as they are mostly dynamic. To do so they use a human estimation network to get a set of human joints that are used to segment the 3D point cloud given by the RGB-D camera. These parts are then removed from the point cloud, the reminders are used in a classical dense SLAM system, Elastic Fusion [Whelan et al., 2015]. RDS-SLAM [Y. Liu and Miura, 2021] proposes a real time semantic SLAM masking out a priori dynamic objects but loses tracking accuracy.

Dynamic Objects tracking

Instead of simply ignoring dynamic objects some SLAM systems [Bârsan et al., 2018; Bescos et al., 2021; Hachiuma et al., 2020; Henein et al., 2020; Huang et al., 2019; 2020; Runz et al., 2018; Rünz and Agapito, 2017; B. Xu et al., 2019; S. Yang and Scherer, 2019a; J. Zhang et al., 2020] propose to track them, estimating their 6 DoF pose which

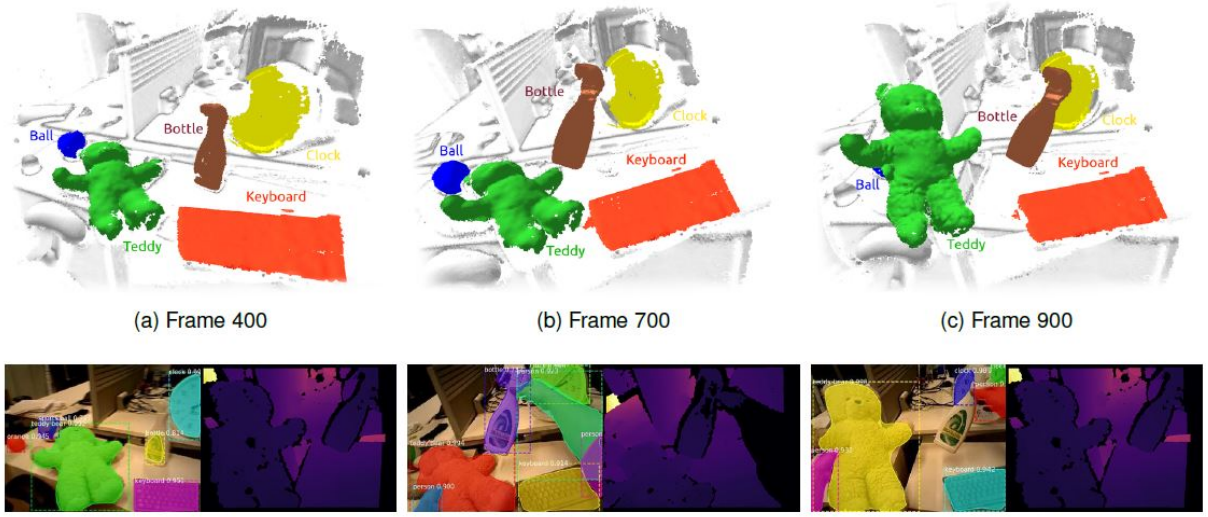

Figure 2.17 – Example of semantic maps produced by [Runz et al., 2018]. As we can see objects are successfully segmented and tracked in RGB-D images.

can be refined in the BA. These approaches are interesting because first, they estimate the pose of objects which can be useful for some applications such as autonomous driving. Second, they can as well improve camera pose estimation by adding constraints about the displacement of objects in the Bundle Adjustment. MaskFusion [Runz et al., 2018] tracks the camera as well as segmented objects in RGB-D frames by minimizing both a photometric and an ICP error. Depth is used to improve the semantic segmentation of Mask-RCNN [K. He et al., 2017]. Furthermore their approach allows to reconstruct the 3D objects. However their evaluation of SLAM only shows the impact of masking out people in the scene, similarly to [Bescos et al., 2018], and does not show the impact of tracking on the camera pose estimation. An example of map produced by MaskFusion is visible in figure 2.17.

MaskFusion is the continuity of CoFusion [Rünz and Agapito, 2017] which is very similar but can as well segment objects based on their motion using a CRF optimization. Pixels with similar motion and appearance are clustered together. MID-Fusion [B. Xu et al., 2019] relates to both those approaches as it is a dense object-level SLAM but uses a volumetric representation based on an octree to optimize space usage. The camera tracking is done by combining an ICP and photometric error. However the object pose estimation is parametrized in the object frame which improves the ICP convergence. Furthermore they combine motion segmentation as in [Rünz and Agapito, 2017] and semantic segmentation as in [Runz et al., 2018]. Their approach shows better tracking and recon-

struction results than previous approaches. EM-Fusion [Strecke and Stuckler, 2019] is a follow up work of [Rünz and Agapito, 2017] and [Runz et al., 2018]. In this paper they propose to track objects in a dense SLAM but use an Expectation-Maximization (EM) framework to estimate data association. Objects are segmented in the image using Mask-RCNN [K. He et al., 2017] and the probabilistic data association is computed using the Signed Distance Function (SDF) between the corresponding point and the object. Each object (including the background) is then tracked in a weighted minimization problem where the weight is the data association probability. This soft framework allows to better track objects moving in the scene. DetectFusion [Hachiuma et al., 2020] proposes a dense SLAM with object tracking as [Runz et al., 2018] but uses an object detector followed by geometrical segmentation and motion based segmentation to reach real-time capacities. DynaSLAM II [Bescos et al., 2021] is an extension of [Bescos et al., 2018] which main limit was that all a priori dynamic objects were masked out, even if they were static at the time. In their work they propose to track the camera using a priori static parts of the image as well as to track segmented objects. They tightly integrate the estimations by optimizing all keyframes poses, object poses, static map points and objects points in a Bundle Adjustment. CubeSLAM [S. Yang and Scherer, 2019a] uses an object detection network to estimate 2D bounding boxes and object classes in RGB images. For each detected object they estimate a set of 3D bounding boxes that tightly project into the 2D boxes. The edges of the 3D boxes and edges extracted from the RGB images are then compared to chose the best 3D bounding box. The estimations are then tightly integrated in a classical SLAM with a Bundle Adjustment. The BA enforces 3D bounding boxes consistency over frames, consistency with the 2D bounding boxes observations and with the object map points. If an object is moving its points are dynamically re-projected using the estimated object pose and a motion model ensures that the objects movement is smooth. [P. Li, Qin, et al., 2018] is similar to CubeSLAM but is limited to cars and uses a simpler approach for 3D bounding box estimation. However they do not tightly integrate the object and camera poses in a Bundle Adjustment, poses are refined separately. ClusterVO [Huang et al., 2020] follows the work of ClusterSLAM [Huang et al., 2019] to track general objects without imposing priors and using only an object detection network [Redmon et al., 2016]. At each frame the detection network outputs a set of bounding boxes that are associated with existing objects in the map. Then new object points are triangulated and associated to existing objects through a CRF optimization based on their 2D and 3D location. Finally the object poses, object points, world points and camera poses are tightly optimized

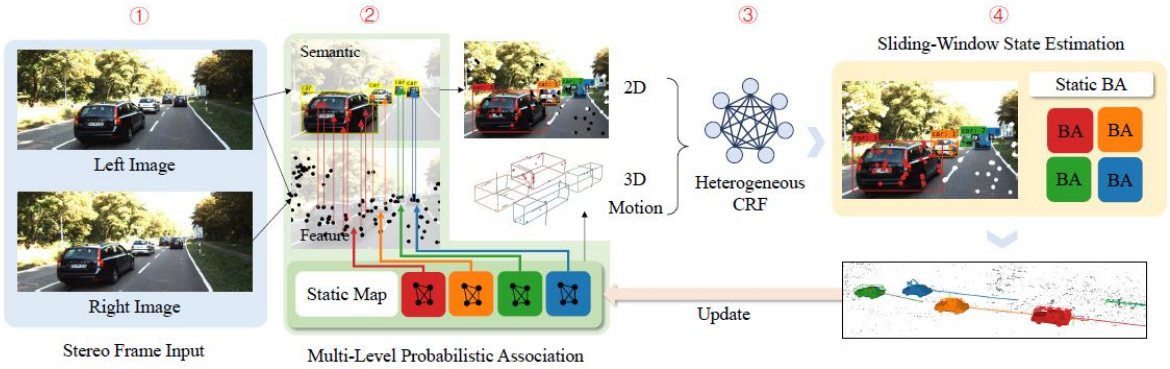

Figure 2.18 – Illustration of the pipeline of [Huang et al., 2020]. First is the data association step based on the speed of clusters, followed by the creation of new points associated to clusters as well as new clusters. At the end a BA is performed.

in a bundle adjustment, similarly to [Bescos et al., 2021] with a white noise acceleration prior. The pipeline of [Huang et al., 2020] is visible figure 2.18.

VDO-SLAM [J. Zhang et al., 2020] also tracks dynamic objects using RGB-D images and instance segmentation. They argue that as objects usually occupy only a small portion of the image not many keypoints can be extracted using sparse approaches. To answer this problem they estimate optical flow for each object and use it to extract and match a large number of points within each object mask as can be seen in figure 2.19.

2.3.5 Semantic data for object based SLAM

The idea of object based SLAM systems is to use objects in the scene as landmarks, instead of geometric primitive like points for example. This allows the SLAM to create a higher level, more meaningful map than a purely geometric map, getting closer to a human perception of space. Object based SLAM systems can be divided in three parts: first are systems that use a priori known objects in the scene [Civera et al., 2011; Fioraio and Di Stefano, 2013; Himri et al., 2018; Ramadasan et al., 2015; Salas-Moreno et al., 2013; Zeng et al., 2018]. These approaches however require specific objects to be present in the scene as well as sepcific object detectors. Second are object based systems that use generic object detectors such as [W. Liu et al., 2016; Redmon et al., 2016] and generic object representations such as cubes, spheres or quadrics [Frey et al., 2019; D. Frost et al., 2018; Hosseinzadeh et al., 2018; 2019; J. Li et al., 2019; McCormac et al., 2018; Nicholson et al., 2018; Ok et al., 2019; Sünderhauf and Milford, 2017; S. Yang and Scherer, 2019a]

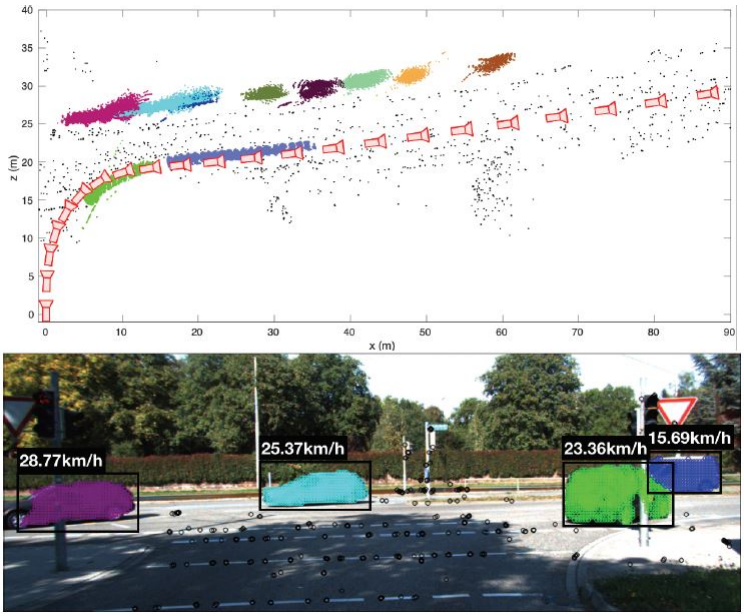

Figure 2.19 – Example of object tracking from VDO-SLAM [J. Zhang et al., 2020]. As we can see many points are extracted from segmented objects thanks to the optical flow.

or that seek to solve specific problems such as scale estimation [D. Frost et al., 2018; D. P. Frost et al., 2016; Knorr and Kurz, 2016; Sucar and Hayet, 2017; 2018] or probabilistic data association for objects [Atanasov et al., 2014; Bowman et al., 2017; K. Doherty et al., 2019; K. J. Doherty et al., 2020; Y. Wu et al., 2020; T. Zhang and Nakamura, 2018].

Using known models

[Civera et al., 2011] is one of the first monocular SLAM that uses a set of known objects in an EKF SLAM system. First a database of objects is built using a Structure From Motion (SFM [Schonberger and Frahm, 2016]) approach. When an object is recognized in the scene (with SURF matching [Bay et al., 2006]) its pose is estimated and 3D objects points are computed in the world frame and injected in the EKF state. This approach allows them to build a map containing dense 3D object models which enables robot interactions. Similarly [Fioraio and Di Stefano, 2013] detect objects with matching keypoints but use a more modern Bundle Adjustment based SLAM in which objects points reprojection error is minimized. Their approach can work with both 2D (as sets of images) and 3D object models. [Salas-Moreno et al., 2013] use a pre-trained object detection module to estimate the 6 DoF pose of objects visible in RGB-D images. Each detected object is then rendered using its 3D model. The camera pose can then be estimated with an ICP minimization

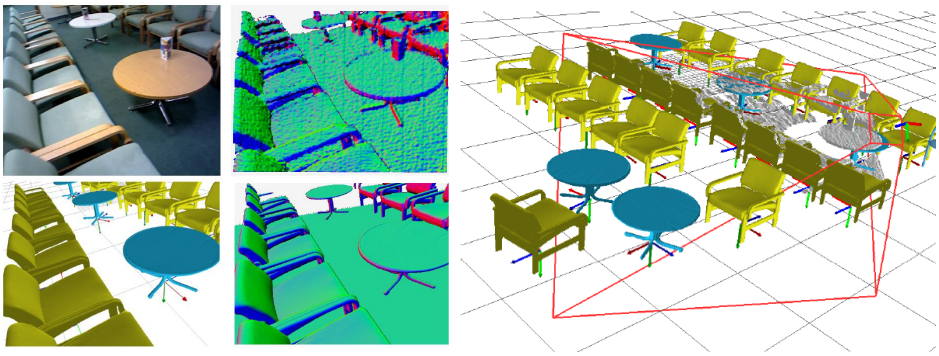

Figure 2.20 – Illustration of SLAM++ [Salas-Moreno et al., 2013]

of the rendered models and the measured depth map using both depth and normals. The system then refines all the objects and cameras poses in a pose graph. Their approach enables seamless augmented reality applications with interaction of the virtual content with the real scene. [Himri et al., 2018] builds a SLAM system in the specific scenarios of UAV (Underwater Autonomous Vehicles). Their approach is similar to [Civera et al., 2011] as they detect specific objects using a data base and estimate their pose with an ICP optimization. The estimated poses are then integrated in an EKF SLAM to obtain camera pose and speed estimations.

Using generic object detectors

The main drawback of object based SLAM that use known models is that they need a specific object detector [Rad and Lepetit, 2017; Xiang et al., 2018] as well as objects to be present in the scene, which limits their applicability. This is the reason why this kind of approaches is rather rare. Specific object detectors can be replaced by generic ones, that do not estimate the pose of objects but rather their 2D location as well as 2D spatial extent under the form of a 2D bounding box. The idea is then to estimate a 3D landmark from multiple 2D bounding boxes seen from different view points. [Nicholson et al., 2018] proposes to generate quadrics from objects 2D detections. Their idea is to optimize the quadric parameters, camera pose and points 3D position such that the projection of the quadric tightly fit each 2D bounding box. While this does not clearly improve the state of the art results the generated map contains additional valuable information. Examples of quadrics can be seen in figure 2.22. [Sünderhauf and Milford, 2017] has a similar approach. [Hosseinzadeh et al., 2018] has a similar approach but proposes the incremental optimization of closed quadrics, that have a peculiar form. Furthermore they detect planes in the

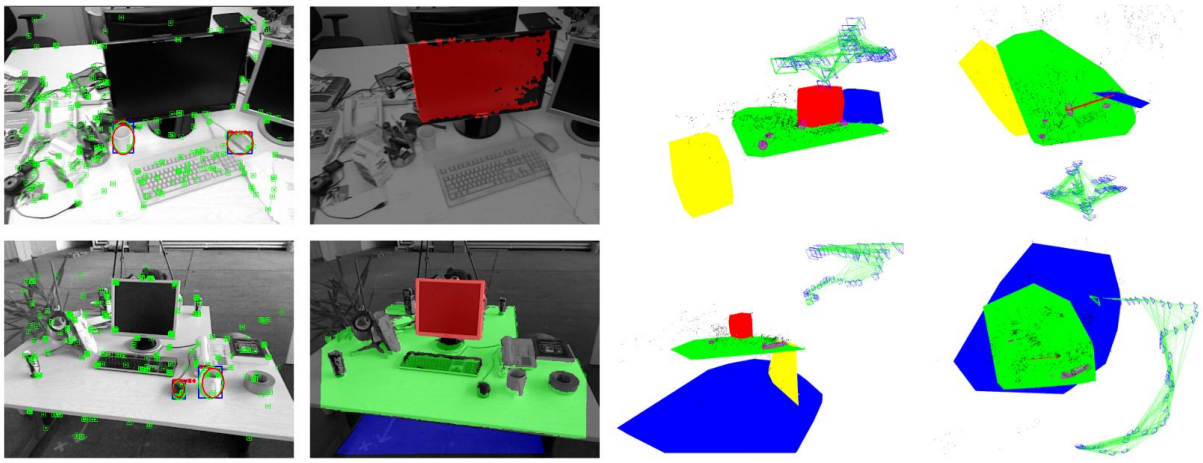

Figure 2.21 – Example of maps with quadrics and planes from [Hosseinzadeh et al., 2018]

scene and impose constraints between quadrics, planes, points and cameras to improve mapping and pose estimation. Example of maps with quadrics and planes can be seen in figure 2.21.

[Hosseinzadeh et al., 2019] is an improvement of [Hosseinzadeh et al., 2018]. In this paper they propose to use only monocular cameras for plane estimation, use a real-time object detector [Redmon et al., 2016] for faster inference and use a CNN to estimate point clouds for detected objects which is used to impose a shape prior on the quadrics. Some approaches [Gaudillière et al., 2019; 2020; Zins et al., 2022] propose to model objects as quadrics to improve relocalization, particularly in challenging cases with repetitive patters or large viewpoint variations. [Gaudillière et al., 2019] proposes a closed form solution to compute the translation of a camera using only a set of detected objects and quadrics in the map. [Zins et al., 2022] proposes to estimate the parameters of quadrics using a learning based approach which improves the accuracy of camera pose estimation.

CubeSLAM [S. Yang and Scherer, 2019a] proposes to use 3D bounding boxes. From a single RGB image and a set of 2D object detections an algorithm generates one 3D bounding box per object such as the projection of the 3D box tightly fits in the 2D box. The Bundle Adjustment minimizes as well this reprojection error to combine all the single 3D boxes estimations. This approach allows [S. Yang and Scherer, 2019a] to create a more meaningful semantic map, reduce the scale drift in the monocular case, handle pure rotations as well as dynamic objects (see section 2.3.4). [S. Yang and Scherer, 2019b] is an extension of their work using planes. [Tschopp et al., 2021] proposes to use superquadrics to represent objects with more complete shapes than simple quadrics. [Ok et al., 2019]

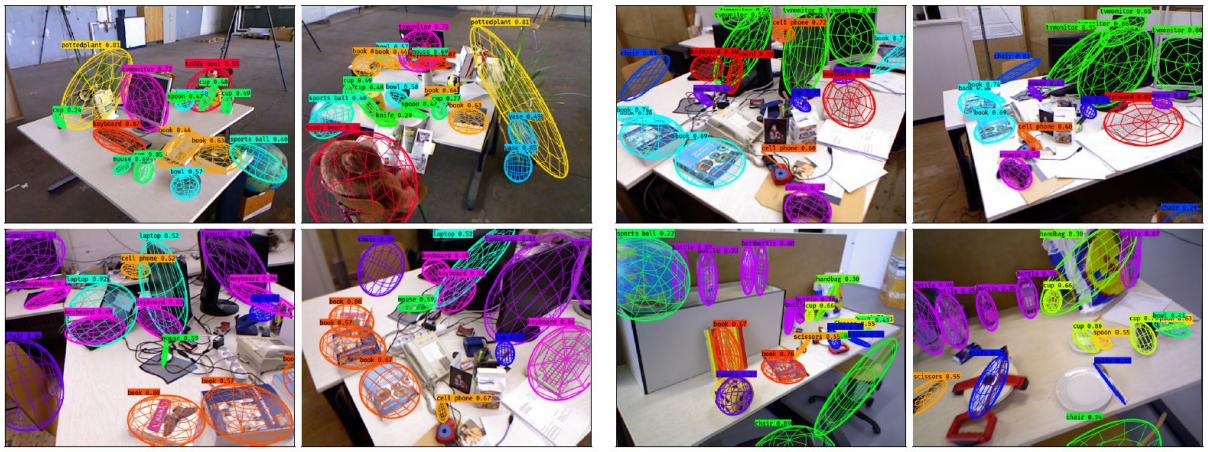

Figure 2.22 – Example of object quadrics build by [Nicholson et al., 2018]

Figure 2.23 – Example of reconstructed objects built by [McCormac et al., 2018].

uses as well ellipsoids to represent detected objects, however they also use texture and semantic information to better fit the quadrics. They use texture information to estimate a plane in the object that constrains the quadric along one dimension. Furthermore the shape of each object is constrained by its semantic class.

Fusion++ [McCormac et al., 2018] is different from other generic objects SLAM systems as it does not use a single model for all objects. Indeed each object segmented by Mask-RCNN [K. He et al., 2017] is densely reconstructed in an object level map using a TSDF [Izadi et al., 2011] adapted to the object size. The camera is tracked by minimizing the ICP error between the map and the observed depth image. Finally the pose graph containing objects and camera poses is optimized, yielding refined pose estimations. The objects are accurately reconstructed however the camera pose accuracy is not as good as [Mur-Artal and Tardós, 2017; Whelan et al., 2015]. Example of reconstructed objects and pose graph can be seen in figure 2.23.

Some works [Bowman et al., 2017; K. Doherty et al., 2019; K. J. Doherty et al., 2020; Y. Wu et al., 2020; T. Zhang and Nakamura, 2018] specifically tackle the problem of data

association for detected objects. Indeed object detectors allow to use objects as high level landmarks, however they may exist multiple objects of the same class in the scene, thus associating a landmark to a detected object becomes a crucial problem. However this problem is complex as data association is discrete while pose estimation is continuous. [Atanasov et al., 2014] is the first to take into account semantic information with data association. They propose to use random finite sets to represent the detected objects, which allows them to model a random number of objects, missed detections and false alarms. However computing the likelihood of a random finite set is expensive, thus they show as well that it can be approximated by computing the permanent of a specific matrix. This allows them to incorporate semantic data association in a particle filter. [Bowman et al., 2017] proposes to solve the problem of estimating poses, objects and point positions with the most probable data association using an expectation maximization approach. They show that as object landmarks are sparse and distinguishable, loop closure detection can be improved.

One of the main downside of monocular SLAM systems is that they can not access the scale of the scene which makes them subject to the well known scale drift problem. Furthermore as the scale is unknown, meaningful augmented reality applications are not possible without the user intervention. Some works [D. Frost et al., 2018; D. P. Frost et al., 2016; Knorr and Kurz, 2016; Sucar and Hayet, 2017; 2018] propose to use objects in the scene to estimate the scale factor. [D. Frost et al., 2018; D. P. Frost et al., 2016] represents objects as sphere, i.e. a 3D position in space and a radius corresponding to the object spatial extent. By minimizing the reprojection of spheres in 2D bounding boxes and assuming that the spatial extent of objects is known and fixed they ensure that the scale factor does not drift. [Sucar and Hayet, 2018] formulates the problem of scale estimation as a bayesian estimation using object bounding boxes and a priori sizes. To do so they model both the evolution of the scale factor and the likelihood of detections given the scale factor, the map and the object size. This approach effectively reduces scale drift however it is based on the assumption that objects are oriented vertically and that the detected surface is parallel to the vertical direction, which limits the applicability. [Knorr and Kurz, 2016] uses human face detection to estimate the absolute scale of the scene.

Using a neural network as a generic object representation

Some very recent work propose to use neural networks to represent objects, which is often designated as reconstruction with shape priors. NodeSLAM [Sucar et al., 2020]

proposes to train a Variational Auto-Encoder (VAE) to compress object geometries into a latent code using a 3D CNN. The latent code is then used to generate the object geometry as a set of voxels. Using a single RGB-D image, NodeSLAM optimizes the pose and latent code corresponding to detected objects, so that their estimated geometry fits the depth image. This reconstruction pipeline is integrated into a SLAM system. The camera is tracked by forcing object poses to be consistent across frames, similarly to what is done in classical object SLAM systems. However contrary to classical SLAM system this approach can work with all objects from a category instead of requiring a specific object pose estimation algorithm. DSP-SLAM [J. Wang et al., 2021] goes further than NodeSLAM by integrating the reconstruction pipeline into a classical sparse SLAM system. It uses DeepSDF [Do et al., 2018] as an implicit representation of objects. The latent code representing objects is optimized to fit 3D points coming either from the SLAM map or from a LidAR scanner. Then the code is fixed and used to estimate the pose of objects at each frame, allowing the SLAM to use objects as high level landmarks. We can also cite FroDO [Runz et al., 2020], which inspired DSP-SLAM. They use a sequence of RGB images to estimate a latent code representing objects. Latent codes are then fused and refined together with the object pose, using additional photometric, geometric and silhouette constraints. This allows them to accurately reconstruct objects, however they do not use objects as a higher level constrain to improve the pose estimation accuracy. Furthermore, contrary to classical SLAM systems their approach is not sequential and not real-time.

All those approaches have the interesting property of yielding in real time complete watertight meshes that closely correspond to reality.

2.4 Conclusion

In this chapter we first introduced a classical keyframe based SLAM system. Beginning with its roots from SfM and filters we introduced the problem of bundle adjustment and how to solve it. Then we built a modern SLAM system around the BA by first adding a front-end that pre-processes the images. We showed that decoupling tracking and mapping in two separate threads could enable a real time SLAM system that solved the BA on using a subset of keyframes. We added the blocks of initialization, relocalization and loop closure, that are not part of the core of a SLAM system but are nonetheless essential to build an accurate large-scale SLAM system. Next we introduced the metrics that are

commonly used to evaluate the quality of trajectories produced by a SLAM system. Finally we presented some limits of classical SLAMs, particularly the lack of high level semantic information in the map.

In the following part we studied how semantic information can be used in SLAM systems. First we presented papers that aim at building a semantic map using segmented images. Then we showed how semantic information can be used for relocalization, when classical approaches fail. While some papers truly use semantic information to represent the scene with higher semantic entities or to distinguish static objects others only use semantic information for the invariance brought by neural networks.

Then we studied how semantic information can be used to enable SLAM in dynamic cases as it is an indicator of objects dynamicity. A first approach consists in using only static classes for SLAM, however this approach can reduce the SLAM accuracy. Thus systems have been developed to detect if a priori dynamic objects are truly moving. Some SLAM systems have gone even further by proposing to use camera pose estimation to track moving objects in the scene.

Finally we presented papers that use objects in the scene as high level landmarks. Those approaches create entities that are closer to the human representation of the world which could enable applications in augmented reality or robotics. Those systems usually have to chose between generic and specific representations, however recent deep learning based approaches tend to close this gap.

Chapter 3

L6DNet: Light 6 DoF Network for Robust and Precise Object Pose Estimation with Small Datasets

Contents

In this chapter we present our object pose estimation algorithm, L6DNet. This kind of algorithms are necessary for object based SLAM systems that use specific objects in the scene as anchors. Doing so they can improve the accuracy of camera tracking and mapping while also creating a higher level map, enabling applications (e.g. A.R. applied on objects). However recent algorithms based on deep learning often require large amount of data, computational power and training time, which prohibits their use in real life scenarios. To solve this problem we propose a new algorithm, able to reach state-of-the-art performances while requiring little resources. To do so, we propose a hybrid, data driven strategy in two steps. First, two CNNs use patches extracted from objects to predict a set of points on the object surface. Second, predicted points are put in correspondence with a priori known points to compute the object 6 DoF pose. We show on a well known dataset that our approach yields results on par with state-of-the-art. A video explaining the approach and showing examples of object pose estimation can be fount at https://youtu.be/uuwjlyRxWtc.

3.1 Introduction

The goal of object pose estimation is to predict the rotation and position of an object relative to a known coordinate frame (usually the camera coordinate frame). This computer vision problem has many applications such as augmented reality or robotics. In the former case, it allows a realistic insertion of virtual objects in an image as described in [Marchand et al., 2016] and shown in Fig. 3.1. In the latter it can be used as an input for a robotic arm to grasp and manipulate the object such as in [C. Wang et al., 2019]. Let us recall that object pose estimation can also be used as an input to integrate semantic information into a SLAM system and improve both camera pose estimation and mapping [Civera et al., 2011; Fioraio and Di Stefano, 2013; Salas-Moreno et al., 2013]. Indeed if an object is present in a scene then its pose relative to the camera can be used as an additional constraint in the tracking and bundle adjustment cost function. Furthermore if the 3D model of the object is known, it can be used to constrain the map points, so that they lie on the 3D model. Finally, we can also make use of the additional pose information, for example by attaching virtual content to the object, enabling A.R. applications.

Although heavily studied, this problem is still relevant as it is unresolved due to its complexity. Indeed some scenes can be highly challenging due to the presence of cluttering, occlusions, changes in illumination, viewpoint, and textureless objects. Nowadays color and depth (RGB-D) sensors are smaller and cheaper than ever, making them relevant for object pose estimation. Indeed, compared to color-only (RGB) sensors, the depth channel provides relevant information for estimating the pose of textureless objects in dimly lit environments.

Classical object pose estimation approaches are either based on local descriptors followed by 2D-3D correspondences [Marchand et al., 2016], or on template matching [Hinterstoisser et al., 2012; Kehl et al., 2016; Tejani et al., 2014]. However the challenging cases listed above limit their performance. To address these limitations most recent methods solve the problem of object pose estimation with a data driven strategy using for example Convolutional Neural Networks (CNNs) [Y. He et al., 2020; Y. Li, Wang, et al., 2018; Z. Li et al., 2019; K. Park et al., 2019; S. Peng et al., 2019; Rad and Lepetit, 2017; Tekin et al., 2018; 2019; C. Wang et al., 2019; Xiang et al., 2018]. These approaches work in a holistic way, considering a whole RGB or RGB-D image as an input and making a single estimation of the pose. While some methods are hybrid, using a learning-based approach followed by a geometrical solver [Y. He et al., 2020; Z. Li et al., 2019; K. Park et al., 2019;

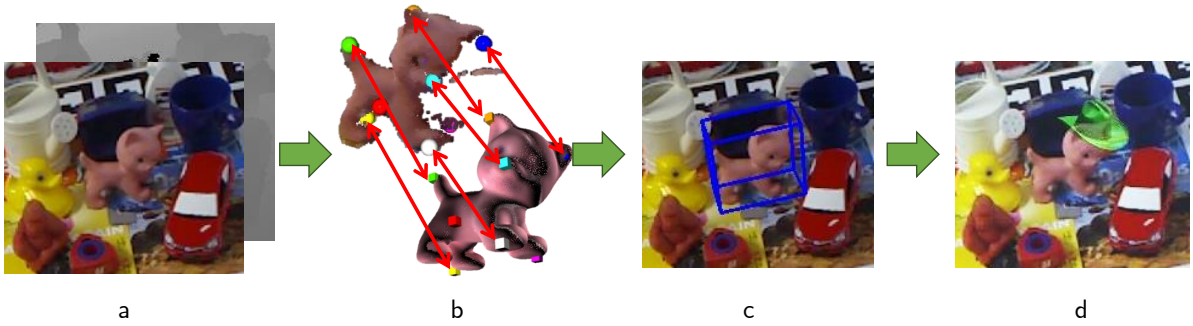

Figure 3.1 – To estimate the pose of an object we propose to use a single RGB-D image (a) to predict the position of a set of sparse 3D keypoints (shown as spheres in b) in the camera coordinate system, which are then registered in 3D with corresponding points in the world coordinate system (shown as cubes in b) to retrieve the pose (c) that can be used to insert a virtual object (d) in the scene as an AR application

S. Peng et al., 2019; Rad and Lepetit, 2017; Tekin et al., 2018], others use an end-to-end CNN to predict the pose [Y. Li, Wang, et al., 2018; C. Wang et al., 2019; Xiang et al., 2018].

Some older methods however have proven to be reliable using patch voting approaches coupled to a learning algorithm [Fanelli et al., 2011; Kacete et al., 2016; Kehl et al., 2016; Riegler et al., 2013; Tejani et al., 2014]. Those strategies predict a set of pose hypothesis from local patches using data driven functions and, from this set of hypothesis, retrieve a pose.

We argue that we can leverage the robustness brought by local approaches with a two stages strategy, predicting the pose in an intermediate Euclidean 3D space and retrieving it with a geometrical solver. The intermediate representation makes it natural to apply a voting strategy to the set of pose hypothesis. Our hybrid strategy allows us to correctly supervise our CNN training, not being dependent on the choice of pose representation, not requiring a custom loss function to compute the pose error and not having to predict rotation and translation separately. Moreover, like [Tekin et al., 2019] we argue that predicting keypoints in 3D and solving a 3D-3D correspondence problem yields to more accurate results rather than predicting in 2D and solving a 2D-3D correspondence problem.

In this chapter we tackle the problem of pose estimation considering a single RGB-D image as input. We design a robust and accurate algorithm to predict the pose of a generic rigid object in a scene. Our contributions are :

- We propose an hybrid pipeline in two parts: a data driven block that predicts a set

of 3D points in the camera coordinate system and a geometrical block. The latter retrieves the pose given the estimated points and a priori chosen keypoints in the world coordinate system, minimizing a registration error.

- We propose to use two CNNs in cascade in the former part. First we predict the 2D location of the object in the image, classifying local patches extracted from the image with a CNN. Then we use a regression CNN to predict a set of possible 3D positions of points in the camera coordinate system. The position hypothesis are then robustly aggregated to obtain a single estimation of the 3D location of the points.
- We demonstrate performance improvements in terms of accuracy over state-of-the-art methods of RGB-D pose estimation on the standard LineMod [Hinterstoisser et al., 2012] dataset and study the impact of some parameters of our method. We also validate our approach within a visual servoing experiment.

3.2 Related Work

In this section we focus on object pose estimation algorithms. For a state-of-the-art of object based SLAM systems that make use of object pose estimation we refer the reader to the chapter 2 of this manuscript. We will limit ourselves to learning based methods as the literature on object pose estimation is vast. We can separate those methods into two main categories: patch-based methods and holistic methods. The latter can be as well separated into two categories: direct and indirect strategies.

Patch-Based Methods. Patch-based methods output multiple pose hypothesis for a single image [Fanelli et al., 2011; Gall et al., 2011; Kehl et al., 2016; Riegler et al., 2013; Tejani et al., 2014]. The predictions, called votes, which are obtained from local patches in the image are then aggregated to get a single estimation, which is more robust than each vote taken independently. Hough based methods is such a type of voting scheme. Hough Random Forests (HRFs) have been introduced by [Gall et al., 2011] to estimate the Hough transform with a learning based approach for object detection, tracking in 2D and actions recognition. The concept of HRFs has also been applied to object pose estimation by [Fanelli et al., 2011] to predict the translation and rotation of human heads. In that case, both the nose 3D position and Euler angles are regressed. Those methods rely on binary tests to describe the split hypothesis used in random forests. [Tejani et al., 2014]

proposes to use a split function based on a template patch descriptor. It also proposes to train a random forest using only object patches. As HRFs are based on handcrafted split functions, their performance is limited by image variations. To overcome this, Hough Convolutional Neural Networks (HCNNs) have been introduced by [Riegler et al., 2013] as an alternative to HRFs. A CNN was designed by [Riegler et al., 2013] to regress at once the probability of a patch belonging to the foreground as well as the object pose. In all cases a non parametric clustering algorithm is then used on object patches to robustly retrieve the pose.

Direct Holistic Methods. Recently, most studies [Do et al., 2018; Kehl et al., 2017; Y. Li, Wang, et al., 2018; C. Wang et al., 2019; Xiang et al., 2018] take a whole image as an input and try to leverage the capabilities of CNNs by directly estimating the pose. PoseCNN [Xiang et al., 2018] proposes an end-to-end CNN to perform 3 related tasks: semantic labeling, translation prediction from the object estimated 2D center and depth and rotation inference. To correctly supervise the network training, [Xiang et al., 2018] uses a specific loss called PoseLoss, defining the error as an average euclidean distance between rotated point clouds. SSD6D, [Kehl et al., 2017], uses a CNN to predict the object class with its bounding box, as well as to classify discretized viewpoints and in-plane rotations to create a set of pose hypothesis. Thus, the network loss is a parametric combination of multiple losses. DenseFusion, [C. Wang et al., 2019], combines color and depth channels in a deep network to fuse them, creating a set of features which are then used by a CNN to predict the pose. It can be further rapidly refined by a network in an iterative manner. In some recent works [Do et al., 2018; Mahendran et al., 2017; Sundermeyer et al., 2018] the choice of representation for rotations has been studied as it shows to have an impact on the accuracy of the pose estimation [Mahendran et al., 2017].

Indirect Holistic Methods. On the other side some methods [Grabner et al., 2018; Y. He et al., 2020; Z. Li et al., 2019; Pavlakos et al., 2017; S. Peng et al., 2019; Rad and Lepetit, 2017; Tekin et al., 2018; J. Wu et al., 2016] are inspired by classical pose estimation from 2D-3D correspondence. However CNNs are used to address the limits imposed by handcrafted features. To do so the 2D location of the projection of prior chosen 3D keypoints is predicted in the image. The pose is then retrieved using a 2D-3D geometrical solver e.g. a Perspective-n-Points (PnP) algorithm. For example BB8, [Rad and Lepetit, 2017] coarsely segments the object and apply a deep convolutional network to the local

window around the object to predict the 2D location of the projection of the 8 corners of the object bounding box. This estimation is followed by a PnP that recovers the pose. [Tekin et al., 2018] proposes a single-shot CNN that classifies the object, predicts a confidence term as well as the 2D location of the projection of 9 keypoints in the bounding box. PVNet[S. Peng et al., 2019] proposes to apply an offset based approach to predict the 2D location of a set of keypoints on the object surface. To do so, they segment the object in the image and predict a vector field on the segmented object, the spatial probability distribution of each keypoint is then retrieved and used in an uncertainty driven PnP to estimate the pose. H+O [Tekin et al., 2019] estimates at once hand-object poses as well as objects and actions classes from RGB images. A CNN predicts the 3D position of 21 points of the object bounding box and the object pose can be retrieved from 3D-3D correspondences. PVN3D [Y. He et al., 2020] uses 6 different networks to estimate the pose: 3 (including [C. Wang et al., 2019]) are used to extract and fuse features from a single RGB-D image and 3 are used to predict a set of 3D keypoints and the objects segmentation.

3.3 Proposed Approach: our Hybrid, Patch-Based Strategy

Our goal is to achieve a robust and accurate 6-DoF pose estimation of a 3D object, i.e. to estimate the transformation ${}^{c}\mathbf{T}_{w} \in SE(3)$ of an object of interest, provided with a coordinate system called world coordinate system $\mathcal{F}_w$, in the camera coordinate system $\mathcal{F}_c$. We represent the transformation ${}^{c}\mathbf{T}_{w}$ between $\mathcal{F}_c$ and $\mathcal{F}_w$ as

$$ {}^{c}\mathbf{T}_{w} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.1} $$

where $\mathbf{R} \in SO(3)$ is a 3D rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a 3D translation vector. For simplicity, in this chapter we denote $\mathbf{T}$ the transformation ${}^{c}\mathbf{T}_{w}$.

The pipeline of the proposed strategy can be seen Fig. 3.2: we adopt a patch voting based approach inspired from [Gall et al., 2011; Riegler et al., 2013], using multiple local information to predict a sparsified version of the object geometry in $\mathcal{F}_c$. First, we design and train a classification CNN to predict the class of patches extracted from the input image either as object or background. This allows us to roughly localize the object in

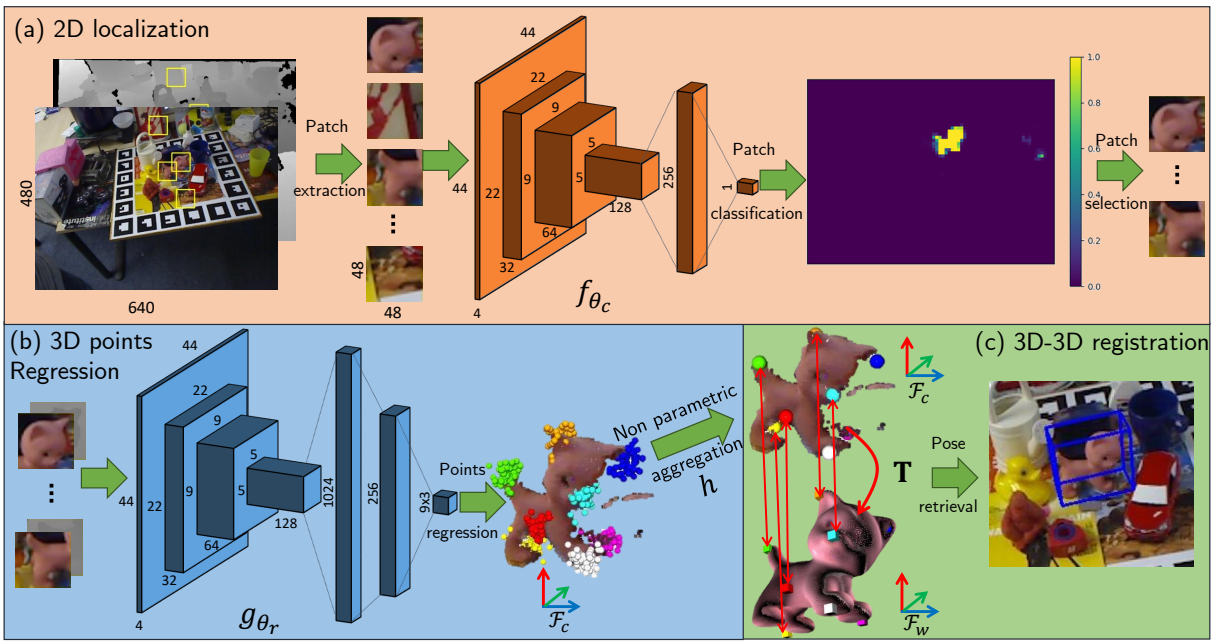

Figure 3.2 – Overview of our pipeline solution: first (a), patches are extracted from an RGB-D image and classified as object or background. Next (b), for each object patch, a regression CNN predicts a set of vectors estimating the position of specific 3D keypoints in $\mathcal{F}_c$. Those votes are then aggregated in a non parametric way to obtain a robust estimation of the points in $\mathcal{F}_c$. (c) by minimizing the registration error between corresponding estimated keypoints in $\mathcal{F}_c$ and reference keypoints in $\mathcal{F}_w$ we retrieve the 6D pose. The pipeline is illustrated here with patches of size 48×48.

2D. Then, we design and train a regression network to predict for each extracted object patch the 3D position in $\mathcal{F}_c$ of a set of prior chosen keypoints, selected in $\mathcal{F}_w$. Finally, by minimizing the 3D-3D registration error between corresponding estimated keypoints in $\mathcal{F}_c$ and reference keypoints in $\mathcal{F}_w$ we retrieve the 6D pose. Hence our method belongs to both the indirect and patch-based set of methods, contrary to most recent methods [Y. He et al., 2020; S. Peng et al., 2019] that are indirect and holistic.

3.3.1 2D Localization

In this section we show how we take in account the visibility of the object in each patch. Indeed not all patches contain relevant information about the object pose. Unlike in [Riegler et al., 2013] who uses a single network for both classification and pose estimation, we first use a classification network to decide whether or not a patch contains a representation of the object. We argue that classifying the patches, keeping only rele-

vant ones before transmitting them to the regression network allows the CNN to fit using only relevant information about the object pose. Moreover we do not need a sophisticated parametric loss function whose parameters have to be optimized to supervise the training.

Model. Our model is inspired by a light VGG-like architecture and can be seen in the first block of Fig. 3.2. It is composed of a set of convolutional layers to extract features from the images and max-pooling layers to introduce scale invariance followed by 2 dense layers to classify the extracted features. For the last layer, we use a sigmoid activation function, for each other layer we use the classical ReLu activation function. To help reduce overfitting dropout is also used on the first fully connected layer as it contains the most weights.

Data. To train our classification network in a supervised manner, we need labeled data. We capture a set of images representing the object of interest from multiple points of view. The classification neural network is trained using a set of patches $\{P_i = (\mathbf{I}_i, b_i)\}$ where $\mathbf{I}_i$ is the RGB image of the patch of size $[h \times w]$, i.e. $\mathbf{I}_i \in \mathbb{R}^{[h \times w] \times 3}$ and $b_i \in [0, 1]$ represents whether or not the object is visible in the image $\mathbf{I}_i$. We obtain it by producing a binary mask of the object created by a 2D projection of the object 3D model using its ground truth pose. To increase the robustness of our algorithm across changes in illumination we proceed to do data augmentation by randomly modifying patches brightness.

Training. We denote the classification function f_{θ_c} optimized over θ_c which represents our CNN weights. The classification parameters are optimized by minimizing over the training data set:

$$\theta_c^* = \arg\min_{\theta_c} \mathcal{L}_c(b, \hat{b}). \tag{3.2}$$

where $\hat{b} = f_{\theta_c}(\mathbf{I})$ and $\mathcal{L}_c$ is the classical weighted binary cross entropy.

Inference. Given an unseen image, we extract K patches from the image in a sliding window fashion, with K depending on the window stride d (in our experiment K can vary from to a few hundreds to a few thousands with d going from 4 to 48 pixels) and get a set of patches $\mathcal{P} = \{P_i, i \in [1, K]\}$. Each patch is then fed to the classification network $f_{\theta_c^*}$ whose output is $\left\{\hat{b}_i = p(b_i|\mathbf{I}_i) = f_{\theta_c^*}(\mathbf{I}_i), i \in [1, K]\right\}$ where p denotes the probability. We show in Fig. 3.3 some heat maps obtained using the probability estimated for each patch. We can see that the patches extracted from the object have a high probability of

being classified as object while the patches extracted from the background have a low probability. As we can see the probability maps are very similar to 2D segmentation. However our method gives coarser results and needs less data than classical segmentation methods. Moreover our strategy is flexible as we can tune the patch extraction stride to balance inference time and segmentation precision.

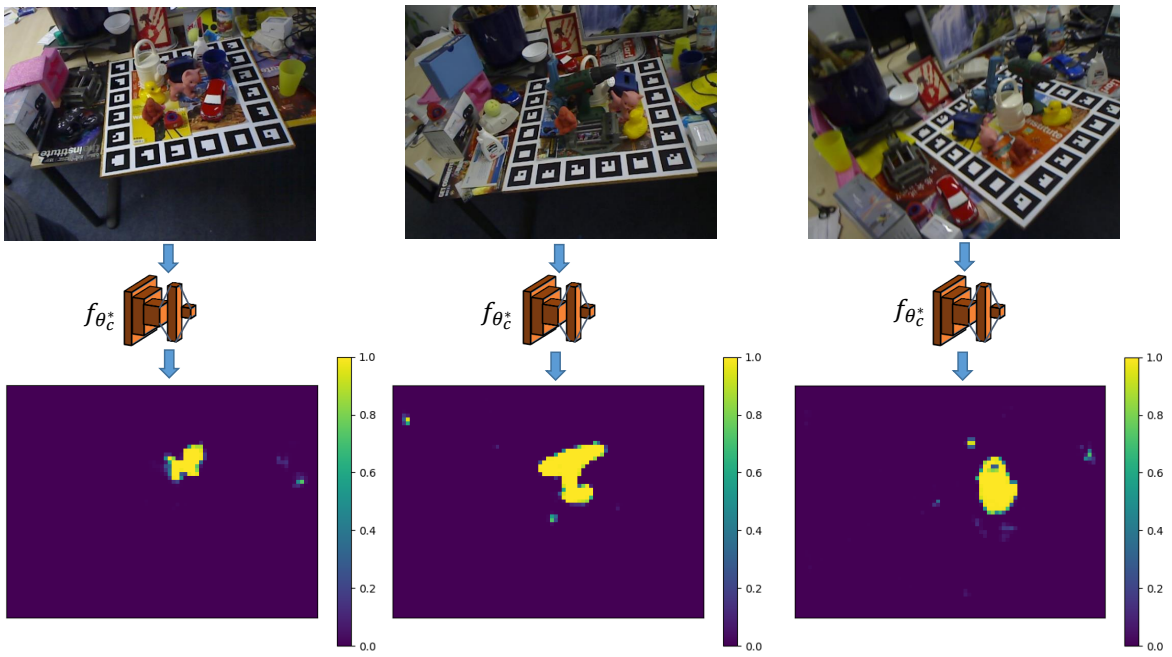

Figure 3.3 – Examples of probability maps for the cat, driller and can from the LineMod dataset

3.3.2 3D Points Prediction

We now show how we predict the position of a set of 3D keypoints in $\mathcal{F}_c$, using the object patches classified in the previous step. We use a regression network to predict the 3D location of a set of M points in $\mathcal{F}_c$. First, using a point selection algorithm like the farthest point sampling algorithm we create a set of M 3D keypoints, denoted $\mathcal{S} = \{\mathbf{X}_j \in \mathbb{R}^3, j \in [1, M]\}$, chosen in the object model in $\mathcal{F}_w$. For a given pose $\mathbf{T}$ of the object in $\mathcal{F}_c$ we express the points in $\mathcal{S}$ in $\mathcal{F}_c$, and denote the set $\mathcal{S}_\mathbf{T} \coloneqq \{\mathbf{Y}_j \coloneqq \mathbf{T}\mathbf{X}_j, j \in [1, M]\}$. Our goal is to estimate the location of the points in $\mathcal{S}_\mathbf{T}$ i.e. to estimate the location of the keypoints of $\mathcal{S}$ in $\mathcal{F}_c$. We argue that it is easier for the neural network to predict points in a euclidean space than to predict a pose over $SE(3)$. Indeed even if a geodesic distance exists

over $SE(3)$ using the twist parametrization, optimization over this distance is complex [B. Hou et al., 2018]. Like [Z. Li et al., 2019] we argue that rotations and translations should be treated differently or at least that adaptation is required to learn to regress coherently in $SE(3)$. In a way with our change of variables we suppress the direct impact of the peculiarities of rotation space as every variable stays in $\mathbb{R}^3$. Furthermore we argue that establishing 2D-3D correspondences is more ambiguous than 3D-3D correspondences for object pose estimation. Indeed the PnP algorithm aims at minimizing the 2D projection errors of keypoints. However, there may be small reprojection errors between keypoints that are large in 3D. Comparisons between 2D-3D and 3D-3D correspondences can be found in [Y. He et al., 2020].

Model. The architecture of the regression network can be seen in the second block of Fig. 3.2. We use an architecture that is very close to the classification network because we showed that we could reliably extract information from the patches with it. However we change the fully connected part, adding one layer and using more weights for each layer to give the regression CNN more flexibility.

Data. We extract only object patches P'_i from the image. A regression neural network is trained using a set of patches $\mathcal{P}' = \{P'_i = (\mathbf{I'}_i, \boldsymbol{\delta}_i)\}$ where $\mathbf{I'}_i$ is the RGB-D image of the patch, i.e., $\mathbf{I'}_i \in \mathbb{R}^{[h \times w] \times 4}$ and $\boldsymbol{\delta}_i \in \mathbb{R}^{3 \times M}$ is a set of M 3D vectors, called offsets and defined in Equation 3.3:

$$\begin{aligned} \boldsymbol{\delta}_i &= \{\boldsymbol{\delta}_{1,i}, \boldsymbol{\delta}_{2,i}, ..., \boldsymbol{\delta}_{M,i}\} \\ &= \{\mathbf{Y}_1 - \mathbf{C}_i, \mathbf{Y}_2 - \mathbf{C}_i, ..., \mathbf{Y}_M - \mathbf{C}_i\} \end{aligned} \tag{3.3}$$

with $\mathbf{Y}_j \in \mathcal{S}_{\mathbf{T}_i} \quad \forall j \in [1, M]$, $\mathbf{T}_i$ is the pose of the object visible in the i^{th} patch and $\mathbf{C}_i \in \mathbb{R}^3$ is defined by :

$$\mathbf{C}_i = \begin{pmatrix} \frac{u_i - c_x}{f_x} Z_i \\ \frac{v_i - c_y}{f_y} Z_i \\ Z_i \end{pmatrix} \tag{3.4}$$

with f_x, f_y, c_x, c_y the camera intrinsics, (u_i, v_i) the 2D position of the center of the i^{th} patch and Z_i the value of the patch depth at location (u_i, v_i). Equation 3.4 corresponds to the 3D back projection of the 2D center of the i^{th} patch, using a pinhole model. Thus, $\boldsymbol{\delta}_i$ is a set of M vectors, each one going from the 3D center of the patch and one of the M points in $\mathcal{S}_{\mathbf{T}_i}$. An example of offsets is visible in Fig. 3.4: for 3 patches extracted in the image,

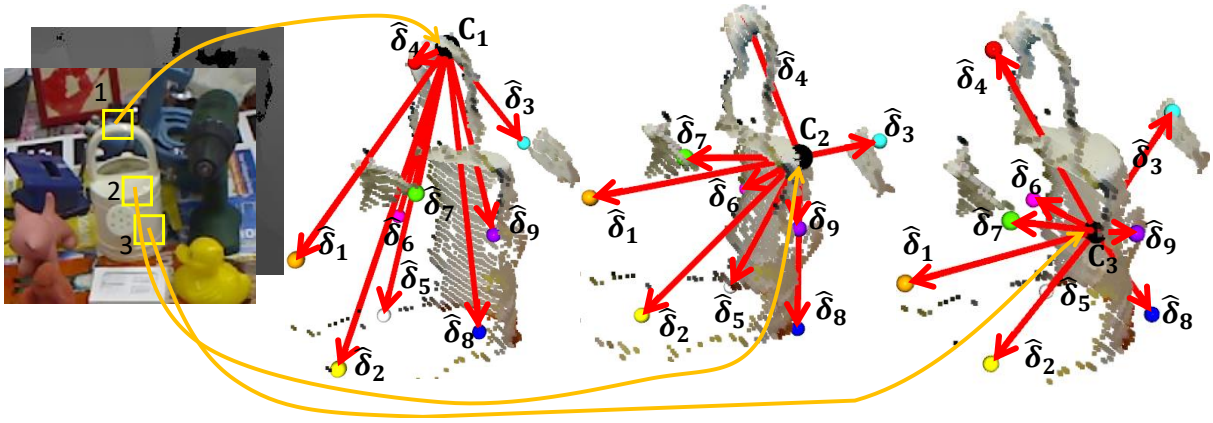

Figure 3.4 – Example of 3 patches extracted from an RGB-D image with the estimated offsets for each patch in red and the corresponding points represented by colored spheres, the center of each patch is shown by a black sphere

we show $M = 9$ offsets. The use of offsets is very interesting for object pose estimation for two reasons. First, offsets bring invariance translation that is necessary due to the fact that we consider local patches. Indeed, 2 patches extracted from 2 different images with different poses may be very resembling. If displacement vectors are not used, the difference in terms of pose can thus only be seen as noise by the network. On the contrary, if offsets are employed the variable to regress is more correlated to patches aspect. Second, let's consider the space of all possibles object translation denoted Ω_t, if we do not use offsets then this space is at most $\mathbb{R}^3$. However when considering displacement vectors, the set of all possible offsets has an upper bound of D where D is the largest diameter of the object, thus $\Omega_t \subseteq \Omega_\delta = B(0, D)$ where $B(0, D)$ is the ball of center 0 and radius D and necessarily we have $\Omega_\delta \subset \mathbb{R}^3$. The manifold in which predictions exist is reduced when using offsets, making the learning procedure easier for a data driven algorithm.

Training. We denote the regression function g_{θ_r} where θ_r is the vector of weights of the network. The regression parameters are optimized by minimizing over the training data set:

$$\theta_r^* = \arg\min_{\theta_r} \mathcal{L}_r(\boldsymbol{\delta}, \hat{\boldsymbol{\delta}}) \tag{3.5}$$

where $\hat{\boldsymbol{\delta}} = g_{\theta_r}(\mathbf{I})$ and $\mathcal{L}_r$ is the mean absolute error:

$$\mathcal{L}_r(\boldsymbol{\delta}, \hat{\boldsymbol{\delta}}) = \frac{1}{3M} ||\boldsymbol{\delta} - \hat{\boldsymbol{\delta}}||_1 \tag{3.6}$$

where $||.||_1$ is the usual L^1 norm, thus $\mathcal{L}_r$ represents the averaged L^1 distance between

the estimated and ground truth points. The L^1 norm is preferred to the L^2 norm because it is less sensitive to outliers, that are robustly handled by the Gaussian kernel of the mean-shift algorithm (see below) during the voting step.

Inference and Voting. Given the patches extracted in Sec. 3.3.1 and their associated estimated probability, we discard the patches which probability is lower than a threshold τ. Thus we get a set: $\mathcal{P}_\tau = \left\{\mathbf{I}_i | \hat{b}_i > \tau, i \in [1, K]\right\}$ that can be written: $\mathcal{P}_\tau = \{\mathbf{I}_i, i \in [1, N]\}$ where N is the number of object patches that can go from a few tens to a few hundreds. For the i^{th} patch $\mathbf{I}_i$ fed to the regression network we get M predicted 3D offsets $g_{\theta_r^*}(\mathbf{I}_i)$ denoted $\hat{\boldsymbol{\delta}}_i = \left\{\hat{\boldsymbol{\delta}}_{j,i}, j \in [1, M]\right\}$. We can then get an estimation of the 3D location of the transformed points by adding the position $\mathbf{C}_i$ of the 3D center of the patch obtained from 3.4. This way we get a set of M estimated points positions $\left\{\hat{\mathbf{Y}}_{j,i}, j \in [1, M]\right\}$ in $\mathcal{F}_c$. When we take in account all the N patches we get $N \times M$ points: $\hat{V} \coloneqq \left\{\hat{\mathbf{Y}}_{j,i}, i \in [1, N], j \in [1, M]\right\}$ that can be viewed as M clusters of N points or votes in the $\mathcal{F}_c$. We denote the j^{th} cluster of points $\hat{V}_j \coloneqq \left\{\hat{\mathbf{Y}}_{j,i}, i \in [1, N]\right\}$. The votes must then be aggregated to get a robust estimate of the 3D position of each point in the $\mathcal{F}_c$. We denote the aggregation function $h : \mathbb{R}^{3\times N} \longmapsto \mathbb{R}^3$. It is necessary to aggregate the N 3D votes in a robust manner to limit the impact of possible outliers, hence h is chosen to be a non-parametric estimator of the maxima of density. In our case we use a mean-shift estimator [Y. Cheng, 1995; Comaniciu and Meer, 2002] which iteratively estimates the local weighted mean in 3.7:

$$m(\mathbf{X}) = \frac{\sum_i k(\mathbf{X}_i - \mathbf{X})\mathbf{X}_i}{\sum_i k(\mathbf{X}_i - \mathbf{X})} \tag{3.7}$$

where k is a kernel function such as a Gaussian kernel: $k_\sigma(\mathbf{X} - \mathbf{Y}) = \exp(-\frac{||\mathbf{X}-\mathbf{Y}||^2}{2\sigma^2})$ which gives less weight to outliers. The parameter σ was not optimized here but it can simply be chosen to optimize a pose error metric on a validation set. Thus we can define the set $\hat{\mathcal{S}}_\mathbf{T} \coloneqq \left\{\tilde{\mathbf{Y}}_j \coloneqq h(\hat{V}_j), \forall j \in [1, M]\right\}$ which corresponds to the aggregated centroid of each cluster in $\mathcal{F}_c$. We show Fig. 3.5 such examples of votes.

3.3.3 3D-3D Correspondence Alignment

In this section we show how to retrieve the pose using the estimated 3D keypoints in $\mathcal{F}_c$ that we obtained in the previous step and their corresponding reference keypoints in $\mathcal{F}_w$. Once the centroids have been voted we align the estimated points and their corresponding reference to get a pose estimation from the estimated location of the points. To do so we

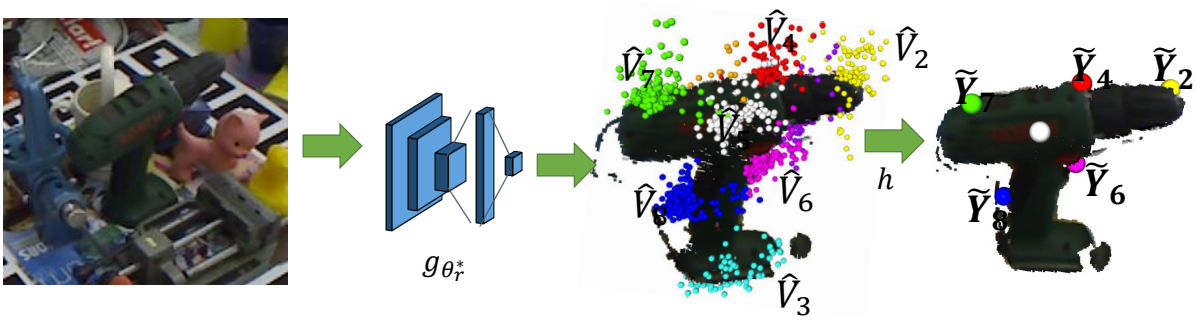

Figure 3.5 – Example of predicted 3D points. From left to right: the cropped input image, the cluster of 3D votes $\hat{V}_j$where each color corresponds to votes for a single point, the aggregated points $\tilde{\mathbf{Y}}_j$ obtained by mean-shift (best seen in color). The aggregated point of cluster $\tilde{V}_3$ is hidden behind the driller point cloud.

seek to find the transformation $\mathbf{T}^* \in SE(3)$ that minimizes:

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} \sum_{j=1}^{M} ||\mathbf{T}\mathbf{X}_j - \tilde{\mathbf{Y}}_j||_2^2 \tag{3.8}$$

where $\mathbf{T}$ can also be represented with a minimal representation $q \in \mathfrak{se}(3)$ where $\mathfrak{se}(3)$ is the Lie algebra associated to $SE(3)$, $X_j \in \mathcal{S}$, $\tilde{Y}_j \in \hat{\mathcal{S}}_{\mathbf{T}}$ and $||.||_2$ is the euclidean norm of $\mathbb{R}^3$. That is finding the pose that best fits the points estimated by the aggregation of votes in $\mathcal{F}_c$. This problem is called the Orthogonal Procrustes Problem and can be solved using SVD decomposition as shown in [Arun et al., 1987] or an Iteratively Reweighted Least Square algorithm [Fitzgibbon, 2003; Malis and Marchand, 2006] to discard outliers and obtain a robust estimation. To further refine the pose we can apply an Iterative Closest Point (ICP) algorithm [Besl and McKay, 1992]. This consists in solving 3.8, using the 3D model points and the points measured by the RGB-D camera projected in 3D using 3.4.

3.4 Experiments

We now present the results we obtained on the LineMod [Hinterstoisser et al., 2012] dataset. This section is divided in four parts: first we present the technical details of our implementation, then we evaluate our method in terms of classification accuracy, 3D points regression accuracy and we measure the object pose accuracy using a standard metric and compare it to state-of-the-art results. Finally we compute the average inference

time for a given object and study the impact of some hyper-parameters such as the level of patch density on both pose accuracy and inference time. Last but not least, a visual servoing task based on the proposed method is also considered.

3.4.1 Implementation Details

Training. To build our training data we extract patches in a sliding window fashion. We train the classification network for 100 epochs and the regression network for 500 epochs. We use a learning rate of 10^{-4} with the Adam optimizer. A dropout of 50% is used for the classification CNN and dropout of 20% and 10% is used on the two first fully connected layers of the regression CNN. We implement the neural networks using the tensorflow [Martıén Abadi et al., 2015] framework.

Keypoints Selection. Inspired by [S. Peng et al., 2019], we select the keypoints using the farthest point sampling algorithm which allows us to get a good coverage of the object. In our experiments we chose to use M = 9 points.

Inference. Our algorithm is implemented in Python. During the inference we extract patches with a stride d of 4 pixels. We chose to set the threshold τ at 0.98.The votes are aggregated using the mean-shift algorithm and a gaussian kernel with variance $\sigma^2 = 40^2 mm$. We use open3d ICP, on the sub map defined by the estimated bounding box. For testing we use a Nvidia RTX2070 and an Intel Xeon @3.7 GHz. The patch size is chosen to be $h \times w = 64 \times 64$ unless precised otherwise.

3.4.2 Datasets

The LineMod dataset consists of about 15 000 RGB-D images of 13 objects with multiple recorded poses in a variable and cluttered environment. It is widely used by the pose estimation community. We use the same method as [S. Peng et al., 2019; Rad and Lepetit, 2017; C. Wang et al., 2019] to select training and testing images. The dataset being small (about 100 images for training, yielding tens of thousands of patches), it is very challenging for CNN based approaches as highlighted in [Xiang et al., 2018]. This makes some methods like [S. Peng et al., 2019; Xiang et al., 2018] need synthetic data. Very large datasets such as the YCB-Video dataset [Xiang et al., 2018] may not be available for some objects as their creation is expensive in time as well as complex. Using data

Table 3.1 – True positive rate (TPR) and true negative rate (TNR) (in %) for each object using our classification network

	ape	ben.	cam	can	cat	drill.	duck
TPR	98.2	89.8	92.8	91.0	97.5	95.1	91.3
TNR	99.9	99.5	99.6	99.7	99.7	99.7	99.9
	hole.	iron	lamp	eggbox	glue	phone	MEAN
TPR	97.1	94.9	88.4	94.2	90.7	89.6	93.1
TNR	99.7	99.5	99.5	99.8	99.8	99.5	99.7

augmentation on the brightness we manage to get state of the art results without needing additional synthetic data, contrary to [Y. He et al., 2020; S. Peng et al., 2019] that need, in their present state, to use 20 000 new synthetic images per object.

3.4.3 Classification Accuracy

In this subsection, we measure the performance of the classification network. Having a bad classification accuracy could lead to multiple patches being misclassified. A high false positive rate would create noise in the Hough space and complexify the task of finding the maximum density. On the contrary, a high false negative rate would reduce the number of patches used for regression and thus the number of votes, leading to a less robust estimation. We can see in table 3.1 that for every object we get a high true negative rate above 99.5 % meaning we do not pollute the vote space. The true positive rate is more variable but stays above about 90%, so not too many patches are discarded.

3.4.4 3D Points Regression Accuracy

In this subsection, we study the accuracy of the regression network. For each object we measure the average euclidean distance between the estimated position of each keypoint after it has been aggregated and its ground truth position

We can see in table 3.2 that the euclidean distance between predicted and ground truth points is on average of 11.8 mm.

Table 3.2 – Average euclidean distance and standard deviation (in mm) between ground truth and predicted 3D points for each object

	ape	ben.	cam	can	cat	drill.	duck
Avg. (mm)	7.2	12	16.9	11.7	9.4	12.8	10.1
Std. (mm)	4.5	7	65.6	6.7	5.5	8.3	6.3
	hole.	iron	lamp	egg.	glue	phone	MEAN
Avg. (mm)	10.2	10.6	12.4	14	14.6	11.1	11.8
Std. (mm)	19.6	36.9	42.6	8.8	14.4	22.9	19.2

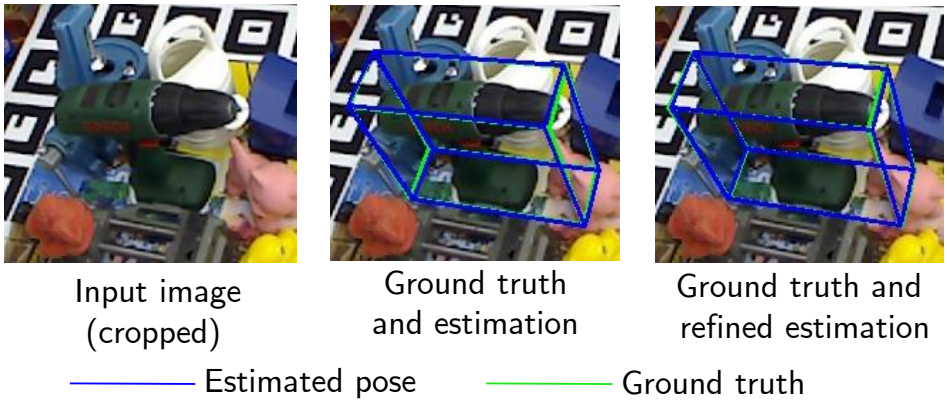

Figure 3.6 – Some qualitative examples of pose estimation results on LineMod. The pose estimation is represented as the blue bounding box and the ground truth pose as the green bounding box

3.4.5 Object Pose Accuracy

Metric. We use the standard 6 DoF metric developed in [Hinterstoisser et al., 2012], the average distance of model points (ADD-S). A pose is considered correct if the value of the ADD-S is less than 10% of the object diameter D. We report the results in table 3.3. As we can see we obtain on average very close results to [Y. He et al., 2020] while using about 200 times less data which shows the superiority of our method in the small data regime. For some objects we even obtain better results than [Y. He et al., 2020].

3.4.6 Inference Time

Inference time is greatly dependent on the choice of the density with which patches are extracted. The lower the stride is, the more patches have to be extracted and fed

Table 3.3 – Percentage of correctly predicted poses using the ADD-S metric on the LineMod dataset compared to state-of-the-art methods. Eggbox and glue are considered as symmetric objects.

Input	RGB	RGB-D			
Method	[S. Peng et al., 2019]	[C. Wang et al., 2019] w. ref.	[Y. He et al., 2020]	Ours	Ours + ICP
ape	43.6	92.0	**97.3**	91.2	**97.3**
ben.	99.9	93.0	99.7	**100.0**	**100.0**
cam	86.9	94.0	**99.6**	95.2	98.4
can	95.5	86.0	**99.5**	98.1	**99.5**
cat	79.3	93.0	**99.8**	98.4	99.7
drill.	96.4	97.0	99.3	98.8	**99.8**
duck	52.6	87.0	**98.2**	82.2	98.0
hole.	81.9	92.0	**99.8**	93.7	98.8
iron	98.9	97.0	**100.0**	99.1	99.9
lamp	99.3	95.0	**99.9**	98.6	99.1
egg. (sym.)	99.2	**99.8**	99.7	99.3	99.3
glue (sym.)	95.7	**100.0**	99.8	99.2	99.0
phone	92.4	93.0	99.5	98.3	**99.6**
MEAN	86.3	94.3	**99.4**	96.3	99.1

Table 3.4 – Inference times (in ms) and pose accuracy (ADD-S %) for the driller, using tiny YOLOv3 and different strides (pixels).

Stride	Bbox estimation	Patch extract.	Classification	Regression.	Voting	3D-3D solving	Total	ADD-S
4	8.8	83.4	187.1	69.1	176.5	0.3	525.2	98.7
8	9	20.8	51.2	23	46.4	0.2	150.6	98
12	8.7	3	11	9	11.1	0.3	43.1	96.3
16	8.9	1,9	8,6	7.6	7.7	0.3	35	95.3
20	8.6	1.2	7.6	6.4	5.4	0,3	29.5	94.3
24	9	1	7.4	5.5	4.6	0.3	27.8	90.4

to the networks and the longer the inference will be. However we expect the accuracy to be growing with the number of patches extracted. This balance allows our method to be suitable to a wide range of methods. The flexibility it brings lets the user tune the extraction stride to better meet the application needs. To decrease inference time we retrained a light 2D detection algorithm (namely tiny YOLOv3 [Redmon and Farhadi, 2018] with a darknet backbone) on the driller. As the training set is very small the estimated bounding box is coarse but sufficiently precise to reduce inference time which is reported in table 3.4. As we can see, using a stride of 12 to 16 we can reach real time inference while losing little accuracy. To speed up the voting step we chose to use only 3 clusters that are selected to minimize their respective variance. We also report inference times and accuracy for different strides in Fig. 3.7.

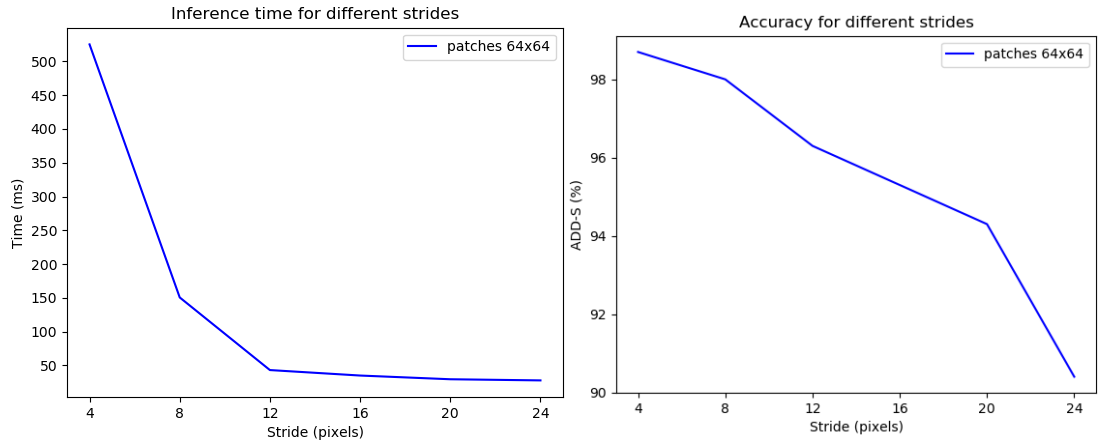

Figure 3.7 – Inference time (in ms) and accuracy for the driller for varying strides (in pixels)

Table 3.5 – Total inference time (in ms) for each patch size (in pixels) using tiny YOLOv3 and a stride of 4 pixels

Patch size (pixels)	32	48	64	80
Total inference time (ms)	391.6	413	525.2	634.9

3.4.7 Ablation studies.

Influence of patch size. In this section we study the influence of different patch sizes for 4 LineMod objects on the pose estimation accuracy. Patch size is an important parameter and depends on the object size and texture. For example textureless objects may be better represented using larger patches that can capture some of the object shape. The influence on inference time can be seen for the driller in table 3.5, as expected inference time grows with patch size as more convolutions are needed. The pose estimation accuracy can be seen in figure 3.8 and shows that pose accuracy grows with patch size, up to 64×64 and then goes down for most objects.

Importance of depth for the regression network. In this section we study the influence of depth for the offset prediction. A regression CNN is trained using only the RGB channels as input. We compute the object pose accuracy using the ADD-S metric on 6 different objects and report the results in table 3.6. As we can see we obtain slightly better results with this method for some objects. This shows that locally an RGB patch can be more discriminant than a depth patch, this can be explained by the fact that depth patches show little details and variation compared to color. However this depends on the

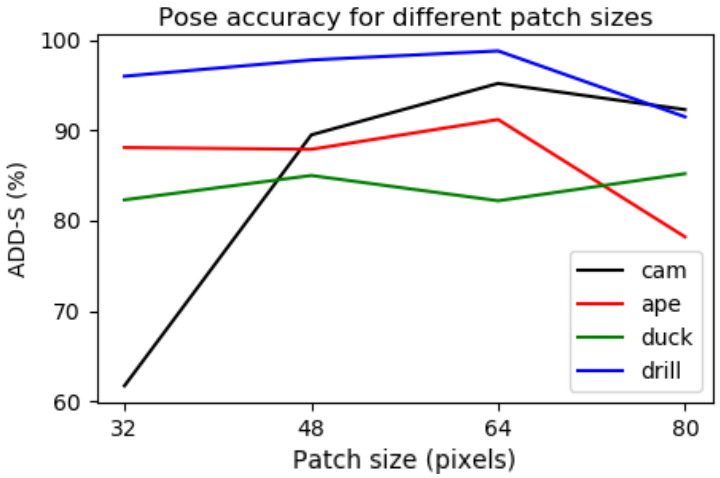

Figure 3.8 – ADD-S (%) for different patch size (in pixels) and objects.

Table 3.6 – Comparison of ADD-S using RGB only and RGB-D.

Objects	RGB ADD-S (%)	RGB-D ADD-S (%)
ape	**91.9**	91.2
cam	94.4	**95.2**
can	**99.1**	98.1
drill	**99.8**	98.8
lamp	98.5	**98.6**
phone	97.6	**98.3**
MEAN	96.9	96.7

object. For example many patches extracted on the object "can" or "lamp" just have a slightly curved depth. The depth patches extracted from the driller show very little details compared to the RGB, contrary to the cam that shows little texture on the RGB while being textured for the depth.

Importance of voting. In this section we validate the voting approach. For two small objects we use only one 80×80 patch to regress 3D points that are directly used to estimate the 3D transform. Doing so we obtain an ADD-S of 44.2% for the ape and 51.7% for the duck, compared to 78.2% and 85.2% which clearly shows the benefits of using multiple patches.

3.4.8 Visual Servoing Experiment

To illustrate that ou approach is well suited for robotics application, We also propose to validate our approach within a visual servoing experiment [Chaumette and Hutchinson, 2006]. Such experiment required that the pose estimation algorithm is not only precise but fast and stable over time. We consider a positioning task with respect to an object. The aim of a positioning task is to reach a desired pose of the camera $\mathbf{r}^*$, starting from an arbitrary initial pose. We proposed to consider a position-based VS (PBVS) scheme [Chaumette and Hutchinson, 2006] for which the relative pose (position and orientation) between the current and desired camera position has to be estimated. This relative pose will be estimated using the approach presented in this paper.

The PBVS task is achieved by iteratively applying a velocity to the camera in order to minimize $\Delta\mathbf{T}$ which is defined such that $\Delta\mathbf{T} = \mathbf{T}^*\mathbf{T}^{-1}$ (where both $\mathbf{T}^*$ and $\mathbf{T}$ are computed using L6DNet at the desired and current position). The control law is then given by (see [Chaumette and Hutchinson, 2006] for details):

$$\mathbf{v} = -\lambda \mathbf{L}^{+} \mathbf{\Delta r} \tag{3.9}$$

where λ a positive scalar and $\mathbf{L}^+$ is the pseudo inverse of the interaction matrix $\mathbf{L}$ that links the variation of the pose to the camera velocity $\mathbf{v}$. The error is defined by $\mathbf{\Delta r} = (\mathbf{t}, \theta\mathbf{u})$, where $\mathbf{t}$ describes the translation part of the homogeneous matrix $\Delta\mathbf{T}$ related to the transformation from the current $\mathcal{F}_c$ to the desired frame $\mathcal{F}_{c^*}$, while its rotation part $\mathbf{R}$ is expressed under the form $\theta\mathbf{u}$, where $\mathbf{u}$ represents the unit rotation-axis vector and θ the rotation angle around this axis.

Once the displacement $\mathbf{\Delta r}$ to be achieved is computed using our approach, it is immediate to compute the camera velocity using a classical PBVS control law [Chaumette and Hutchinson, 2006; Wilson et al., 1996]:

$$\mathbf{v} = -\lambda \begin{pmatrix} \mathbf{R}\ \mathbf{t} \\ \theta\mathbf{u} \end{pmatrix} \tag{3.10}$$

In such approaches, the quality of the positioning task and camera trajectory is then dependent on the quality of the estimation of the relative pose.

Experiments have been carried out on a 6 DoF gantry robot, with an Intel D435 mounted on the end-effector. Fig. 3.9 illustrates the behavior of the considered VS control law. The displacement to be achieved is $\mathbf{\Delta}r = (-400mm, -140mm, -240mm, 6.22^o, 36.80^o,$

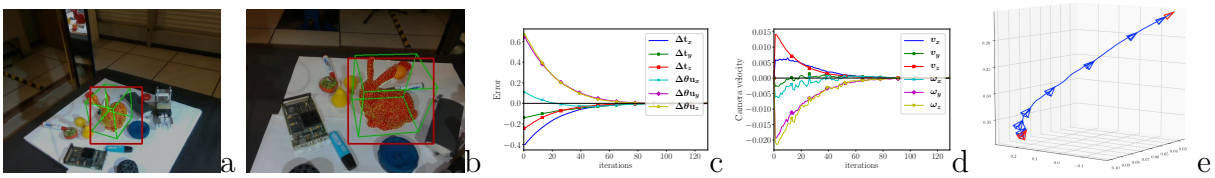

Figure 3.9 – Visual servoing experiment: (a) initial view (the 3D model of the target is superimposed in the image according to the estimated pose) (b) final image (c) error $\mathbf{\Delta r}$ in (meter and radian) (d) camera velocities (in meter/s and radian/s) (e) camera position over time.

38.36^o). The transformation between the initial and desired poses (and particularly the rotation around the y and z axes and translation along the x and z axis) is very large and makes this experiment very challenging. The final error is $\mathbf{\Delta r} = (1.2mm, -1.6mm, -0.4mm, -0.09^o, -0.18^o, 0.07^o)$. Note that the evolution of the errors and of the velocities are very smooth and that the camera trajectory is close to the expected straight line (despite a coarse eye-to-hand calibration) which demonstrated both the accuracy and efficiency of the proposed approach on long image sequences.

3.5 Conclusions

In this paper, we introduced a novel approach to estimate 6-DoF object pose in a RGB-D image. Our method leverages the strengths of patch voting based strategies and hybrid learning-geometrical methods, using patches extracted from the image to predict a set of sparse 3D keypoints representing the object geometry in $\mathcal{F}_c$. Those points are then put in correspondence and aligned with reference keypoints to retrieve the pose. We showed that our strategy is more robust and accurate than state-of-the-art and efficient to control a camera mounted on a robot end-effector in real-time. As we can train our networks very rapidly it is possible to integrate our pipeline in an object based SLAM system to improve camera pose estimation and obtain a higher level map. However our approach still requires specific objects to be present in the scene, which can limit the applicability of the object based SLAM system. To go beyond this limit, we propose new, more generic algorithms, that we present in the following chapters.

This chapter was previously accepted as a paper:

Mathieu Gonzalez, Amine Kacete, Albert Murienne and Eric Marchand, "L6DNet: Light 6 DoF Network for Robust and Precise Object Pose Estimation with Small Datasets", *in: IEEE Robotics and Automation Letters (RA-L), vol. 6 (2), pp. 2914-2921*, 2020. It was also presented presented in the: *IEEE International Conference on Robotics and Automation (ICRA)*, 2020

Chapter 4

S^{3}LAM: Structured Scene SLAM

Contents

In this chapter we propose to go beyond the work of L6DNet, that was limited to specific objects, by developing a SLAM system that can create a map of generic objects. Our SLAM system creates clusters of 3D points in the map, each cluster uniquely corresponding to an object in the scene. To do so we propose to use a panoptic segmentation network to segment 2D keyframes which are then processed to obtain a consistent 3D map. Then we associate this semantic information with an a priori knowledge of objects geometry to improve the quality of the map and obtain higher level entities in the map. We specifically focus here on objects that can be represented using planes that we estimate to constrain 3D map points. Finally we show in experiments on public datasets that our approach allows us to obtain a high level semantic map. We also show that the geometrical constraints can improve the map and camera poses estimation accuracy on planar scenes. A video explaining the approach and showing examples of semantic map building and plane fitting can be found at https://youtu.be/JFZaLCXtOiw.

4.1 Introduction

In the previous chapter we have presented our object pose estimation network L6DNet, that can be integrated into a SLAM system to use objects as high level landmarks in the map [Civera et al., 2011; Fioraio and Di Stefano, 2013; Gálvez-López et al., 2016; Salas-Moreno et al., 2013]. However the main limit of this approach is that it requires the

network to be trained on specific objects with a given 3D model. Hence it cannot be applied on many objects (e.g. trees, cars, pedestrians). This hinders one of our goal, which is the development of a generic SLAM system that can work in any environment. Furthermore a SLAM system that uses objects as high level landmarks cannot be considered as fully semantic as the major part of the environment often does not correspond to object instances (e.g. the road, buildings, vegetation, etc.) With the rise of CNNs, methods for object detection [Redmon et al., 2016] and segmentation [K. He et al., 2017; Kirillov et al., 2019a] are now available for practical use applications. Those approaches are particularly interesting to solve our problem as they are trained on object categories instead of specific objects. Two families of SLAM systems can be described, depending on the type of semantic information used. On the one hand several approaches fuse multiple segmentations to obtain consistent semantic maps [McCormac et al., 2017; Tateno et al., 2017].

On the other hand some works propose to represent objects in a generic way using for example quadrics [Hosseinzadeh et al., 2019; Nicholson et al., 2018] or 3D bounding boxes [S. Yang and Scherer, 2019a] and use a generic object detector. Planes can be seen as specific objects that are numerous in many environments and can be also integrated into SLAM [Arndt et al., 2020; Hosseinzadeh et al., 2019; Kaess, 2015; S. Yang et al., 2016].

In this chapter we present a monocular SLAM system, called S^3LAM for Structured Scene SLAM, based on the state-of-the-art ORB-SLAM2 [Mur-Artal and Tardós, 2017] that can segment generic objects in the scene using a panoptic segmentation CNN, namely detectron2 [Y. Wu et al., 2019]. We propose to create a new scene representation in which objects are seen as clusters of triangulated 3D points with semantic information. This allows us to create a semantic map with unique object instances and structures. Using this map of clusters we make use of prior information about object classes to constrain the map, which improves camera localization, and provides a higher level semantic map as shown in figure 4.1. Our work is a first step towards the creation of the scene graph presented in the introduction of this thesis.

In summary, contributions presented in this chapter are:

- A SLAM framework that can detect object instances in the scene to create clusters of 3D points corresponding to such objects.
- A monocular SLAM system that can infer structures from clusters and constrain the map given such estimations.
- An evaluation of our approach on sequences from several public datasets which

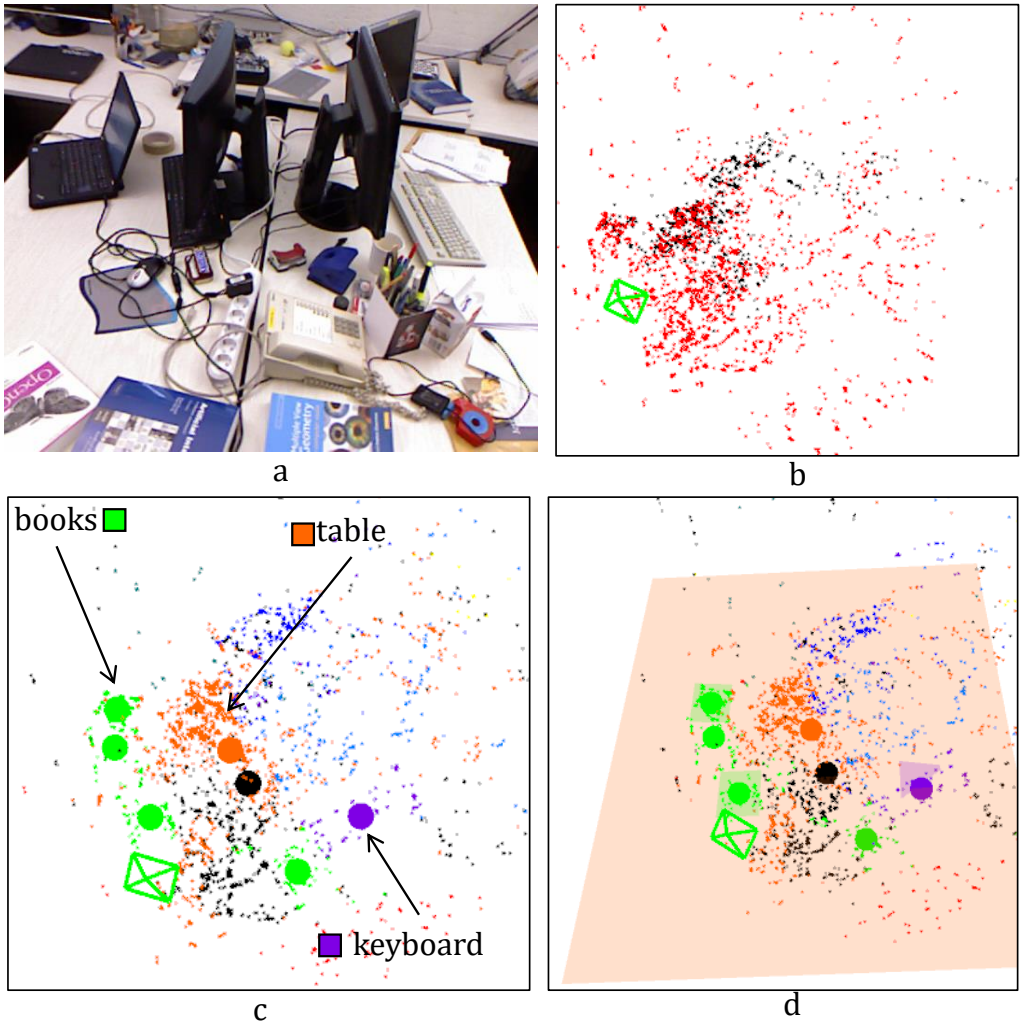

Figure 4.1 – (a) frame from the sequence fr1_desk. Comparison of (b) a map built by ORB-SLAM2, (c) a map of clusters where each cluster centroid is represented with a big sphere (books are in green, table in orange, keyboard in purple, other objects are not used) and (d) a map of clusters with estimated planes.

demonstrates the benefits of our method in terms of camera pose precision.

The rest of the chqpter is described as follows. First we describe related work on classical and semantic SLAM. Then we describe our approach to create a structured map and make use of those structures. Finally, we demonstrate the benefits of our approach on several sequences from a public dataset.

4.2 Related work: Semantic SLAM

In this section we first propose a rapid reminder of the state of the art of semantic SLAM and object based SLAM and we introduce some plane-based SLAM systems.

Semantic information can be calculated using CNNs and then injected into a SLAM map [McCormac et al., 2017; Sünderhauf et al., 2017; Tateno et al., 2017]. SemanticFusion [McCormac et al., 2017] is a dense SLAM which computes a pixelwise probability distribution for each frame and fuse the results for each surfel using a bayesian approach, which gives a semantic dense map. The semantic map can then be used to improve the SLAM.

Object based SLAM systems consist in detecting objects in the scene and inserting them in the map to add constraints between frames, thus adding temporal consistency [Fioraio and Di Stefano, 2013; Gálvez-López et al., 2016; Hosseinzadeh et al., 2019; Nicholson et al., 2018; Salas-Moreno et al., 2013; S. Yang and Scherer, 2019a]. This can bring robustness to the SLAM and accuracy by having access to the objects scale. Those systems can be divided into two main categories. The first one consist of object based SLAM systems that use specific objects [Civera et al., 2011; Fioraio and Di Stefano, 2013; Gálvez-López et al., 2016; Salas-Moreno et al., 2013]. SLAM++ [Salas-Moreno et al., 2013] proposes to estimate the 6 DoF pose of objects in the scene from RGB-D images. Each estimated object is rendered using its mesh and the pose of the camera is estimated by minimizing the ICP error with the live depth frame. SLAM++ builds a graph of keyframes and objects as a map and optimizes the pose graph. On the other hand some works propose to model objects using quadrics [Hosseinzadeh et al., 2019; Nicholson et al., 2018] or 3D bounding boxes [S. Yang and Scherer, 2019a]. QuadricSLAM [Nicholson et al., 2018] uses quadrics for localization and mapping. The main idea is to generate quadrics from 2D bounding boxes predicted by an object detection network. The quadrics parameters and keyframe poses are then refined in a BA so that the 2D projection of quadrics tightly fits the 2D bounding boxes.

Planar SLAM systems consist in detecting planar structures in the scene and using them as high level landmarks [Arndt et al., 2020; Concha and Civera, 2015; Concha et al., 2014; Hosseinzadeh et al., 2019; Hsiao et al., 2017; Kaess, 2015; S. Yang and Scherer, 2019b; S. Yang et al., 2016]. The goal is threefold: first, planes are usually large structures and can thus constrain different parts of a scene without visual overlap. Second, some man-made planar structures do not contain valuable information for keypoint based SLAM, for example white walls, hence using planes as landmarks can enable tracking and mapping in those challenging cases. Finally detecting planes in the scene allows to get a better understanding of its physical structure which can enable interaction, contrary to a simple sparse point cloud. [Kaess, 2015] proposes to use only planes as landmarks. Planes are extracted from RGB-D images and injected in a graph based SLAM using a minimal representation which allows them to be optimized with keyframes trajectory. [Hosseinzadeh et al., 2019] uses planes to constrain the SLAM map. Planes are estimated from 3 different neural networks that estimate depth, normals and plane segmentation. From these redundant estimations planes are inserted in the map. The point-plane distance is then minimized within the BA. [Arndt et al., 2020] proposes a monocular SLAM using planes to constrain the structure of the scene. Planes are estimated using a RANSAC on the whole map, thus they need to be large enough and the framerate need to be low enough for the RANSAC to find the panes. In-plane points are then projected onto the estimated planes and both 2D plane points as well as plane parameters are optimized to minimize the reprojection error.

4.3 S^{3}LAM: A cluster based SLAM

In S^{3}LAM the map is represented as a set of point clouds, grouped according to the object instance they belong. Our goal is to use prior knowledge about these objects to enrich the SLAM, improve camera pose estimation as well as to obtain a better representation of the structure of the scene. Our goal is to estimate the pose of a monocular camera moving in 3D space, represented at time i by the 6 DoF transformation between world frame $\mathcal{F}_w$ and camera frame $\mathcal{F}_{c_i}$ with the homogeneous matrix that defines the transformation denoted ${}^{c_i}\mathbf{T}_w \in SE(3)$. The pipeline of our approach can be seen in figure 4.2. We segment each keyframe using a panoptic segmentation network. In the mapping thread we update the class distribution of map points using the output of the panoptic segmentation network. Segmented map points allow us to create semantic clusters that

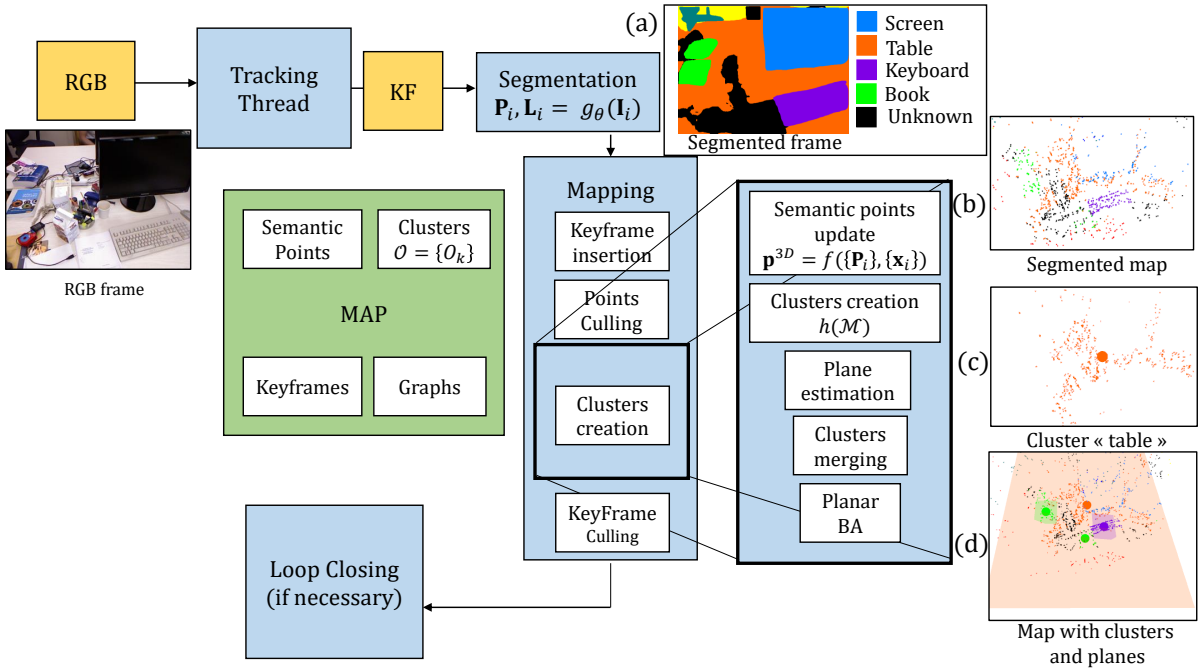

Figure 4.2 – Illustration of our pipeline integrated in [Mur-Artal and Tardós, 2017]. (a) A CNN segments the keyframes, (b) the output of the CNN allows us to compute the probability distribution of map points, (c) we create clusters of points based on their semantic class and fit planes for planar classes, (d) we apply our BA constrained by planes.

uniquely correspond to object instances. For some clusters of classes corresponding to planar objects a plane is fitted using the 3D points and a merging step avoids the creation of duplicated clusters. A bundle adjustment with a planar constraint is then applied to refine camera pose estimation and map points position. Compared to state of the art our approach relates to object based methods that represent objects as quadrics [Hosseinzadeh et al., 2018; 2019; Nicholson et al., 2018] that are very generic and approaches that estimate planes to represent the map [Arndt et al., 2020; Hosseinzadeh et al., 2018; 2019]. However contrary to approaches that use quadrics, ours yields a representation that is closer to reality, which can further improve camera pose estimation accuracy. And contrary to approaches that use planes, ours does not need depth information, nor specific CNN to estimate planes, nor very large planes and can work in real time.

4.3.1 Clusters Creation.

In this section we show how we manage to cluster points, according to the object instance they belong using semantic information.

Panoptic segmentation: Unlike most recent works [Hosseinzadeh et al., 2019; S. Yang and Scherer, 2019a] we do not consider object bounding boxes as input but rather the panoptic segmentation of the image. Panoptic segmentation is a combination of semantic segmentation where each pixel is classified in a given class and instance segmentation where multiple objects of the same class are segmented separately. While panoptic segmentation is harder and takes longer to obtain it allows us to naturally know which keypoints belong to detected objects. Indeed while bounding boxes give the coarse location and size of an object in the image they do not separate the object from the background within the box. Hence a refinement process is required [Huang et al., 2020]. Moreover contrary to object detection CNN, panoptic segmentation networks, like semantic segmentation networks, are not limited to objects and can segment whole areas in the image, such as floor, which correspond to the global structure of the scene. However contrary to semantic segmentation networks, panoptic segmentation separates multiple instances of a single class, allowing to treat each object separately.

The downside of the additional information brought by panoptic segmentation is its complexity, while object detectors can easily process tens of images per second [Redmon et al., 2016], most recent panoptic segmentation networks run only at 10 to 20 fps. However we do not need to segment images at frame rate. As shown in [McCormac et al., 2017], segmenting frames with a low frequency leads to a small drop in mapping segmentation accuracy while allowing the SLAM to run in real time. Moreover panoptic segmentation is an active field of research and real time networks can be expected in the near future.

To represent the panoptic segmentation network we define a function $g_\theta(\mathbf{I}_i) \rightarrow \mathbf{P}_i, \mathbf{L}_i$ which, given an RGB image at time i, $\mathbf{I}_i$ and θ, the network parameters, yields a probability map $\mathbf{P}_i \in [0,1]^{W \times H \times C}$ [1]. Thus for each pixel $\mathbf{x} = (u, v)$ in the image we can obtain a probability distribution $(\mathbf{P}_i(u, v, 1), ..., \mathbf{P}_i(u, v, C))$ where $\mathbf{P}_i(u, v, c)$ corresponds to the probability that this pixel belongs to class c. The second output of the network is the instance map $\mathbf{L}_i \in \mathbb{N}^{W \times H}$ in which each object is segmented and given a unique id.

1. W and H represent the width and height of an RGB image, C is the total number of semantic classes.

One problem of panoptic segmentation however is that the instance map is not temporally consistent, meaning that we can not simply rely on the object ids to create the map. To solve this problem we propose a 2 stages strategy: at the level of the segmentation network and in the SLAM. First, for each detected instance in $\mathcal{L}^i$ we compute the Intersection Over Union (IOU) with all instances of the same class in $\mathcal{L}^{i-1}$ and $\mathcal{L}^{i-2}$ and take its maximum to track the id. We consider an instance to be well tracked if the IOU is above a threshold $\tau_{i-1} = 0.6$ for $\mathcal{L}^{i-1}$ and $\tau_{i-2} = 0.4$ for $\mathcal{L}^{i-2}$. A similar approach can also be applied by propagating the 2D id using optical flow, which requires more computations. Using our strategy, clusters are well defined. However when a cluster that has left the camera field of view reappears in the image, the segmentation network creates a new instance, which leads to the creation of a new cluster. To avoid creating infinitely many clusters and propagate the cluster id we define a merging function to fuse clusters together when the distance between their centroid is lower than a threshold τ_{merge} and more than 80% of clusters points descriptors match. This approach can also be robustified by reprojecting cluster points in 2D and assigning them the id of the segment in which they fall. The combination of those approaches in 2D and 3D allows us to create a sparse consistent map.

Clusters creation: Following the 2D segmentation, we define a function $f(\{\mathbf{P}_i\}, \{{}^i\mathbf{x}\}) \rightarrow \mathbf{p}^{3D}$ where $\{\mathbf{P}_i\}$ is a set of probability maps at different times, $\{{}^i\mathbf{x}\}$ is a set of keypoints corresponding to one 3D point ${}^w\mathbf{X}$ and $\mathbf{p}^{3D} = (p_1, ..., p_C)$ is its probability distribution. This function is the fusion of multiple observations and can be written using Bayes rule, which was inspired by [McCormac et al., 2017] that we adapt to work with our sparse monocular approach as it is mostly used by dense SLAM systems using RGB-D inputs.

$$p_c = \mathbb{P}(c|\{\mathbf{P}_i\}) = \frac{1}{Z}\mathbb{P}(c|\{\mathbf{P}_{i-1}\})\mathbf{P}_i(u, v, c) \tag{4.1}$$

where c is the class label of ${}^w\mathbf{X}$ and Z is a normalization factor. Hence f allows us to obtain a semantic map $\mathcal{M} = \{({}^w\mathbf{X}, \mathbf{p}^{3D}, c^*, l)_j\}$, where each point ${}^w\mathbf{X}$ has a probability distribution $\mathbf{p}^{3D}$, an id l extracted from the instance map $\mathbf{L}^i$ as well as a semantic class $c^* = \underset{c}{\text{argmax}}\, \mathbf{p}^{3D}$. This fusion allows the map to be temporally consistent, even if the panoptic segmentation is noisy.

Using this semantic map we can define a clustering function $h(\mathcal{M}) \rightarrow \mathcal{O}$ where $\mathcal{O}$ is a partition of $\mathcal{M}$ in K clusters $\mathcal{O} = \{O_{k,k\in[1,K]}\}$. This function groups points according to

their semantic class and instance. Each cluster can be defined as $O_k = \{\{^w\mathbf{X}\}, c_k\}$ where $\{^w\mathbf{X}\}$ is the position of a set of points belonging to the cluster and c_k is the cluster class.

4.3.2 Map Optimization from structure estimation.

Structure estimation: Most man-made objects can be approximated with a more or less complex geometrical model, from a simple plane or box to the exact 3D model of the object. The advantage of approximating objects is twofold. First, we can add constraints to the optimization process to improve pose estimation. Second, we obtain a more physically accurate representation of the world by understanding the structures within it, contrary to recent methods that use quadrics to represent objects, which only roughly represent the spatial extent of objects but not their shape [Hosseinzadeh et al., 2019; Nicholson et al., 2018]. In our work as an example we propose to model some objects using planes. Not only do we model large surfaces such as tables, walls or floor but we also model small objects like keyboards and books. While this hypothesis is more restrictive than using quadrics we argue that it stays highly generic as we estimated that about 25% of classes from the COCO dataset can be represented with planes. Planes are represented using the classical 4D vector $\pi = (a, b, c, d)^\top$ with $||\pi||_2 = 1$ [Kaess, 2015] and planar points $\bar{\mathbf{X}}$ in homogeneous coordinates satisfy the following equation:

$$\pi^\top \bar{\mathbf{X}} = 0. \tag{4.2}$$

Contrary to most planar SLAM systems we do not need to use multiple specific CNNs [Hosseinzadeh et al., 2019] or depth [Kaess, 2015] to estimate plane parameters, which limits the applicability of those systems. Instead for each a priori planar cluster, we fit a plane using the triangulated 3D points of this cluster in the world coordinates system. This is done using an SVD inside a RANSAC loop to make the estimation more robust to wrong classification and triangulation similarly to [Arndt et al., 2020]. However contrary to [Arndt et al., 2020] we are not limited to simple scenes with few very large planes. Indeed as we create semantic clusters we are able to fit planes even for small specific objects and thus we can apply our approach in a wider variety of scenes. Moreover the fitting procedure is made easier by the clustering and be done in real-time compared to the 5 fps limitation in [Arndt et al., 2020].

To avoid creating wrong planes that would corrupt the map, a plane is accepted if it is

supported by enough inliers, which depends on the cluster class. For example, keyboards require at least 50 points to be inliers.

Map optimization: One way to add a structure constraint is to use a lagrangian multiplier [Nocedal and Wright, 2006], however this adds parameters that need to be estimated, thus we chose to include the constraint as a regularizer as can be seen in equation (4.3).

$$\begin{aligned} {}^{c_i}\mathbf{T}_w{}^*, {}^w\mathbf{X}^* = \underset{{}^{c_i}\mathbf{T}_w, {}^w\mathbf{X}_j}{\arg\min} \sum_{i,j} \rho(||{}^i\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^w\mathbf{X}_j)||_{\Sigma_{i,j}}) \\ + \sum_k \sum_{j \in O_k} \rho(||\pi_k^\top {}^w\bar{\mathbf{X}}_j||_\sigma) \end{aligned} \tag{4.3}$$

where $||\pi_k^w \bar{\mathbf{X}}_j||$ is the 3D distance between the 3D point ${}^w\mathbf{X}_j$ (in homogeneous coordinates) and the plane π_k which corresponds to the cluster O_k, σ corresponds to its uncertainty, and ρ is a robust cost function (in our case the Huber loss).

Contrary to [Arndt et al., 2020] we do not project 3D points in their plane as some clusters may not be perfectly planar. However the strength of the constraint can be tuned by changing the value of σ. Moreover contrary to [Arndt et al., 2020] we do not optimize the planes, treating them as landmark but only use them as constraints on the map structure.

We optimize this equation using the framework g2o [Kümmerle et al., 2011]. We build a classical BA graph. Then we add an unary constraint to each point belonging to a planar cluster. We consider those points to be outliers if their error is greater than the 95^{th} percentile of a one-dimensional Chi-squared distribution. To account for points that would be outside of the local BA when a plane is fitted we chose to apply a global planar bundle adjustment after plane fitting. Furthermore to account for segmentation noise, we remove from the cluster points that are too far from the plane.

To optimize equation (4.3) we need to compute its Jacobian. It is composed of the derivatives of the reprojection error with respect to camera poses and 3D points, which is the same as in classical BA [Dellaert, 2014] and of the derivatives of the point-plane error with respect to points position. This derivative can be simply computed as:

$$\frac{\partial \pi_k^\top {}^w\bar{\mathbf{X}}_j}{\partial {}^w\bar{\mathbf{X}}_j} = \pi_k^\top \tag{4.4}$$

4.4 Experiments

In this section we first present the implementation details of our approach, then we show on sequences from the TUM and the KITTI dataset [Geiger et al., 2012; J. Sturm et al., 2012] the validity of our method.

4.4.1 Implementation details

S^{3}LAM runs at 20 fps and is based on the state of the art ORB-SLAM2 [Mur-Artal and Tardós, 2017]. We do not take into account the time needed for the inference of the segmentation network as it depends on the choice of the model and on the GPU used. However we note that from detectron [Y. Wu et al., 2019] model zoo, the inference time of panoptic segmentation networks varies from 53 to 98 ms with their setup, making it close to real time. The optimization is done using the optimization framework g2o [Kümmerle et al., 2011]. All the experiments are performed on a desktop computer with an Intel Xeon @3.7GHz with 16 Gb of RAM and an Nvidia RTX2070. The value of the point-plane uncertainty σ is set constant to 100, which balances both errors as the value of the point-plane distance is around 10^{-2} m while the reprojection error is around one pixel.

Datasets. Our approach is evaluated on sequences from the TUM RGB-D dataset [J. Sturm et al., 2012] which provides a set of RGB frames associated to their groundtruth pose. We also show that our approach can work in larger scale and in outdoors scenes by evaluating it on sequences from the KITTI raw dataset [Geiger et al., 2012]. This dataset contains sequences obtained using a camera mounted on a car in a wide variety of scenes. We evaluate on 4 sequences in which the road can be segmented and is planar and in which few objects are moving.

Metrics. To account for the inherent stochasticity of ORB-SLAM2 we run each sequence 10 times and report the median of the RMSE of absolute trajectory error (ATE, defined in [J. Sturm et al., 2012]) in the tables below. As our experiments are performed in a monocular setting, we scale and align the estimated trajectories with the ground truth as in [Mur-Artal et al., 2015].

4.4.2 Impact of the planes constraints on pose error

In this section we study the impact of adding planar constraints to the classical BA formulation, using only planes robustly estimated from the 3D semantic map. We report

the results of our experiments in table 4.1. We compare our approach to: the base system of S^{3}LAM, ORB-SLAM2 [Mur-Artal and Tardós, 2017] and both the monocular and RGB-D approaches of Hosseinzadeh et al. [Hosseinzadeh et al., 2018; 2019] that report the greatest number of experiments and use both quadrics and planes. We also compare against the plane based approach of [Arndt et al., 2020] that report experiments on a few strongly planar sequences from the TUM dataset, as their code is not available we were not able to run experiments on new sequences.

As we can see, planar constraints improve camera localization in most cases over ORB-SLAM 2.

The most important improvement is obtained on almost perfectly planar scenes like fr1_floor and fr3_nostr_text_near which allow to easily fit planes. As we could expect using large planes gives better ATE improvements, the line fr3_nostr_text_near (merged books) in table 4.1 shows our approach using a single plane for all book clusters, thus treating them as a single large cluster. An interesting way to improve our approach would thus to add plane-plane constraints to increase the spatial extent of small planes. The sequences fr3_nostr_text_near and fr3_nostr_text_near (merged books) are run with the loop closure deactivated to better show the impact of our approach. The activation of loop closure is shown in the row fr3_nostr_text_near (loop). The sequences fr2_xyz and fr2_desk are the most challenging for our approach as the main planes corresponding to the table and the floor are not well segmented and cluttered, yielding to a noisy estimation. Compared to the CNN based approach of [Hosseinzadeh et al., 2019] we can see that we obtain better results for planar scenes, however in scenes in which planes are not well segmented, using quadrics brings a more important improvement, which can be related to the fact that object detection is less noisy than panoptic segmentation. Compared to the approach of [Arndt et al., 2020] we obtain similar results, with almost equal means, however we argue that our approach is more generic as we are not limited to mostly planar scenes and we can run it in near real time. We argue that those results and the reported means, show that our approach is generic as it improves camera pose estimation in a wide variety of scenes with a preference for planar scenes, while other approaches focus either on scenes containing objects or on perfectly planar scenes. To further demonstrate that our approach is generic we show in table 4.2 the evaluation of our approach in a large scale scene, on the KITTI raw dataset, using the class *road* as a plane. As we can see our approach works in large outdoors scenes and we improve camera pose estimation compared to ORB-SLAM2 [Mur-Artal and Tardós, 2017].

Table 4.1 – Comparison of the ATE (mm) of our our approach against state of the art on the TUM dataset. (H1) denotes [Hosseinzadeh et al., 2018], (M) [Mur-Artal and Tardós, 2017], (H2) [Hosseinzadeh et al., 2019], (A) [Arndt et al., 2020]

Sequence	(H1) (RGB-D)	(M)	(H2) (RGB) w. planes	(H2)(RGB) w. quadrics	(A)	Ours
fr1_xyz	9.6	9.2	10.3	10.0	-	**8.8**
fr1_floor	13.8	18.1	16.9	-	-	**14.7**
fr1_desk	15.3	13.9	12.9	**12.6**	-	13.2
fr2_xyz	3.3	2.4	**2.2**	**2.2**	-	2.4
fr2_desk	12.0	8.0	7.3	**7.1**	-	7.8
fr3_nost_text_near	-	20.3	-	-	-	**15.3**
fr3_nost_text_near (merged books)	-	20.3	-	-	-	**13.5**
fr3_nost_text_near (loop)	10.9	14.5	-	-	11.4	**10.9**
fr3_str_text_near	-	14.0	-	-	**10.6**	11.2
fr3_str_text_far	-	10.6	-	-	**8.8**	9.2
fr1 mean	12.9	13.9	13.4	-	-	**12.2**
fr2 mean	7.7	5.2	4.8	**4.7**	-	5.1
fr3 mean	-	13.0	-	-	**10.3**	10.4

Table 4.2 – Comparison of the ATE (cm) of our approach against ORB-SLAM2 on the Kitti raw dataset.

sequence	ORB-SLAM 2 [Mur-Artal and Tardós, 2017]	Ours
0926-0011	17.7	**15.5**
0926-0013	18.0	**7.5**
0926-0014	76.2	**64.5**
0926-0056	49.8	**49.3**
mean	40.4	**34.2**

Table 4.3 – Maximum, minimum and average values of angles between plane normals. The closer to 0 the better.

Sequence	max. angle	min. angle	med. angle
fr1_desk	3.6°	2.4°	2.9°
fr3_nost_text_near	1.4°	0.8°	0.8°
fr3_nost_text_near (merged)	0.0°	0.0°	0.0°

To further show that the estimated map structure is coherent, we compute the angle between pairs of normals that are supposed to be parallel at the end of a sequence (like pairs of books or books on a table for example). The minimum, maximum and median angles between normals are shown in table 4.3. As we can see the estimated normals are pairwise coherent. Moreover the larger the planes are, the more points are extracted and the better the estimation is as visible in the last two lines of the table.

4.4.3 Qualitative analysis of S^{3}LAM

We show in figure 4.3 some qualitative examples of maps obtained using our clustering approach compared to maps obtained using ORB-SLAM2 [Mur-Artal and Tardós, 2017]. The goal of this figure is to show that the clusters and planes are well defined, to better see the effect of planes on the map quality we refer the reader to figure 4.4. To construct these maps we used the following classes: table, keyboard, book. As we can see every object present in the scene has been uniquely clustered, leading to a more comprehensible and higher level map. We also show for each planar cluster the estimated planes, corresponding to the objects table, keyboard and book. Our approach yields a more physically accurate representation of the world, which can be easily used for augmented reality or robotics applications. In figure 4.4 we show a comparison of the map obtained with ORB-SLAM2 and the map obtained using our planar BA. As we can see at the bottom, the lower part of the map corresponding to the floor (visible in orange in the bottom part of the map) is much more planar and coherent using our approach.

4.5 Conclusion

In this chapter we presented our new monocular semantic SLAM system called S^{3}LAM. Our method uses the 2D panoptic segmentation of a sequence of RGB images to create

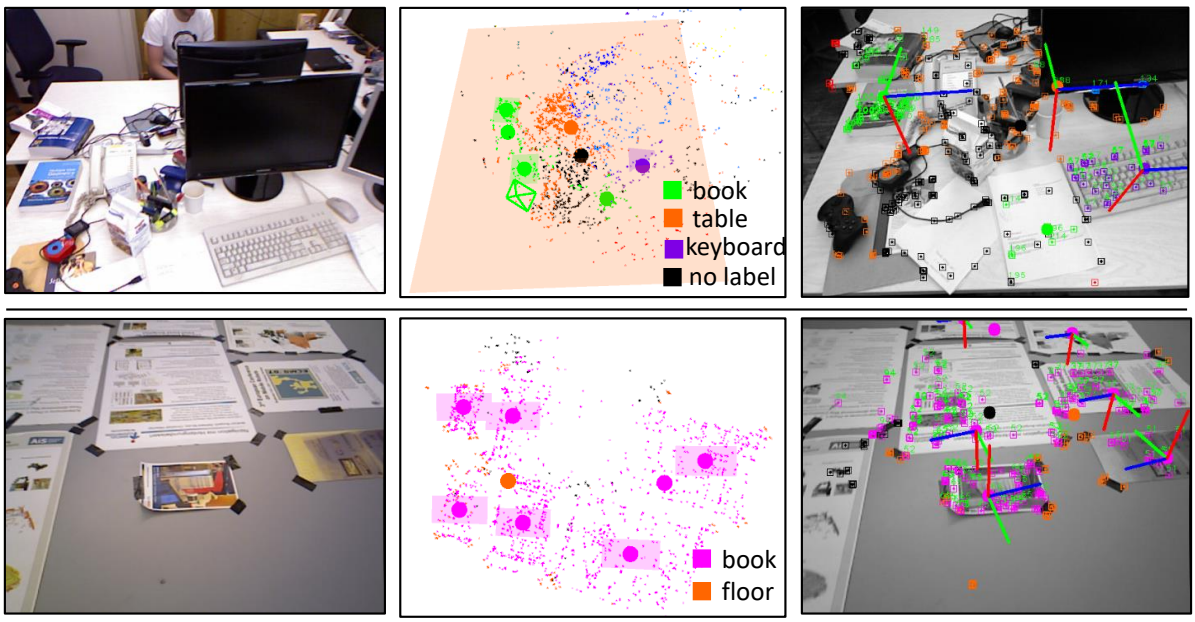

Figure 4.3 – Examples of maps created by our approach for 2 sequences from [J. Sturm et al., 2012]. From left to right: RGB input image from the sequence, S^{3}LAM map, frame with planes normals projected in red, in plane vectors are shown in blue and green. Keypoints are as well shown with a color corresponding to their class.

clusters of 3D points according to their class and instance. This clustering allows us to robustly estimate the structure of some clusters and modify the Bundle Adjustment formulation with structural constraints. We show on sequences from several public datasets that our approach leads to an improvement of camera pose estimation. Our system is more generic than approaches based on specific object detectors as it works at the level of object categories. The main limit of our approach is that it relies on the static scene assumption. Moving objects are likely to decrease the accuracy of camera pose estimation and to corrupt the map. Furthermore moving objects can not be tracked, contrary to object pose estimation algorithms. We solve this problem by further developing our scene graph representation in the following chapter.

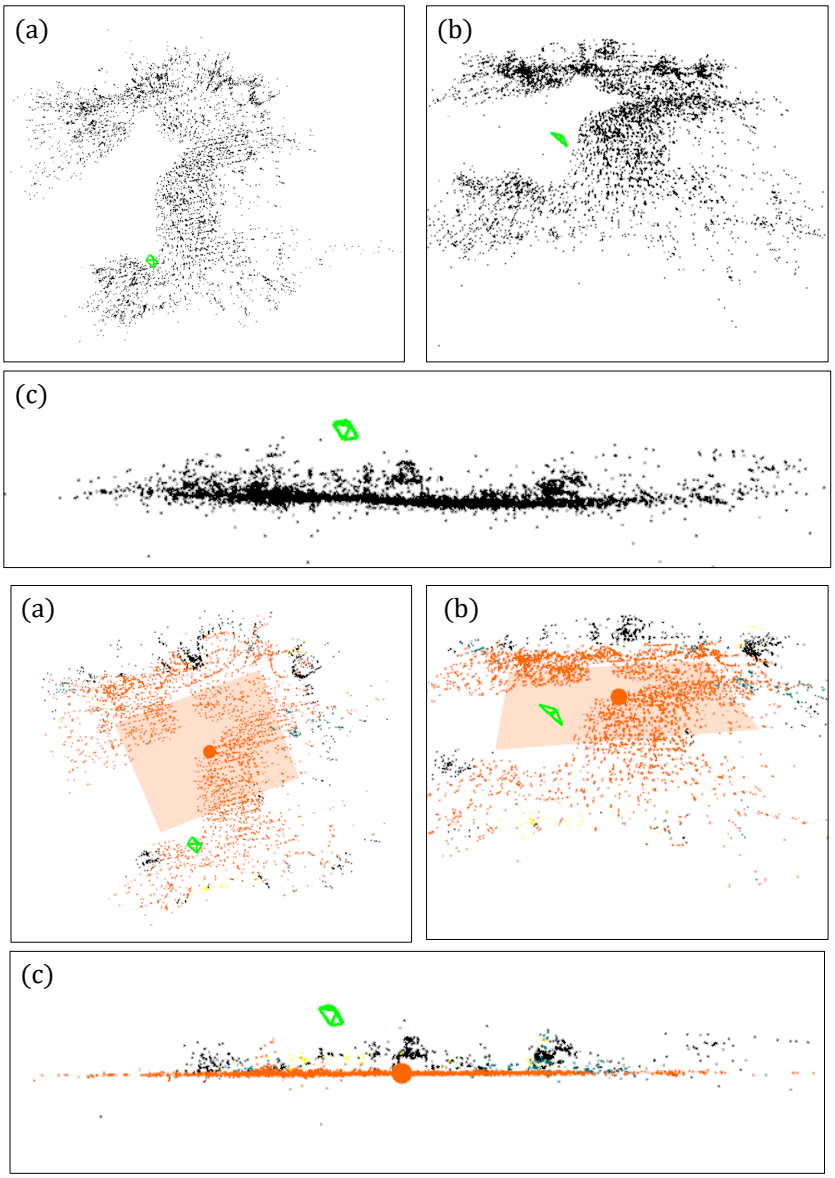

Figure 4.4 – Examples of map created by ORB-SLAM2 [Mur-Artal and Tardós, 2017] (top) and our approach (bottom) for the sequence fr1_floor. (a) Map top view, (b) Map ¾ view, (c) Map side view. As we can see on the side view the floor is more planar with our approach. Black points that are outside of the plane are points that have not been segmented as "floor" and thus do not undergo the planar constraint.

This chapter was published in:

- Mathieu Gonzalez, Eric Marchand, Amine Kacete and Jérome Royan, "S^{3}LAM: Structured Scene SLAM", *in: IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2022

- Mathieu Gonzalez, Eric Marchand, Amine Kacete and Jérome Royan, "S^{3}LAM: SLAM à Scène Structurée", *in: Groupe de Recherche et d'Etudes de Traitement du Signal et des Images (GRETSI)*, 2022

Chapter 5

TwistSLAM: Constrained SLAM in Dynamic Environment

Contents

In this chapter we propose to go beyond S^{3}LAM by developing a SLAM system that can deal with dynamic objects. Moving objects are present in most scenes of our life. However let us recall that they can be very problematic for classical SLAM algorithms that assume the scene to be rigid. This assumption limits the applicability of those algorithms as they are unable to accurately estimate the camera pose and world structure in many scenarios. Some SLAM systems have been proposed to detect and mask out dynamic objects, making the static scene assumption valid. However this information can allow the system to track objects within the scene, while tracking the camera, which can be crucial for some applications. In this chapter we present TwistSLAM a semantic, dynamic, stereo SLAM system that can track dynamic objects in the scene. Based on S^{3}LAM, developed in chapter 4, our algorithm creates clusters of points according to their semantic class. It uses the static parts of the environment to robustly localize the camera and tracks the remaining objects. We propose a new formulation for the tracking and the bundle adjustment to take into account the characteristics of mechanical joints between clusters to constrain and improve their pose estimation. We evaluate our approach on several sequences from a public dataset and show that we improve camera and object tracking compared to state of the art.

A video explaining the approach and showing object tracking examples can be found at https://youtu.be/2uxEQGjiuXQ.

5.1 Introduction

Several algorithms such as [Engel et al., 2014; Mur-Artal and Tardós, 2017] have been very successful in the past few years to solve the problem of SLAM, however they often rely on the static scene assumption. Let us recall that this hypothesis assumes that the world is a single rigid body and thus that no object can move within it. This assumption, which is false in most real world scenes limits the scenarios in which a SLAM algorithm can be used. Classical SLAM systems such as [Mur-Artal and Tardós, 2017] try to alleviate this assumption using robust estimators, allowing them to flag moving parts as outliers. However as soon as the number of moving points is too important, the estimated camera pose accuracy decreases. This makes this approach unsuitable for some scenes (e.g. crowded or urban scenes). Some systems [Bescos et al., 2018; C. Yu et al., 2018] have been proposed to detect and mask out dynamic objects in images, thus making the static scene assumption valid. However some recent approaches [Bescos et al., 2021; Huang et al., 2020; Runz et al., 2018; J. Zhang et al., 2020] argue that moving objects represent valuable information that can be necessary for some applications. Most recent approaches trying to solve both SLAM and object tracking have used semantics as an additional source of information. Semantic knowledge can indeed be beneficial to SLAM [Cadena et al., 2016; Rosinol et al., 2020] as it contains information about the class dynamicity which is higher level information than simple 3D points.

In this chapter we present a stereo SLAM system called TwistSLAM as we estimate objects twists to track them and consider that objects are linked to each others through mechanical joints, similarly to joints linking different parts of a robot. An illustration of our algorithm is visible figure 5.1: the camera pose is estimated simultaneously with all moving objects in the scene and the map structure (here the plane of the road) constrains the movement of objects. Our approach is based on ORB-SLAM2 [Mur-Artal and Tardós, 2017] and S^{3}LAM [Gonzalez, Marchand, et al., 2021]. In our work we use semantic information to build a map of clusters corresponding to objects in the scene. The clustering of the scene allows us to estimate the pose of the camera using static clusters only such

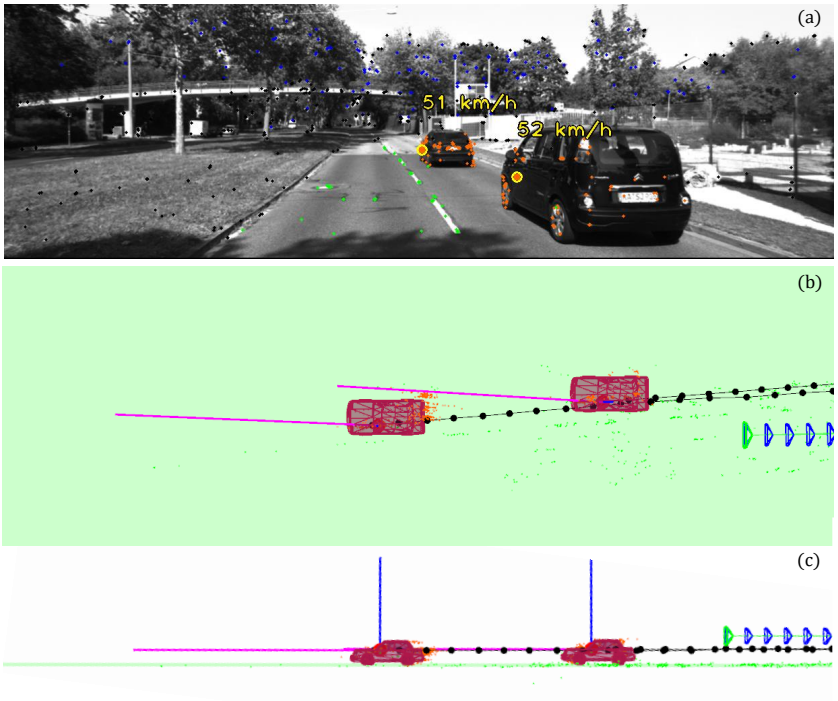

Figure 5.1 – Our approach allows us to track objects in the scene such as cars. Here we can see: (a) the frame with semantic points and tracked clusters (orange cars) with their estimated speed. (b) a map top view with tracked clusters (orange cars), clusters trajectories (black spheres and lines), clusters twists (blue and purple lines), road points and plane (in green) and camera trajectory (green and blue frustums). (c) Map side view. The rotation part of the twists (blue lines) is perpendicular to the road plane, the translation part (purple lines) is parallel to the plane. Only points from the classes *car* and *road* are shown and the rotation part of twists has been magnified for visualization.

as *road* or *house*. The other clusters that can be dynamic are tracked and their pose is updated in the map through the estimation of twists that represent their velocity. Most SLAM systems that can track dynamic objects directly estimate their pose through the minimization of a reprojection error function [Bescos et al., 2021; Huang et al., 2020] or with 3D points registration [Runz et al., 2018]. Doing so the estimated pose of an object has 6 degrees of freedom. It is obvious that this does not correspond to reality, for example a car has only 3 degrees of freedom, 2 translations in the road plane and 1 rotation around its normal, hence its pose should be constrained. Our goal is thus to remove degrees of freedom corresponding to physically unfeasible movements. To do so we chose to represent those constraints as mechanical joints which makes our approach highly generic. A mechanical joint between clusters constrains the estimated twist of a dynamic cluster by

blocking some of its degrees of freedom thus reducing the effect of noise on the estimation. Once an object twist has been estimated it can be used to update the object pose which enables object tracking. Mechanical joints have already been used in the past [J. Sturm et al., 2010; 2011], [Comport et al., 2007] for object tracking. However to the best of our knowledge our approach is the first to introduce them in a dynamic SLAM system.

The object poses can then be tightly refined with camera poses and 3D points within a bundle adjustment that also applies mechanical joints constraints.

Our contributions presented in this chapter are:

- A semantic SLAM system that can robustly estimate the pose of a camera in static as well as dynamic scenes.
- A stereo SLAM framework that can track multiple moving objects in the scene.
- A new formulation for both the tracking and bundle adjustment that takes into account the characteristics of mechanical joints between objects in the scene.
- An evaluation of our approach on several sequences from a public dataset which demonstrates the benefits of our method in terms of object poses and velocities and camera pose estimation accuracy.

The rest of the chapter is described as follows. First we rapidly recall related work on dynamic classical and semantic SLAM. We also rapidly recall the mathematical concepts that will be used in this chapter. Following, we describe our approach to build a semantic map of clusters, estimate the pose of a camera in a dynamic scene, track moving clusters within the scene while using joint constraints to improve object tracking and refine all estimations with a bundle adjustment. Finally we demonstrate the benefits of our approach on multiple sequences from a public dataset.

5.2 Related work: Semantic dynamic SLAM

In this section we rapidly recall a few dynamic dynamic SLAM systems already presented in chapter 2 and present additional work.

To tackle the problem of dynamic objects, some methods propose to roughly estimate the pose of the camera and robustly find outliers in the scene or in the image. Outliers are then removed or downweighted and the camera pose is refined. For example [S. Li

and Lee, 2017] proposes a direct approach based on the alignment of depth edges using an ICP scheme. For each point they robustly estimate a staticity confidence score which downweights dynamic objects and an intensity assisted ICP robustly refines the pose using those weights. [Y. Sun et al., 2017] segments dynamic objects in 2D with a 3 stages strategy. First dynamic objects are roughly detected by computing the difference between the current frame and its warped predecessor. This difference is then enhanced and cleaned with a particle filter. Finally the depth image is quantized and clusters which show a high concentration of particles are masked out.

DynaSLAM [Bescos et al., 2018] uses semantic information to segment a priori moving objects which are not used for tracking and mapping. The segmentation is refined using the depth. This approach improves camera localization in dynamic scenes but deteriorates it when a priori moving objects are in reality static such as parked cars. MaskFusion [Runz et al., 2018] is one of the first semantic dynamic SLAM that can track objects. Inspired from [Rünz and Agapito, 2017] it makes use of 2D masks inferred by Mask-RCNN [K. He et al., 2017] to detect objects in the scene and tracks them using both photometric and geometric information from an RGB-D camera. DynaSLAM II [Bescos et al., 2021] uses semantic information to detect objects. Object 3D points are represented in the object reference frame and used to estimate the object pose at all time by minimizing their reprojection error. ClusterVO [Huang et al., 2020] is similar to [Bescos et al., 2021], but they consider object detection (i.e. 2D bounding boxes) as input that is much faster to infer than dense masks. They also apply a cleaning procedure to improve dynamic keypoints matching and make sure that 2D points do not come from the background of the bounding box. VDO-SLAM [J. Zhang et al., 2020] proposes to use optical flow to track features extracted more densely than other systems, which allows them to obtain a more precise object pose estimation. Furthermore the optical flow and the object and camera motions are tightly refined.

5.3 Notations

In this section we recall formulas and notations defined in chapter 1. The pose of the i^{th} frame, of a rigid body o associated with the coordinate frame $\mathcal{F}_o$ in the world coordinate

frame $\mathcal{F}_w$ can be represented by the homogeneous matrix

$$ {}^{w}\mathbf{T}_{o_i} = \begin{pmatrix} {}^{w}\mathbf{R}_{o_i} & {}^{w}\mathbf{t}_{o_i} \\ 0 & 1 \end{pmatrix} \quad \in \quad \mathrm{SE}(3) \tag{5.1} $$

Let us recall that this matrix maps points in the object frame ${}^{o}\mathbf{X}$ to points in the world frame ${}^{w}\mathbf{X}$ according to the following equation: ${}^{w}\mathbf{X} = {}^{w}\mathbf{T}_{o}{}^{o}\mathbf{X}$. The velocity of a moving object can be represented using a twist $\boldsymbol{\xi}$ defined as

$$ \boldsymbol{\xi} = \begin{pmatrix} v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{pmatrix}^{\top} = \begin{pmatrix} \mathbf{v} & \boldsymbol{\omega} \end{pmatrix}^{\top} \in \mathbb{R}^6 \tag{5.2} $$

where the first 3 components $\mathbf{v} = (v_x, v_y, v_z)^{\top} \in \mathbb{R}^3$ denote the translational velocity and the other components $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^{\top} \in \mathbb{R}^3$ represent the rotational velocity. We denote ${}^{w}\boldsymbol{\xi}_{o_i}$ the twist corresponding to the velocity of the object o at frame i expressed in the world coordinate frame. Similarly, ${}^{o_i}\boldsymbol{\xi}_{o_i}$ is the velocity of the object o at frame i expressed in its own coordinate frame. The exponential map can be used to recover the pose of the object o moving according to the twist ${}^{w}\boldsymbol{\xi}_{o_i}$ from its initial pose at frame i ${}^{w}\mathbf{T}_{o_i}$ to its next pose at frame $i+1$ ${}^{w}\mathbf{T}_{o_{i+1}}$ using the following formula:

$$ {}^{w}\mathbf{T}_{o_{i+1}} = \exp({}^{w}\boldsymbol{\xi}_{o_i}\delta t_i){}^{w}\mathbf{T}_{o_i} = {}^{w}\mathbf{T}_{o_i}\exp({}^{o_i}\boldsymbol{\xi}_{o_i}\delta t_i) \tag{5.3} $$

The adjoint map ${}^{w}\mathbf{V}_{o_i} \in \mathbb{R}^{6\times 6}$ links twists in different frame coordinates according to:

$$ {}^{w}\boldsymbol{\xi}_{o_i} = {}^{w}\mathbf{V}_{o_i}\,{}^{o_i}\boldsymbol{\xi}_{o_i} \tag{5.4} $$

it can be computed using the relative pose ${}^{w}\mathbf{T}_{o_i}$ between $\mathcal{F}_{o_i}$ and $\mathcal{F}_w$:

$$ {}^{w}\mathbf{V}_{o_i} = \begin{pmatrix} {}^{w}\mathbf{R}_{o_i} & [{}^{w}\mathbf{t}_{o_i}]_{\times}\,{}^{w}\mathbf{R}_{o_i} \\ 0 & {}^{w}\mathbf{R}_{o_i} \end{pmatrix} \tag{5.5} $$

For simplicity we will consider in the remainder of this chapter that $\delta t = 1$ without loss of generality.

5.4 TwistSLAM: Constrained SLAM in Dynamic Environment

In this section we present our approach. Let us recall that our general idea is to represent the world as a graph of semantic clusters, which is similar to a scene graph, presented in the introduction of this manuscript and which can be seen in figure 5.2. The vertices of the graph correspond to objects in the scene and the edges to physical links that exist between objects. Our goal is to estimate the pose of the camera and the pose of every moving object while using mechanical joints between objects to improve those estimations. For example both clusters *car* in our graph are linked to the road with a planar constraint that allows only 3 degrees of freedom: a rotation around the normal of the plane and 2 translations within the plane. Such simple representations allow us to be highly generic as, for a given semantic class, we only need to define its static parent and the type of mechanical joint.

The pipeline of our approach is presented in figure 5.3. Using semantic information we create clusters of points corresponding to objects in the scene. We then use static semantic clusters (e.g. road, floor, house) to robustly track the camera, even in dynamic scenes. Then, we match keypoints corresponding to dynamic objects (e.g., car, bike,...) to either track them or triangulate new 3D points using stereo information. All poses estimations from the camera and the objects are then refined with static and dynamic 3D points with a bundle adjustment process.

The main novelty of our approach comes from the fact that we optimize the velocities of the dynamic objects rather than their pose and constrain the velocities according to mechanical joints between objects. This approach is highly generic as we only need to define a handful of joints (that correspond to normalized joints in mechanics) and a list of semantic classes pairs for each joint (e.g. the wall-door joint corresponds to a revolute joint, the car-road joint corresponds to a planar joint). As we will show latter, it allows us to remove displacements along directions that are not physically possible (e.g. a car translating vertically) and thus that correspond to optimization errors. This allows us to obtain a more precise estimation of the dynamic object poses.

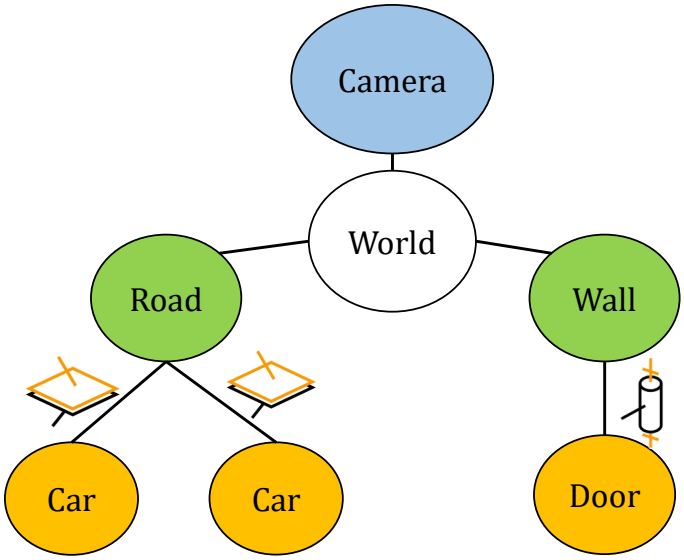

Figure 5.2 – Example of semantic graph: dynamic clusters are linked to static parent clusters with mechanical joints such as *planar* or *revolute*.

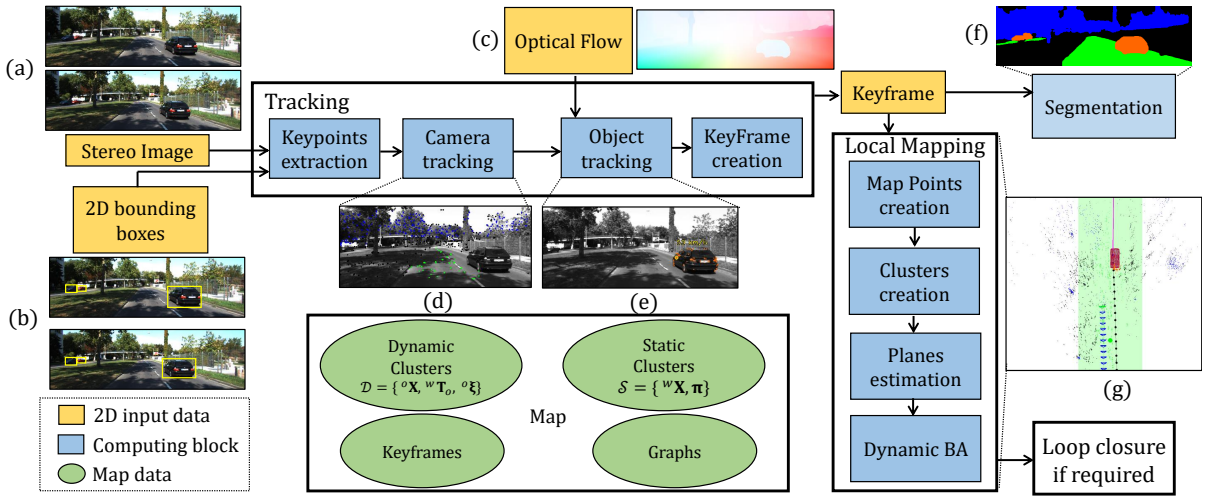

Figure 5.3 – The pipeline of our approach: static keypoints are extracted from a stereo image (a) and used for camera tracking (d), dynamic keypoints are extracted from bounding boxes within the stereo images (b) and matched using optical flow (c) with the previous frame to track dynamic objects (e). The keyframe is then segmented (f) to create new semantic map points and clusters. Finally the object and camera poses are jointly refined with the dynamic and static map points in a BA (g).

5.4.1 Creating Clusters from panoptic segmentation

Most recent semantic dynamic SLAM systems use either an object detection or an instance segmentation algorithm. Working in the continuity of S^{3}LAM [Gonzalez, Marchand, et al., 2021] we chose to estimate the panoptic segmentation (obtained using detectron 2 [Y. Wu et al., 2019]). This allows us to know the semantic class of each pixel in the image and to give a unique id to each object in the image. Similarly to [Gonzalez, Marchand, et al., 2021] we fuse multiple 2D observations of a single 3D point to obtain its class and id. Doing so we obtain a semantic map, which allows us to create a set of K clusters $\mathcal{O} = \{O_k, k \in [1, K]\}$. A cluster is a set of 3D points corresponding to a single object in the scene. Points are grouped according to their class and instance id. The set of clusters can be expressed as the set of a priori static clusters $\mathcal{S}$ (such as *road, building, ...*) and the set of a priori dynamic clusters $\mathcal{D}$ (such as *car, bike, human, bus, ...*). As static clusters are fixed, we represent their 3D points $\{{}^w\mathbf{X}\}$ in the world frame. In contrast, each dynamic cluster contains a set of 3D points $\{{}^o\mathbf{X}\}$ expressed in the object coordinate frame, a set of poses $\{{}^o\mathbf{T}_w\}$ and a set of twists $\{{}^w\boldsymbol{\xi}_o\}$ representing the cluster trajectory and velocity through time. For simplicity in the remainder of this chapter we will omit the object index k as its use is straightforward.

5.4.2 Clusters geometry

Our goal is to constrain the velocity of moving clusters according to mechanical joints. To do so we need to estimate the pose of those joints. We propose to do this using the estimated geometry of some clusters. We chose to consider only planar clusters, which allows our approach to be highly generic as planes are common in man-made environment. For clusters corresponding to a priori chosen classes (such as the road or the facade of a building) we estimate a 3D plane $\pi = (a, b, c, d)^\top$ with $||\pi||^2 = 1$, using only its 3D points $\{{}^w\mathbf{X}\}$. The plane follows the following equation: $\pi^\top {}^w\bar{\mathbf{X}} = 0$ and can be estimated using an SVD. To make it robust to outliers (due to segmentation or triangulation errors) we use a RANSAC scheme.

5.4.3 Dynamic SLAM

As we do not know which dynamic objects in the scene are really moving we chose to estimate the camera pose using only static objects. Using points from static clusters we

minimize the following cost function

$$E({}^{c_i}\mathbf{T}_w) = \sum_{j \in \mathcal{S}} \rho(||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j)||_{\Sigma_{i,j}^{-1}}) \tag{5.6}$$

where ${}^{i}\mathbf{x}_j$ is the 2D keypoint corresponding to the observation of ${}^{w}\mathbf{X}_j$ in the i^{th} frame, p is the pinhole camera projection function, ρ is a robust cost function (in our case Huber) [Malis and Marchand, 2006] and $\Sigma_{i,j}$ is the covariance matrix of the reprojection error. Doing so the estimated camera pose does not take into account potentially moving objects, hence it is robust in dynamic scenes. However the estimation can be deteriorated in scenes that contain many potentially moving objects that are in reality static, like for example parked cars. To solve this problem, we chose to estimate the pose of all moving objects and integrate them in the bundle adjustment, so that the velocity of static objects is close to 0 and their points act as static points.

5.4.4 Dynamic data association and keypoints

Dynamic data association is a challenging problem for two reasons: first the combination of the camera and the object movements can produce large displacements in the image space thus needing a large radius search for keypoints matching. Second a large movement can cause an important visual variation of the object in the image (e.g. due to luminosity changes on the object or to viewpoint changes) which makes the matching process more difficult. To overcome those challenges we propose to use the optical flow estimation produced by a CNN (namely RAFT [Teed and Deng, 2020]) to have a good estimate of the keypoints location and reduce the search radius, thus reducing both search time and the probability of false matches.

One problem of object tracking compared to classical SLAM is that dynamic objects usually occupy a small part of the image, meaning that if keypoints are extracted in an uniform way from the image they may be too few to obtain a precise estimation [Bescos et al., 2021; J. Zhang et al., 2020]. To solve this problem we force the keypoint extraction process to keep more keypoints from areas defined by dynamic objects bounding boxes. The keypoints are then used either to create new 3D points with stereo triangulation, which are added to existing clusters or used to create new clusters, or to track the existing cluster as shown in the following section.

5.4.5 Mechanical joints as inter-cluster constraints

Using matches found by the data association process we seek to estimate the pose of dynamic objects in the scene. Our assumption in this work is that many moving clusters can be represented as being linked to a static parent cluster with a specific mechanical joint. There exist 12 normalized joints (ISO 3952) that can be associated with the degrees of freedom they have. For example the planar joint has 3 degrees of freedom: 2 translations in the plane and 1 rotation around its normal, this joint can represent the displacement of a car relative to its static parent, the road. Another example is the revolute joint which has a single degree of freedom corresponding to the rotation around a single axis. In this case the static parent cluster is the wall, the moving cluster is the door and its only possible movements are rotations around the axis of the joint (corresponding to the hinge).

To easily model all types of joints, similarly to [Comport et al., 2007], we propose to decompose the space of twist as the sum of two orthogonal spaces:

$$\mathbb{R}^6 = \mathbb{F}_l + \mathbb{F}_l^\top \tag{5.7}$$

where $\mathbb{F}$ (which stands for freedom) is the space of twists allowed by the mechanical joint l with coordinate frame $\mathcal{F}_l$. In the case of a planar joint with axis z, $\mathbb{F}_l$ is defined as:

$$\mathbb{F}_l = Span(\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^\top) \tag{5.8}$$

where $Span$ is the linear span [Axler, 2014]. In general we note:

$$\mathbb{F}_l = Span(\mathbf{A}_l) \tag{5.9}$$

where $\mathbf{A}_l$ is a basis of $\mathbb{F}_l$. To make the displacement of an object physically accurate, its twists have to lie within the $\mathbb{F}_l$ space. To do so we project the twist from its original space to $\mathbb{F}_l$. This is straightforward as $\mathbb{R}^6$ is Euclidean, the operation projector is a 6×6 matrix defined as:

$$\mathbf{\Pi}_l = \mathbf{A}_l(\mathbf{A}_l^\top \mathbf{A}_l)^{-1}\mathbf{A}_l^\top \tag{5.10}$$

In the example of a planar joint, it is easy to compute that:

$$\mathbf{\Pi}_l = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.11}$$

in that case, a general twist can be projected such that:

$$\mathbf{\Pi}_l \, \boldsymbol{\xi} = \begin{pmatrix} v_x & v_y & 0 & 0 & 0 & \omega_z \end{pmatrix}^\top \tag{5.12}$$

Let us give another example, in the case of a revolute joint of axis z, the $\mathbf{A}_l$ matrix can be written as:

$$\mathbf{A}_l = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^\top \tag{5.13}$$

We can thus compute the projection operator for that case:

$$\mathbf{\Pi}_l = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.14}$$

This projection operator applied to a twist yields the following projected twist:

$$\mathbf{\Pi}_l \, \boldsymbol{\xi} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \omega_z \end{pmatrix}^\top \tag{5.15}$$

which only remaining speed corresponds to a rotation around the z axis, the hinge of the revolute joint. We show in figure 5.4 examples of planar and revolute joints.

As we can see from both examples, the only remaining degrees of freedom of the projected twist are coherent with the joint. Using this new constraint, we can modify the reprojection equation:

$${}^i\mathbf{x}_j = p({}^{c_i}\mathbf{T}_w \exp\left(\mathbf{\Pi}_l {}^w\boldsymbol{\xi}_{o_i}\right) {}^w\mathbf{T}_{o_{i-1}}, {}^o\mathbf{X}_j) \tag{5.16}$$

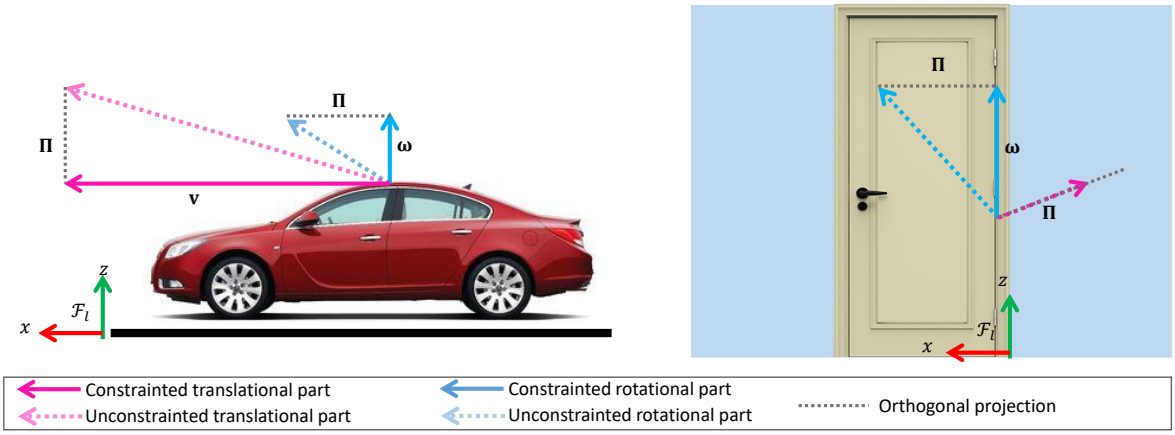

Figure 5.4 – Example of twist projections for a planar joint for a car (left) and a revolute joint for a door (right). Note how the translation part of the twist is completely canceled for the rotational joint, thus superimposing the orthogonal projection and the twist.

However this equation is only true if the twist is expressed in the joint coordinate frame, yet according to (5.3), it is naturally expressed either in the world or in the object frame. To change the coordinate frame of a twist we can use the adjoint map defined in (5.5). Hence the reprojection equation of the j^{th} point in frame i becomes:

$$ {}^{i}\mathbf{x}_j = p({}^{c_i}\mathbf{T}_w \exp\left({}^{w}\mathbf{V}_l \, \mathbf{\Pi}_l \, {}^{l}\mathbf{V}_w \, {}^{w}\boldsymbol{\xi}_{o_i}\right)^{c_{i-1}}\mathbf{T}_w, {}^{o}\mathbf{X}_j) \tag{5.17} $$

where the first pose ${}^{c_0}\mathbf{T}_w$ is initialized with the identity for the rotation and the centroid of 3D points for the translation. In the remainder of this chapter we will note $\mathbf{\Pi} = {}^{w}\mathbf{V}_l \, \mathbf{\Pi}_l \, {}^{l}\mathbf{V}_w$ for simplicity. This equation takes a 3D point in the object frame, transforms it in the world frame using the previous object pose and multiplies it by the exponential of the current twist to get its current position. The twist is expressed in the joint coordinate frame, with the adjoint map, projected using $\mathbf{\Pi}_l$ to keep only the relevant components and expressed again in the world frame with the inverse adjoint map. Doing this we obtain a 3D point in the world frame for frame i with a transformation that perfectly respects the mechanical joint. We then apply the camera pose to obtain the point in the camera coordinate frame, which allows us to project it in the image.

Using the reprojection function we can estimate the twist corresponding to the transformation of a set of object points between frame $i-1$ and i by minimizing the following

error:

$$E({}^{w}\boldsymbol{\xi}_{o_i}) = \sum_j \rho(||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w \exp(\mathbf{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i})\,{}^{w}\mathbf{T}_{o_{i-1}}, {}^{o}\mathbf{X}_j)||_{\Sigma_{i,j}^{-1}}) = \sum_j \rho(||e^j({}^{w}\boldsymbol{\xi}_{o_i})||_{\Sigma_{i,j}^{-1}}) \tag{5.18}$$

where ρ is the Huber robust estimator [Malis and Marchand, 2006] and $\Sigma_{i,j}$ is the covariance matrix of the reprojection error. In [Mur-Artal and Tardós, 2017] the covariance matrix depends on the scale at which the keypoints are observed. In our case we chose to estimate it using the mean absolute deviation (MAD) [Malis and Marchand, 2006] that is a robust estimator of the standard deviation of the reprojection error. We perform the optimization using the Levenberg-Marquardt algorithm on matches found between the current and the previous frame. Then we refine this twist with an approach similar to [Mur-Artal and Tardós, 2017] by projecting map points, transformed with the estimated twist, in the current frame to search for additional matches and obtain a more accurate estimation. The object pose in frame i is then updated as ${}^{w}\mathbf{T}_{o_i} = \exp(\mathbf{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i})\,{}^{w}\mathbf{T}_{o_{i-1}}$. This tracking procedure is repeated for all objects, however they could be done in parallel as the estimations are independent.

5.4.6 Dynamic Bundle Adjustment

The goal of classical bundle adjustment is to refine the camera trajectory and 3D points position estimation. The dynamic bundle adjustment has multiple goals. First, the refinement of the dynamic objects trajectory and their 3D points position, jointly with the camera trajectory and 3D static points position. Second, it allows to link the object and the camera trajectory, indeed if the bundle adjustment did not take into account dynamic objects, only the camera pose would have an impact on the object pose, which would not improve it. By taking into account dynamic points whose position is estimated over time we can use them to refine the camera pose, similarly to static points but with less accuracy since object pose estimation is noisier. Finally, it allows us to apply a soft constrain on twists within a temporal window. Doing so we obtain smoother trajectories and velocities that are more physically plausible.

Our bundle adjustment cost function can be written as follows:

$$E(\{{}^{w}\tilde{\boldsymbol{\xi}}_o, {}^{c}\mathbf{T}_w, {}^{w}\mathbf{X}, {}^{o}\mathbf{X}\}) = \sum_{i,j} e^{i,j}_{stat} + \sum_{i,j} e^{i,j}_{dyna} + \sum_i e^{i}_{const} \tag{5.19}$$

where $e^{i,j}_{stat}$ is the classical static reprojection error:

$$e^{i,j}_{stat} = \rho(||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j)||_{\Sigma^{-1}_{i,j}}) \tag{5.20}$$

$e^{i,j}_{dyna}$ is a dynamic reprojection error:

$$e^{i,j}_{dyna} = \rho(||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w \exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i}, {}^{o}\mathbf{X})||_{\Sigma^{-1}_{i,j}}) \tag{5.21}$$

where $\Sigma^{-1}_{i,j}$ is estimated using the MAD as in equation (5.17). Note that the twist optimized here is not the same as the one defined earlier as it is applied on the current object pose and not on the previous pose. It should thus be seen as a twist that refines the current pose rather than a twist linking consecutive poses, which is why we denote it with a tilde. e^{i}_{const} is a constant velocity model that penalizes twists variations by linking 3 consecutive poses:

$$e^{i}_{const} = \rho(||\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_{i+1}} - \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}||_{\mathbf{W}}) \tag{5.22}$$

where $\mathbf{W}$ is a diagonal weight matrix used to balance the errors, ${}^{w}\boldsymbol{\xi}_{o_{i+1}}$ is the twist linking the poses $\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i}$ and $\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}$ and ${}^{w}\boldsymbol{\xi}_{o_i}$ is the twist linking the poses $\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}){}^{w}\mathbf{T}_{o_{i-1}}$ and $\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i}$. Those twists are computed using the logmap from SE(3) to se(3) defined in [Blanco, 2010] and can be written for ${}^{w}\boldsymbol{\xi}_{o_{i+1}}$ as:

$${}^{w}\boldsymbol{\xi}_{o_{i+1}} = \log(\exp((\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}})(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i})^{-1}) \tag{5.23}$$

and for ${}^{w}\boldsymbol{\xi}_{o_i}$ as:

$${}^{w}\boldsymbol{\xi}_{o_i} = \log(\exp((\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i})(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}){}^{w}\mathbf{T}_{o_{i-1}})^{-1}) \tag{5.24}$$

This equation moves each pose while respecting the mechanical joints constraints. Optimizing it can be cumbersome however the Schur trick can be applied as its Hessian is sparse [Bescos et al., 2021]. These equations are classically optimized on a set of local keyframes that share visual information, but in our case, inspired by [Huang et al., 2020] we chose to have 2 sets of keyframes: temporal and spatial. All frames are converted to temporal keyframes to improve the tracking of fast moving objects and be able to track an object as soon as it enters the field of view of the camera. Keyframes stay in the temporal set for a fixed duration (in our case 5 seconds) they are then culled more severely than

in ORB-SLAM2. We chose to optimize camera poses on the set of temporal and local keyframes, while object poses are only optimized on the set of temporal keyframes and fixed in all other keyframes. Doing so, we apply our constant motion model only on the temporal window, allowing clusters to accelerate or decelerate.

Note: It seems interesting here to underline an unsuccessful approach we experimented. We first designed our object reprojection cost function to be:

$$e^{i,j} = \rho(||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w \prod_{l=1}^{i} \exp(\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_l})^{w}\mathbf{T}_{o_0}, {}^{o}\mathbf{X})||_{\Sigma_{i,j}^{-1}}) \tag{5.25}$$

which makes sense as the i^{th} object pose can be obtained by applying the product of the previous twists in the exponential map to the first pose. However this has the effect of linking object twists and poses. Indeed the i^{th} object pose depends on all the previous twists, thus its derivative with respect to all previous twists is non null. This creates dense triangular blocks in the Jacobian and Hessian. The triangular shape is due to the fact that each twist depends only on the previous ones and not on the following ones. We give in figure 5.5 an example of Hessian containing object and camera poses for a single object. We can see it contains four blocks. The upper left diagonal block corresponds to the second order derivatives with respect to camera poses. The lower right dense block corresponds to the derivatives with respect to object poses and is the result of the product of two triangular blocks. The other blocks are the crossed derivatives with respect to cameras and object poses. The problem with those dense blocks is that they considerably increase the time necessary to solve the normal equations as the matrix is not block diagonal anymore. For example, the BA using this Hessian with 19 camera poses and 18 object poses takes tens of seconds to solve. It prevents us to apply this approach in real-time SLAM scenarios. On the other hand, in the approach that we proposed each object pose depends only on itself and on the camera pose, making the Hessian sparser.

5.4.7 Computing the cost functions Jacobians

To optimize the cost functions (5.18) and (5.19) with a Levenberg-Marquardt optimizer we need to compute their Jacobian. They can be either estimated using the finite difference

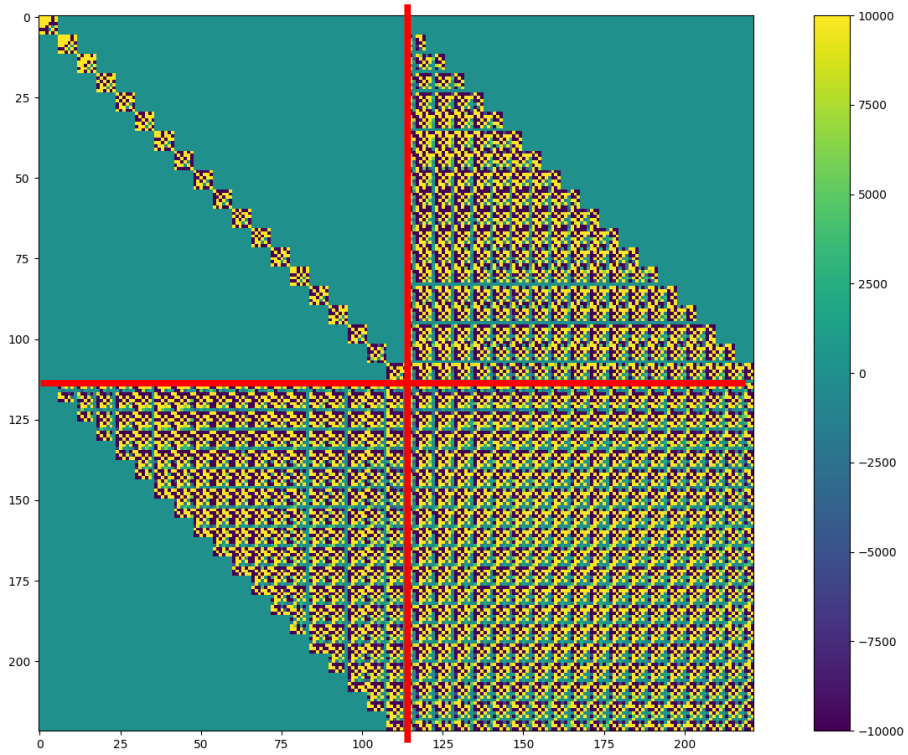

Figure 5.5 – Example of semi-dense Hessian with 19 camera poses and 18 object poses. The blocks are delimited by red lines. Non zero entries are yellow and dark blue.

method or computed analytically. To obtain the best convergence possible we chose the second option. First we compute the Jacobian of the cost function used for object tracking $E({}^{w}\boldsymbol{\xi}_{o_i})$. Using the chain rule and getting inspiration from [Blanco, 2010] it can be shown that:

$$\frac{\partial e^j({}^{w}\boldsymbol{\xi}_{o_i})}{\partial {}^{w}\boldsymbol{\xi}_{o_i}} = \frac{\partial \pi({}^{c}\mathbf{X}_j)}{\partial {}^{c}\mathbf{X}_j}({}^{c}\mathbf{X}_j^{\top} \otimes \mathbf{I}_3)(\mathbf{I}_4 \otimes {}^{c_i}\mathbf{R}_w)({}^{w}\mathbf{T}_{o_{i-1}}{}^{\top} \otimes \mathbf{I}_3)\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} \tag{5.26}$$

where $\otimes$ is the Kronecker product, $\mathbf{I}_N$ is an identity matrix of size N, ${}^{c_i}\mathbf{R}_w$ is the rotation matrix of ${}^{c_i}\mathbf{T}_w$ and $\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$ is:

$$\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \begin{pmatrix} \mathbf{0}_{3\times3} & -[\mathbf{e}_1]_\times \\ \mathbf{0}_{3\times3} & -[\mathbf{e}_2]_\times \\ \mathbf{0}_{3\times3} & -[\mathbf{e}_3]_\times \\ \mathbf{I}_3 & \mathbf{0}_{3\times3} \end{pmatrix} \tag{5.27}$$

where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ is the canonical base of $\mathbb{R}^3$.

Proof. The reprojection error can be seen as a composition of functions:

$$e^j({}^{w}\boldsymbol{\xi}_{o_i}) = {}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w \exp(\boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i})\,{}^{w}\mathbf{T}_{o_{i-1}}, {}^{o}\mathbf{X}_j) \tag{5.28}$$

$$= {}^{i}\mathbf{x}_j - \pi(f_1(f_2(f_3(\exp(\boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i}))))) \tag{5.29}$$

where $f_1(\mathbf{T}) = \mathbf{T}\,{}^{o}\bar{\mathbf{X}}_j$ corresponds to the application of a transformation to the 3D point ${}^{o}\mathbf{X}_j$, $f_2(\mathbf{T}) = {}^{c_i}\mathbf{T}_w\,\mathbf{T}$ is the left multiplication by the transformation ${}^{c_i}\mathbf{T}_w$ and $f_3(\mathbf{T}) = \mathbf{T}\,{}^{w}\mathbf{T}_{o_{i-1}}$ is the right multiplication by ${}^{w}\mathbf{T}_{o_{i-1}}$. We compute the derivative of this error with respect to the twist using the chain rule:

$$\left.\frac{\partial e^j({}^{w}\boldsymbol{\xi}_{o_i})}{\partial {}^{w}\boldsymbol{\xi}_{o_i}}\right|_{{}^{w}\boldsymbol{\xi}_{o_i}=0} = -\left.\frac{\partial \pi(\mathbf{X})}{\partial \mathbf{X}}\right|_{\mathbf{X}=\mathbf{Y}}\left.\frac{\partial f_1(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}\left.\frac{\partial f_2(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{B}}\left.\frac{\partial f_3(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{C}}\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i}}\boldsymbol{\Pi} \tag{5.30}$$

where $\mathbf{Y}$ is the non homogeneous form of ${}^{c_i}\mathbf{T}_w \exp(\boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i})\,{}^{w}\mathbf{T}_{o_{i-1}}\,{}^{o}\bar{\mathbf{X}}_j$,
$\mathbf{A} = {}^{c_i}\mathbf{T}_w \exp(\boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i})\,{}^{w}\mathbf{T}_{o_{i-1}}$, $\mathbf{B} = \exp(\boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i})\,{}^{w}\mathbf{T}_{o_{i-1}}$ and $\mathbf{C} = \exp(\boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i})$.

As detailed in [Blanco, 2010] the derivatives of left and right multiplication as well as pose-point composition for poses $\mathbf{T_A}$ and $\mathbf{T_B}$ are:

$$\frac{\partial \mathbf{T_A}\mathbf{T_B}}{\partial \mathbf{T_A}} = \mathbf{T_B}{}^{\top} \otimes \mathbf{I}_3 \tag{5.31}$$

$$\frac{\partial \mathbf{T_A T_B}}{\partial \mathbf{T_B}} = \mathbf{I}_4 \otimes \mathbf{R_A} \tag{5.32}$$

$$\frac{\partial \mathbf{T_A X}}{\partial \mathbf{T_A}} = \bar{\mathbf{X}}^\top \otimes \mathbf{I}_3 \tag{5.33}$$

Thus equation (5.30) can be re-written:

$$\left.\frac{\partial e^j({}^w\boldsymbol{\xi}_{o_i})}{\partial {}^w\boldsymbol{\xi}_{o_i}}\right|_{{}^w\boldsymbol{\xi}_{o_i}=0} = -\left.\frac{\partial \pi(\mathbf{X})}{\partial \mathbf{X}}\right|_{\mathbf{X}=\mathbf{Y}} (\bar{\mathbf{Y}}^\top \otimes \mathbf{I}_3)(\mathbf{I}_4 \otimes {}^{c_i}\mathbf{R}_w)({}^w\mathbf{T}_{o_{i-1}}{}^\top \otimes \mathbf{I}_3)\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} \tag{5.34}$$

thus yielding the Jacobian of the reprojection error. ■

Then we compute the Jacobian of e_{dyna} in equation (5.19), which is very similar to the previous Jacobian. The derivatives of the function with respect to camera poses and points are the same as for classical bundle adjustment [Dellaert, 2014]. For the object poses we can compute:

$$\begin{aligned}(J_{e_{dyna}})_{i,j} =& \frac{\partial e^{i,j}_{dyna}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_i}} \\ =& \frac{\partial \pi({}^{c_i}\mathbf{X}_j)}{\partial {}^{c_i}\mathbf{X}_j}({}^{c_i}\bar{\mathbf{X}}_j^\top \otimes \mathbf{I}_3)(\mathbf{I}_4 \otimes {}^{c_i}\mathbf{R}_w)({}^w\mathbf{T}_{o_i}{}^\top \otimes \mathbf{I}_3)\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi}\end{aligned} \tag{5.35}$$

Proof. The reprojection errors for tracking and for the BA are very similar, we can write:

$$e^{i,j} = {}^i\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w \exp(\boldsymbol{\Pi}\,{}^w\tilde{\boldsymbol{\xi}}_{o_i})\,{}^w\mathbf{T}_{o_i}, {}^o\mathbf{X}_j) \tag{5.36}$$

$$= {}^i\mathbf{x}_j - \pi(f_1(f_2(f_3(\exp(\boldsymbol{\Pi}\,{}^w\tilde{\boldsymbol{\xi}}_{o_i}))))) \tag{5.37}$$

where $f_1(\mathbf{T}) = \mathbf{T}\,{}^o\bar{\mathbf{X}}_j$ corresponds to the application of a transformation to the 3D point ${}^o\mathbf{X}_j$, $f_2(\mathbf{T}) = {}^{c_i}\mathbf{T}_w\,\mathbf{T}$ is the left multiplication by the transformation ${}^{c_i}\mathbf{T}_w$ and $f_3(\mathbf{T}) = \mathbf{T}\,{}^w\mathbf{T}_{o_i}$ is the right multiplication by ${}^w\mathbf{T}_{o_i}$.

Using the previously computed derivative it follows immediately that

$$\left.\frac{\partial e_{dyna}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^w\tilde{\boldsymbol{\xi}}_{o_i}=0} = \frac{\partial \pi({}^{c_i}\mathbf{X}_j)}{\partial {}^{c_i}\mathbf{X}_j}({}^{c_i}\bar{\mathbf{X}}_j^\top \otimes \mathbf{I}_3)(\mathbf{I}_4 \otimes {}^{c_i}\mathbf{R}_w)({}^w\mathbf{T}_{o_i}{}^\top \otimes \mathbf{I}_3)\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} \tag{5.38}$$

where ${}^{c_i}\bar{\mathbf{X}}_j = {}^{c_i}\mathbf{T}_w \exp(\boldsymbol{\Pi}\,{}^w\tilde{\boldsymbol{\xi}}_{o_i})\,{}^w\mathbf{T}_{o_i}\,{}^o\bar{\mathbf{X}}_j$ which yields the Jacobian for the reprojection error of the BA.

■

Finally we compute the Jacobian of the constant velocity constraint with respect to each of the 3 twists involved in the constraint:

$$(J_{e_{const}})_i = \begin{pmatrix} \frac{\partial e^i_{const}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}} & \frac{\partial e^i_{const}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_i}} & \frac{\partial e^i_{const}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}} \end{pmatrix} \tag{5.39}$$

The derivative with respect to ${}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}$ can be written:

$$\frac{\partial e^i_{const}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}} = \boldsymbol{\Pi}\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}(\mathbf{I}_4 \otimes \mathbf{R})\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} \tag{5.40}$$

with $\mathbf{T} = \exp((\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i})(\exp(\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}){}^w\mathbf{T}_{o_{i-1}})^{-1}$, $\mathbf{R}$ is the rotation matrix of $\exp((\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i})({}^w\mathbf{T}_{o_{i-1}})^{-1}$ and the derivative of the logmap is given by [Blanco, 2010]. The derivative with respect to ${}^w\tilde{\boldsymbol{\xi}}_{o_i}$ can be written:

$$\frac{\partial e_{const}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_i}} = -\frac{\partial \log(\mathbf{T_A})}{\partial \mathbf{T_A}}(\mathbf{T_B}^\top \otimes \mathbf{I}_3)\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} - \boldsymbol{\Pi}\frac{\partial \log(\mathbf{T_C})}{\partial \mathbf{T_C}}(\mathbf{I}_4 \otimes \mathbf{R})\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} \tag{5.41}$$

with $\mathbf{T_A} = \exp((\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i})(\exp(\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}){}^w\mathbf{T}_{o_{i-1}})^{-1}$,
$\mathbf{T_C} = \exp((\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^w\mathbf{T}_{o_{i+1}})(\exp(\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i})^{-1}$, $\mathbf{T_B} = {}^w\mathbf{T}_{o_i}(\exp((\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}){}^w\mathbf{T}_{o_{i-1}})^{-1}$ and $\mathbf{R}$ is the rotation matrix of $\exp((\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^w\mathbf{T}_{o_{i+1}})^{o_i}\mathbf{T}_w$

The derivative with respect to ${}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}$ can be written:

$$\frac{\partial e_{const}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}} = \boldsymbol{\Pi}\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}(\mathbf{T_B}^\top \otimes \mathbf{I}_3)\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} \tag{5.42}$$

with $\mathbf{T} = \exp((\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^w\mathbf{T}_{o_{i+1}})(\exp(\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i})^{-1}$, $\mathbf{T_B} = {}^w\mathbf{T}_{o_{i+1}}(\exp((\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i})^{-1}$

Proof. First we show how to compute the derivative with respect to ${}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}$. Only the right part of e_{const} that depends on ${}^w\boldsymbol{\xi}_{o_i}$ also depends on ${}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}$. The derivative of the left part is thus 0. We write the right part as:

$$\begin{aligned}
\boldsymbol{\Pi}{}^w\boldsymbol{\xi}_{o_i} &= \boldsymbol{\Pi}\log(\exp(\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i}(\exp(\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}){}^w\mathbf{T}_{o_{i-1}})^{-1}) && (5.43)\\
&= \boldsymbol{\Pi}\log(\exp(\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i}({}^{o_{i-1}}\mathbf{T}_w\exp(-\boldsymbol{\Pi}{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}))) && (5.44)\\
&= \boldsymbol{\Pi}\log(f_1(f_2({}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}))) && (5.45)
\end{aligned}$$

where $f_1(\mathbf{T}) = \exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i})^w\mathbf{T}_{o_i}{}^{o_{i-1}}\mathbf{T}_w\mathbf{T}$ is the left multiplication by the pose $\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i})^w\mathbf{T}_{o_i}{}^{o_{i-1}}\mathbf{T}_w$ and $f_2({}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}) = \exp(-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}})$. Using the chain rule it follows that:

$$\left.\frac{\partial \mathbf{\Pi}^w\boldsymbol{\xi}_{o_i}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}}\right|_{{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0} = \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}\left.\frac{\partial f_1(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{B}}\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0}(-\mathbf{\Pi}) \tag{5.46}$$

where $\mathbf{A} = \exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i})^w\mathbf{T}_{o_i}(\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}})^w\mathbf{T}_{o_{i-1}})^{-1}$ and $\mathbf{B} = \exp(-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}})$. Using the derivative of right multiplication transformation, we can write:

$$\left.\frac{\partial \mathbf{\Pi}^w\boldsymbol{\xi}_{o_i}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}}\right|_{{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0} = \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}(\mathbf{I}_4\otimes\mathbf{R})\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0}(-\mathbf{\Pi}) \tag{5.47}$$

where $\mathbf{R}$ is the rotation part of $\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i})^w\mathbf{T}_{o_i}{}^{o_{i-1}}\mathbf{T}_w$. We thus have:

$$\left.\frac{\partial e_{const}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}}\right|_{{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0} = \frac{\partial \mathbf{\Pi}^w\boldsymbol{\xi}_{o_{i+1}}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}} - \left.\frac{\partial \mathbf{\Pi}^w\boldsymbol{\xi}_{o_i}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}}\right|_{{}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0} \tag{5.48}$$

$$= 0 - \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}(\mathbf{I}_4\otimes\mathbf{R})\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0}(-\mathbf{\Pi}) \tag{5.49}$$

$$= \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}(\mathbf{I}_4\otimes\mathbf{R})\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0}\mathbf{\Pi} \tag{5.50}$$

The details for the computation of the derivative of the log map and exponential map can be found in [Blanco, 2010] p. 54 and p. 58.

We then show how to compute the derivative with respect to ${}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}$. Only the left part of e_{const} that depends on ${}^w\boldsymbol{\xi}_{o_{i+1}}$ also depends on ${}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}$. The derivative of the right part is thus 0. We write the left part as:

$$\mathbf{\Pi}^w\boldsymbol{\xi}_{o_{i+1}} = \mathbf{\Pi}\log(\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}})^w\mathbf{T}_{o_{i+1}}(\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i})^w\mathbf{T}_{o_i})^{-1}) \tag{5.51}$$

$$= \mathbf{\Pi}\log(\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}})^w\mathbf{T}_{o_{i+1}}({}^{o_i}\mathbf{T}_w\exp(-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i}))) \tag{5.52}$$

$$= \mathbf{\Pi}\log(f_1(f_2({}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}))) \tag{5.53}$$

where $f_1(\mathbf{T}) = \mathbf{T}^w\mathbf{T}_{o_{i+1}}({}^{o_i}\mathbf{T}_w\exp(-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i}))$ is the right multiplication by ${}^w\mathbf{T}_{o_{i+1}}({}^{o_i}\mathbf{T}_w\exp(-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i}))$ and $f_2({}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}) = \exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}})$. Again, using the chain rule we

can write:

$$\left.\frac{\partial \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_{i+1}}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}=0} = \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}} (\mathbf{T_B}^{\top} \otimes \mathbf{I}_3)\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}=0} \mathbf{\Pi} \tag{5.54}$$

where $\mathbf{A} = \exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i})^{-1}$ and $\mathbf{T_B} = {}^{w}\mathbf{T}_{o_{i+1}}({}^{o_i}\mathbf{T}_{w}\exp(-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}))$. It follows that:

$$\left.\frac{\partial e_{const}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}=0} = \frac{\partial \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_{i+1}}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}} - \left.\frac{\partial \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{\boldsymbol{\xi}=0} \tag{5.55}$$

$$= \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}} (\mathbf{T_B}^{\top} \otimes \mathbf{I}_3)\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}=0} \mathbf{\Pi} + 0 \tag{5.56}$$

Finally we compute the derivative with respect to ${}^{w}\tilde{\boldsymbol{\xi}}_{o_i}$. Contrary to the other derivatives both right and left parts of e_{const} depend on ${}^{w}\tilde{\boldsymbol{\xi}}_{o_i}$. We first write the left part:

$$\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_{i+1}} = \mathbf{\Pi}\log(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i})^{-1}) \tag{5.57}$$

$$= \mathbf{\Pi}\log(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}({}^{o_i}\mathbf{T}_{w}\exp(-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}))) \tag{5.58}$$

$$= \mathbf{\Pi}\log(f_1(f_2({}^{w}\tilde{\boldsymbol{\xi}}_{o_i}))) \tag{5.59}$$

where $f_1(\mathbf{T}) = \exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}{}^{o_i}\mathbf{T}_{w}\mathbf{T}$ is the left multiplication by $\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}{}^{o_i}\mathbf{T}_{w}$ and $f_2({}^{w}\tilde{\boldsymbol{\xi}}_{o_i}) = \exp(-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i})$. As we can see this equation is very similar to equation (5.45), the only difference being the index i. Thus we deduce the derivative:

$$\left.\frac{\partial \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_{i+1}}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} = \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{C}} (\mathbf{I}_4 \otimes \mathbf{R})\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} (-\mathbf{\Pi}) \tag{5.60}$$

where $\mathbf{C} = \exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i})^{-1}$ and $\mathbf{R}$ is the rotation part of $\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^{w}\mathbf{T}_{o_{i+1}}{}^{o_i}\mathbf{T}_{w}$. The right part of the derivative can be written:

$$\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i} = \mathbf{\Pi}\log(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i}(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}){}^{w}\mathbf{T}_{o_{i-1}})^{-1}) \tag{5.61}$$

$$= \mathbf{\Pi}\log(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}){}^{w}\mathbf{T}_{o_i}({}^{o_{i-1}}\mathbf{T}_{w}\exp(-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}))) \tag{5.62}$$

$$= \mathbf{\Pi}\log(f_1(f_2({}^{w}\tilde{\boldsymbol{\xi}}_{o_i}))) \tag{5.63}$$

where $f_1(\mathbf{T}) = \mathbf{T}{}^{w}\mathbf{T}_{o_i}({}^{o_{i-1}}\mathbf{T}_{w}\exp(-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}))$ is the right multiplication by

${}^{w}\mathbf{T}_{o_i}({}^{o_{i-1}}\mathbf{T}_{w}\exp(-\mathbf{\Pi}{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}))$ and $f_2({}^{w}\tilde{\boldsymbol{\xi}}_{o_i}) = \exp(\mathbf{\Pi}{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})$. This equation is similar to equation (5.53), the difference being again on the index i. We can thus write:

$$\left.\frac{\partial \mathbf{\Pi}{}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} = \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}(\mathbf{T_B}^{\top}\otimes \mathbf{I}_3)\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{\Pi}{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0}\mathbf{\Pi} \tag{5.64}$$

where $\mathbf{A} = \exp(\mathbf{\Pi}{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})^{w}\mathbf{T}_{o_i}(\exp(\mathbf{\Pi}{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})^{w}\mathbf{T}_{o_{i-1}})^{-1}$ and
$\mathbf{T_B} = {}^{w}\mathbf{T}_{o_i}({}^{o_{i-1}}\mathbf{T}_{w}\exp(-\mathbf{\Pi}{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}))$. Finally:

$$\begin{aligned}\left.\frac{\partial e_{const}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} &= \frac{\partial \mathbf{\Pi}{}^{w}\boldsymbol{\xi}_{o_{i+1}}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}} - \left.\frac{\partial \mathbf{\Pi}{}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} \qquad (5.65)\\ &= \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{C}}(\mathbf{I}_4\otimes\mathbf{R})\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\mathbf{\Pi}{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0}(-\mathbf{\Pi})\\ &\quad - \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}(\mathbf{T_B}^{\top}\otimes\mathbf{I}_3)\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{\Pi}{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0}\mathbf{\Pi} \qquad (5.66)\end{aligned}$$

which concludes the computation of the Jacobian. ■

All the Jacobians computed here have been verified numerically using finite differences.

5.5 Experiments

In this section we present the experiments we conducted to test our approach. Our goal is to evaluate both the accuracy of the camera pose estimation and the accuracy of the object pose estimation.

5.5.1 Experiments details

Datasets. We evaluate our approach on the KITTI [Geiger et al., 2012] tracking dataset that we presented in chapter 2. This dataset is particularly interesting for our approach as it contains the ground truth for both the camera pose and for some objects poses such as vehicles. Furthermore the sequences present an important variability, including simple short sequences with only a few objects to long crowded scenes with vehicles and pedestrians.

Metrics. The metrics for the evaluation of SLAM systems are usually the absolute translation error (ATE) [J. Sturm et al., 2012] and the relative pose error (RPE) [Z. Zhang

and Scaramuzza, 2018] that we presented in chapter 2. For each sequence we report the translation and rotation parts of the RPE, as it is done by both VDO-SLAM and DynaSLAM2. The object pose estimation accuracy can be evaluated using 2 different types of metrics: on the one hand the ATE and RPE that measure the quality of the objects trajectories and on the other hand the MOTP that evaluates the per-frame accuracy of objects 3D bounding boxes estimations and that we compute similarly to [Bescos et al., 2021] using kitti evaluation tools. As we do not estimate object boxes we use the ground truth box at the first pose of each object and propagate it using our camera and object pose estimations. The confidence needed to compute the MOTP is given by the normalized number of keypoints used for tracking the object at each timestamp. We evaluate the true positive rate (TP) and the MOTP using the projected 3D bounding box (2D), in bird view (BV) and in 3D.

Computation times. As all frames are transformed into keyframes that need to be segmented we make our approach run at 1 to 2 fps on an Nvidia RTX2070. The BA converges rather rapidly even with the additional temporal keyframes and object poses, with a convergence time of 0.3 to 0.5 seconds.

5.5.2 Camera pose estimation

In this subsection we evaluate the accuracy of our camera pose estimation. Similarly to [Bescos et al., 2021] we only show here sequences in which the camera is moving. As we can see in table 5.1 our approach improves camera pose estimation on several sequences. As objects are often either small, only visible for a short time or static, [Mur-Artal and Tardós, 2017] performs well, but as we track clusters using many points, with a good precision, especially for clusters that do not move, we are able to reduce the drift. [J. Zhang et al., 2020] also gives good results but requires depth information while our approach gives similar or better results than RGB based approaches. The most important improvement of our approach is in terms of object tracking accuracy as we can see in table 5.2.

5.5.3 Object pose estimation.

In this subsection we evaluate the accuracy of our object pose estimation. We compare our results to [Bescos et al., 2021] that is the only approach that reports all metrics for separated objects, allowing a better evaluation of the object pose estimation accuracy.

Table 5.1 – Camera pose estimation on the Kitti tracking dataset. Comparison with ORB-SLAM2 [Mur-Artal and Tardós, 2017], DynaSLAM [Bescos et al., 2018], VDO-SLAM [J. Zhang et al., 2020] and DynaSLAM2 [Bescos et al., 2021].

seq	ORB-SLAM2		DynaSLAM		VDO-SLAM (RGB-D)		DynaSLAM2		Ours	
	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)
00	**0.04**	0.06	**0.04**	0.06	0.07	0.07	**0.04**	0.06	**0.04**	**0.05**
01	0.05	0.04	0.05	0.04	**0.04**	0.12	0.05	0.04	**0.04**	**0.03**
02	0.04	0.03	0.04	0.03	**0.02**	0.04	0.04	**0.02**	0.03	0.03
03	0.07	0.04	0.07	0.04	**0.03**	0.08	0.06	0.04	0.06	**0.02**
04	0.07	0.06	0.07	0.06	**0.05**	0.11	0.07	0.06	0.06	**0.04**
05	0.06	0.03	0.06	0.03	**0.02**	0.09	0.06	0.03	0.06	**0.02**
06	**0.02**	0.04	**0.02**	0.04	0.05	0.02	**0.02**	**0.01**	**0.02**	0.04
07	0.05	0.07	0.05	0.07	-	-	0.05	0.07	**0.04**	**0.04**
08	0.08	0.04	0.08	0.04	-	-	0.10	0.04	**0.07**	**0.03**
09	0.06	0.05	0.06	0.05	-	-	0.06	0.06	**0.05**	**0.04**
10	**0.07**	0.04	**0.07**	0.04	-	-	**0.07**	**0.03**	**0.07**	**0.03**
11	0.04	0.03	0.04	0.03	-	-	0.04	0.03	**0.03**	**0.02**
13	0.04	0.05	0.04	0.05	-	-	0.04	**0.04**	**0.03**	**0.04**
14	**0.03**	0.08	**0.03**	0.08	-	-	**0.03**	0.08	**0.03**	**0.06**
18	0.05	0.03	0.05	0.03	**0.02**	0.07	0.05	**0.02**	0.04	**0.02**
19	0.05	0.03	0.05	0.03	-	-	0.05	0.02	**0.03**	**0.03**
20	0.11	0.07	0.05	0.04	**0.03**	0.17	0.07	0.04	0.04	**0.03**
mean	0.055	0.046	0.051	0.045	-	-	0.053	0.043	**0.044**	**0.034**

The reported sequences where chosen by [Bescos et al., 2021] to maximize the tracking duration and the objects size in images. As we can see we improve object tracking accuracy, particularly for static objects such as the car 35 from sequence 11. The most important improvements usually come from the rotational part of the RPE, which is understandable as we only have 1 degree of freedom for the rotation of cars. While object tracking accuracy is very good it is not as good as ego motion estimation. As [Bescos et al., 2021] we argue that this is because we extract less keypoints from objects than from the rest of the scene. Furthermore the amount of high quality keypoints is more limited in objects than in the static scene, meaning that we can not extract more points without getting points that are not well localized in 2D or that are prone to be wrongly matched. We also observe that the most challenging cases happen when an object starts and stays far from the camera (e.g. seq. 05 and 10) because object tracking uses 3D points triangulated from stereo matches that are imprecise when points are far from the camera. We argue that the bruteforce keypoint matching of [Bescos et al., 2021] help them when few frame to frame matches can be found, which can happen when the object is far from the camera. Furthermore we have not implemented a way to relocalize an object that has been lost for multiple frames. Thus on some sequences (e.g. car 0 of seq. 11 and car 12 of seq. 20) in which the objects are alternatively far and close from the camera, we are only able to track them on a small portion of their trajectory. However, we can see that we are generally able to accurately track objects for most of their trajectory. During our experiments we saw that we were able to track pedestrians, despite the fact that they are not rigid. We believe that our approach works because pedestrians only undergo small deformations around arms and legs. However as they are usually small in the image we can only track them when they are close enough to the camera.

We also show some qualitative results for the mapping, the camera and object pose estimation. The results are visible in figures 5.6, 5.7, 5.8. We are able to track multiple objects on all sequences. The estimated speed is very close from the ground truth with a maximum difference of about 3 km/h which occurs when the object is far from the camera or when it is created. Looking at the bounding boxes we can see that they coincide and thus that the poses are well estimated for near and far objects. As we can see in the middle figure we can accurately track non rigid objects such as the cyclist, as long as most of their surface is rigid. We also show on figure 5.8 an example of both a tracked static car and a car that slows down and speeds up. As we can see the estimated speed is close to the 0 for the static car, making the dynamic keypoints act like static ones. We can also

Table 5.2 – Object pose estimation comparison on the Kitti tracking dataset. ATE is in m, RPE_t in m/m, RPE_R in °/m, TP and MOTP in %.

	DynaSLAM 2 [Bescos et al., 2021]									TwistSLAM								
seq / obj. id / class	ATE	RPE_t	RPE_R	2D TP	2D MOTP	BV TP	BV MOTP	3D TP	3D MOTP	ATE	RPE_t	RPE_R	2D TP	2D MOTP	BV TP	BV MOTP	3D TP	3D MOTP
03 / 1 / car	0.69	0.34	1.84	50.0	**71.79**	39.34	56.61	38.53	48.20	**0.31**	**0.10**	**0.28**	**58.02**	60.0	**58.02**	**60.0**	**58.02**	**60.0**
05 / 31 / car	0.51	0.26	13.5	28.96	**60.30**	14.48	**46.84**	11.45	34.20	**0.35**	**0.19**	**0.58**	**30.84**	35.0	**30.84**	35.0	**30.84**	**35.0**
10 / 0 / car	0.95	0.40	2.84	**81.63**	**73.51**	**70.41**	**47.60**	**68.37**	**40.28**	**0.77**	**0.21**	**1.98**	7.2	3.7	6.1	3.1	5.8	2.8
11 / 0 / car	1.05	0.43	12.51	**72.65**	**74.78**	**61.66**	**50.74**	**52.28**	**47.35**	**0.17**	**0.23**	**0.23**	29.61	32.5	29.61	32.5	29.61	32.5
11 / 35 car	1.25	0.89	16.64	53.17	65.25	19.05	31.95	6.35	26.02	**0.10**	**0.03**	**0.11**	**65.0**	**67.5**	**65.0**	**67.5**	**65.0**	**67.5**
18 / 2 / car	1.10	0.30	9.27	**86.36**	74.81	67.05	45.47	62.12	34.80	**0.21**	**0.27**	**0.66**	84.67	**87.5**	**84.67**	**87.5**	**84.67**	**87.5**
18 / 3 / car	1.13	0.55	20.05	**53.33**	**70.94**	21.75	**41.45**	16.84	**35.80**	**0.15**	**0.21**	**0.56**	28.19	30.00	**28.19**	30.0	**28.19**	30.0
19 / 63 / car	0.86	**1.45**	48.80	35.26	63.50	29.48	45.69	26.48	**33.89**	**0.28**	2.17	**1.08**	65.93	70.00	65.93	70.00	36.26	20.64
19 / 72 / car	0.99	1.12	3.36	**29.11**	**62.59**	**29.43**	**55.48**	**29.43**	39.81	**0.16**	**0.05**	**0.34**	16.92	20.00	16.92	20.00	16.92	20.00
20 / 0 / car	0.56	0.45	1.30	63.68	78.54	43.78	45.00	31.84	46.15	**0.17**	**0.20**	**0.72**	**84.75**	**87.5**	**84.75**	**87.5**	**84.75**	**87.5**
20 / 12 / car	1.18	0.40	6.19	**42.77**	**76.77**	**37.64**	**49.29**	**36.23**	40.81	**0.24**	**0.20**	**1.54**	14.24	17.5	13.91	17.45	13.04	17.25
20 / 122 / car	0.87	0.72	5.75	34.90	78.76	34.51	48.05	29.02	44.43	**0.17**	**0.02**	**0.07**	**84.94**	**87.5**	**84.94**	**87.5**	**84.94**	**87.5**

see the noise of the ground truth with the speed of the static car that goes up to 2 km/h.

5.6 Conclusion

In this chapter we proposed a new stereo semantic dynamic SLAM system called TwistSLAM, able to estimate both the pose of the camera as well as to track all dynamic objects in the scene. Using mechanical joints between clusters we can constrain objects movements to physically possible movements, which allows us to improve both camera and objects pose estimation compared to the state of the art. However our work shows some limits: the accuracy of object tracking is often lower than camera tracking. Its robustness is also too weak for real life applications yet as we are barely able to track cars further than 25 meters away from the camera. Furthermore, contrary to our first algorithm, L6DNet we do not have access to a dense 3D model of the object, but rather to a sparse pointcloud that gets polluted when object tracking drifts. Finally, as we do not use an object pose estimation algorithm we are unable to estimate canonical object poses. We propose to solve those problems in the last chapter of this manuscript.

This chapter was published in:

Mathieu Gonzalez, Eric Marchand, Amine Kacete and Jérôme Royan, "TwistSLAM: Constrained SLAM in Dynamic Environment", *in: IEEE Robotics and Automation Letters (RA-L)*, 2022. It was also published *in: IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2022

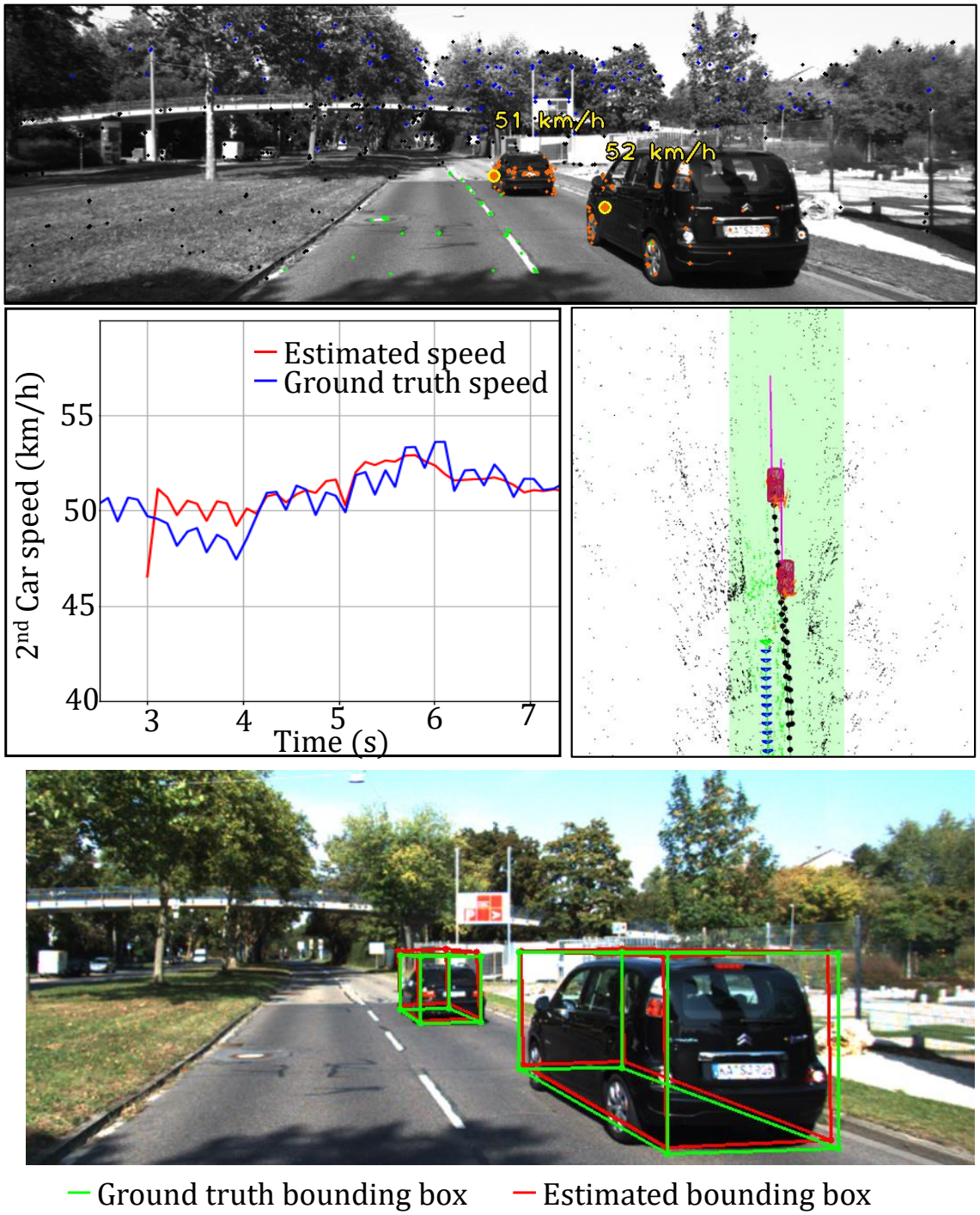

Figure 5.6 – Qualitative example of cluster tracking on sequence 3 from the KITTI tracking dataset. (Top) Frame with tracked clusters and their speed. (Middle left) Comparison of estimated (red) and ground truth (blue) speed (in km/h). (Middle right) Map with tracked clusters and camera poses, seen from above. (Bottom) Visualization of estimated poses (red) and ground truth (green) represented by their 3D bounding boxes.

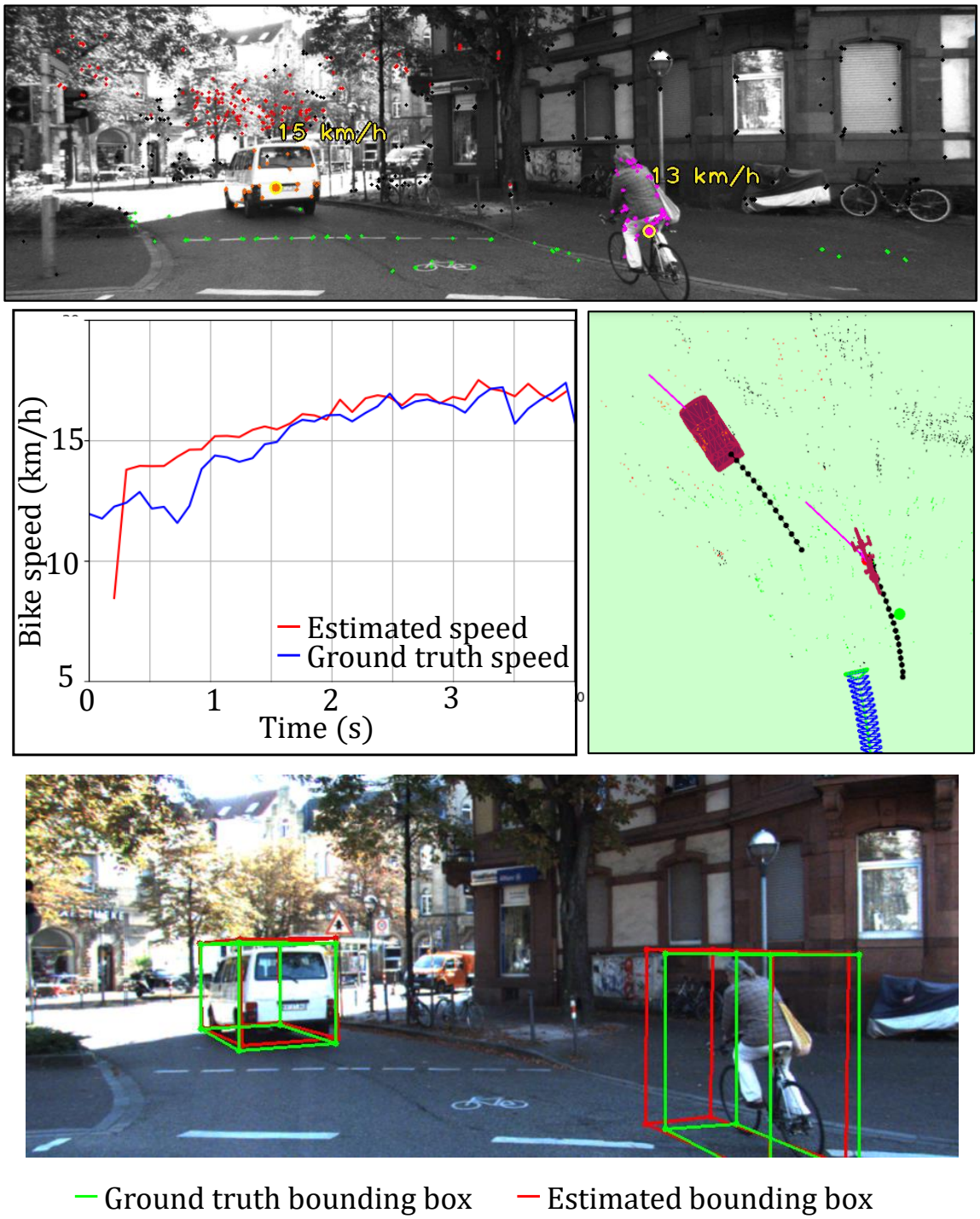

Figure 5.7 – Qualitative example of cluster tracking on sequence 0 from the KITTI tracking dataset. (Top) Frame with tracked clusters and their speed. (Middle left) Comparison of estimated (red) and ground truth (blue) speed (in km/h). (Middle right) Map with tracked clusters and camera poses, seen from above. (Bottom) Visualization of estimated poses (red) and ground truth (green) represented by their 3D bounding boxes.

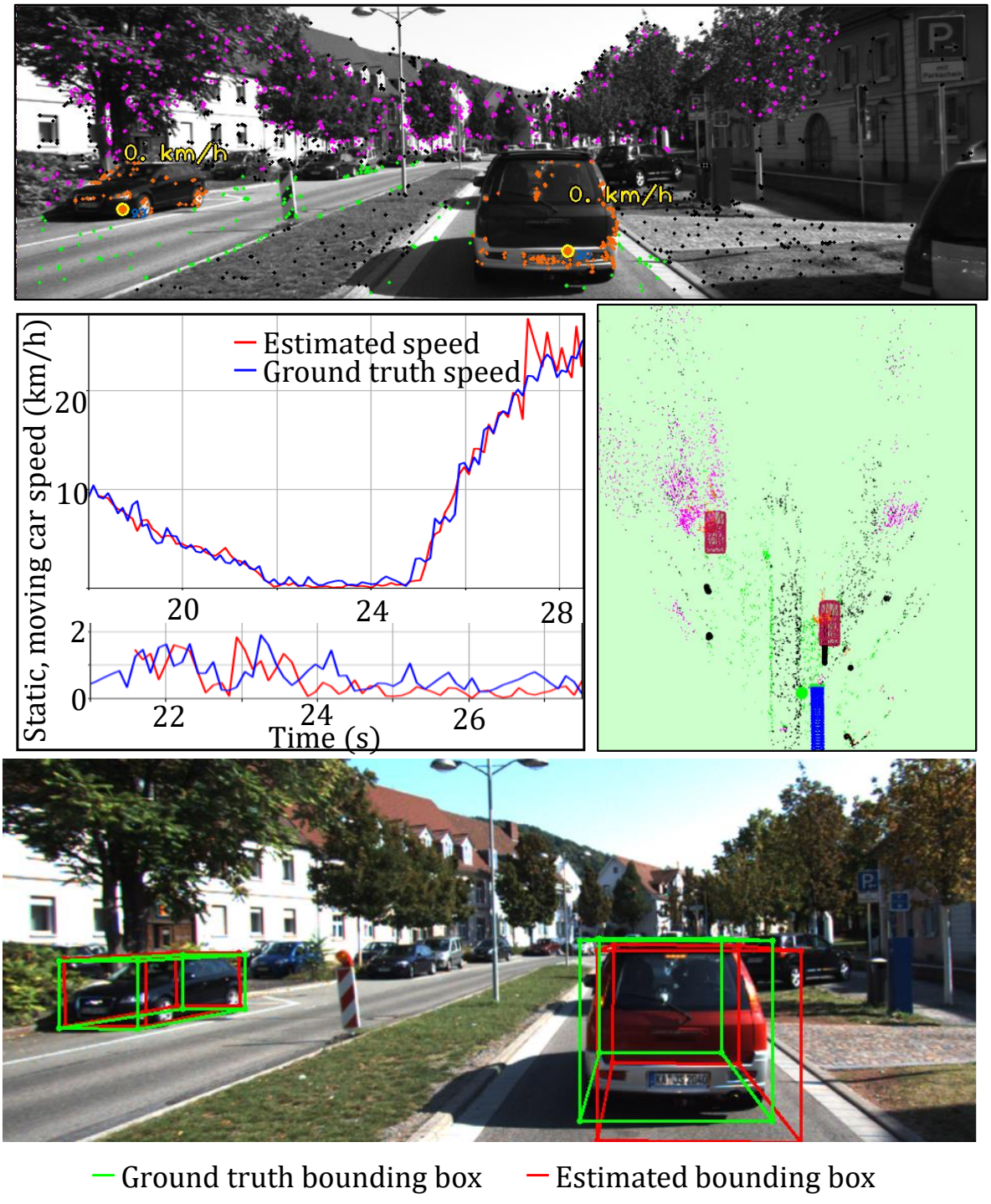

Figure 5.8 – Qualitative example of cluster tracking on sequence 11 from the KITTI tracking dataset. (Top) Frame with tracked clusters and their speed. (Middle left) Comparison of estimated (red) and ground truth (blue) speed (in km/h). (Middle right) Map with tracked clusters and camera poses, seen from above. (Bottom) Visualization of estimated poses (red) and ground truth (green) represented by their 3D bounding boxes.

Chapter 6

TwistSLAM++: Fusing multiple modalities for accurate dynamic semantic SLAM

Contents

In this chapter we propose to push the limits present in TwistSLAM, with a new approach that we call TwistSLAM++ [Gonzalez et al., 2022c]. More precisely we seek to improve the accuracy of object tracking, generalize the approach of S^{3}LAM on non planar objects and estimate the canonical pose of objects, as in L6DNet. To do so we propose to use another source of information: LiDAR scans, that we process in different ways and inject into TwistSLAM. First we use the 3D object detector 3DSSD [Z. Yang et al., 2020] to estimate the oriented 3D bounding boxes of cars in LiDAR scans. We associate the boxes to tracked clusters, allowing us to have access to the canonical pose of objects. We use estimated bounding boxes to associate LiDAR points to clusters and feed them to an ICP algorithm to obtain a new estimate of object poses. This estimate is then used to further constrain the BA, together with the pose estimation. Finally we fit a signed distance function (SDF) on object scans. In the BA we constrain object map points to lie on the surface of the SDF, improving the map. We show that those additional constraints improve object tracking accuracy. A video explaining the approach and showing examples of dynamic

object tracking and reconstruction can be found at https://youtu.be/xipseb0LMeU.

6.1 Introduction

In the previous chapter we have shown that we could accurately estimate camera pose in dynamic scenarios while estimating the trajectory of moving objects. However existing approaches are based solely on RGB information and are less precise than camera pose estimation due to the lower number of extracted points. This limit can even lead to failures in some cases (e.g. when the dynamic object is far from the camera or too small). Furthermore, those approaches suffer from tracking drift that can not be corrected by loop closure. Finally they do not have access to the canonical pose of the object but rather to its relative pose with respect to its initial pose. To solve those problems we propose to update our previous work TwistSLAM [Gonzalez et al., 2022b] by integrating a 3D object detector based on LiDAR information. This detector is used to predict the pose and size of 3D bounding boxes corresponding to potentially moving objects in the scene. We associate detections to tracked clusters to have access to their canonical pose, i.e. their pose with respect to an a priori known object coordinate frame. Furthermore, we use consecutive poses to constrain the displacement of objects, thus reducing the drift. The obtained bounding boxes are then used to associate 3D LiDAR points to tracked clusters which serves two purposes. First they allow us to improve object tracking by feeding successive scans to a generalized ICP algorithm. The computed pose is then used as a constraint in the bundle adjustment. Second, inspired by the work of DSP-SLAM [J. Wang et al., 2021], we use scans to fit a deep-learnt signed distance function (SDF) [J. J. Park et al., 2019] that represents the object geometry. However contrary to DSP-SLAM we do not use the SDF to estimate the object pose as we already have a good estimate of it, but we rather use it to constrain the 3D map points of clusters to lie on the estimated mesh, similarly to our previous work [Gonzalez, Marchand, et al., 2021] which was restricted to planes. We show in figure 6.1 an example of object pose estimation, tracking and reconstruction on cars from the Kitti tracking dataset.
To summarize, our contributions in this chapter are:

- A semantic SLAM system that can robustly estimate the pose of a camera in dynamic scenes.
- A SLAM framework that can track multiple moving objects and estimate their canonical pose.

- A SLAM system able to fuse 3D object pose estimation, object tracking and 3D registration results from LiDAR scans to reduce tracking drift.
- A SLAM framework that uses the 3D reconstruction of object from LiDAR data to constrain the geometry of map points.

We evaluate our approach on sequences from the KITTI tracking datasets. We compare our results with state of the art dynamic SLAM systems DynaSLAM 2 [Bescos et al., 2021] and TwistSLAM [Gonzalez et al., 2022b].

In this chapter we present and recall some related approaches, particularly the ones that use LiDAR scans for odometry and mapping. Following, we rapidly recall the work we presented in previous chapter and show how we inject LiDAR scans into our pipeline using a 3D object detector, a generalized ICP algorithm and a signed distance function. Finally we evaluate our approach on sequences from the Kitti dataset and perform an ablation study ton the different constraints of our SLAM.

6.2 Related work

In this section we recall some semantic dynamic SLAM systems presented in the previous chapter that tackle the problem of dynamic objects by tracking them [Bescos et al., 2021; Huang et al., 2020] or use them as high level landmarks [Salas-Moreno et al., 2013; S. Yang and Scherer, 2019a]. We also present SLAM systems that make use of LiDAR information and fuse multiple modalities to improve the accuracy and robustness of camera tracking.

DynaSLAM II [Bescos et al., 2021] uses semantic information to detect objects. Object 3D points are represented in the object reference frame and used to estimate the object pose at all time by minimizing their reprojection error.

Some approaches propose to detect objects in the scene to use them as high level landmarks. While some of them [Civera et al., 2011; Salas-Moreno et al., 2013] require a specific object pose estimation algorithm [Rad and Lepetit, 2017] which limits their applicability in real world scenarios, others are based on generic object detectors. Those approaches use quadrics [Gaudillière et al., 2019; Nicholson et al., 2018] or 3D bounding boxes [S. Yang and Scherer, 2019a] to represent objects. More recently, some approaches have represented the geometry of objects more accurately using learning based approaches. NodeSLAM [Sucar et al., 2020] optimizes detected object poses and shape, represented by an autoencoder. Object poses are then used in the SLAM system to estimate the camera

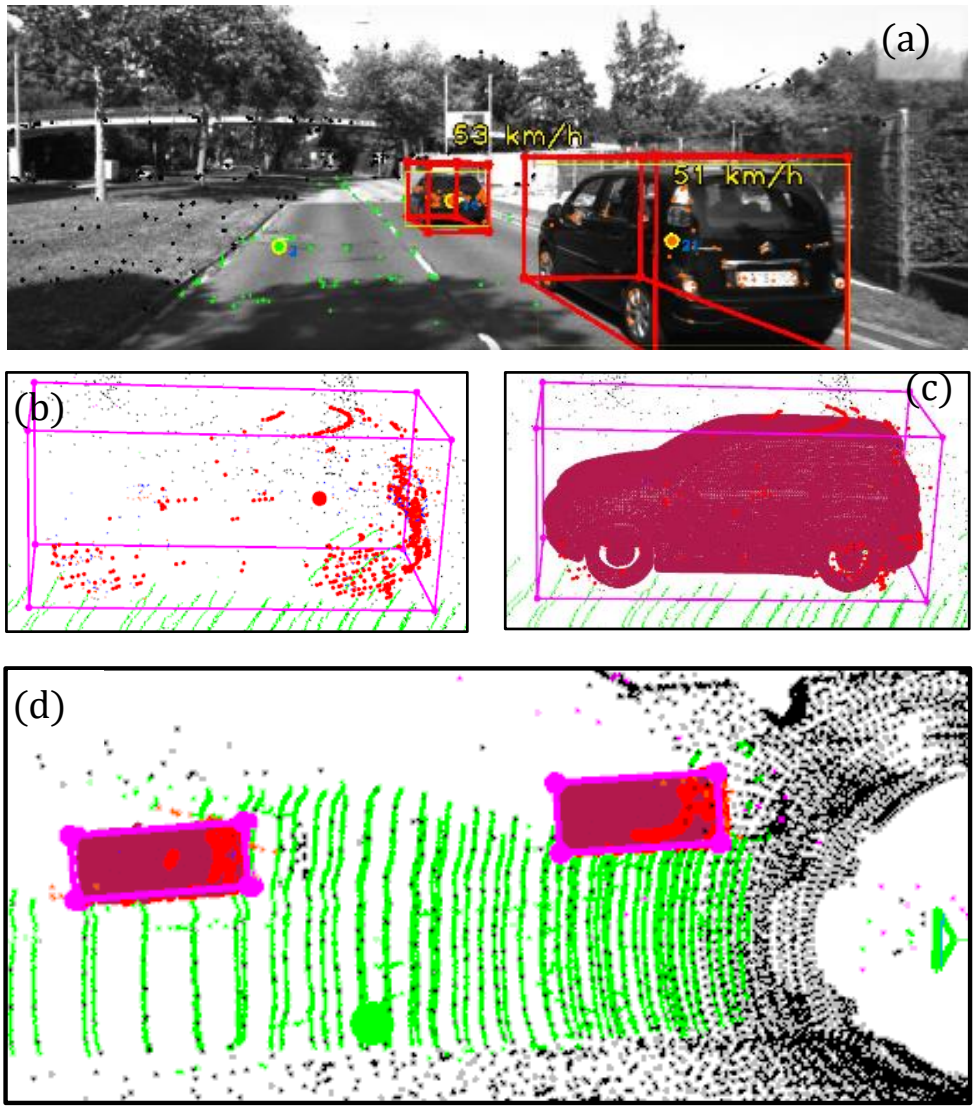

Figure 6.1 – In our SLAM system we track object (here cars) moving in the scene. We show here (a) a frame with tracked cars, their speed and reprojected bounding boxes. (b) the bounding box and clustered LiDAR points (red) for the closest car. (c) the reconstruction of the car, using the approach of DSP-SLAM [J. Wang et al., 2021]. (d) the map seen from above, with LiDAR points (black), road LiDAR points (green) and tracked cars.

pose. DSP-SLAM [J. Wang et al., 2021] optimizes the latent code of a deep learning based SDF [J. J. Park et al., 2019] and uses it to estimate object poses and to reconstruct the object shape. Object poses are then used to constrain camera pose estimation in the BA.

Finally, some approaches [Behley and Stachniss, 2018; C. Park et al., 2018] have been using LiDAR scans instead of images as an input for SLAM: [Behley and Stachniss, 2018] is a full SLAM system based only on LiDAR data, which represents the map using a set of surfels. 3D LiDAR points are transformed to the image plane using a spherical projection, yielding a so-called *vertex map*. This map is used, together with the *normal map* to estimate the updated current pose using point to plane registration after finding associations in the image plane. The current scan is then fused with the map to update it. Finally loop closure is performed. Virtual views are generated with the surfel map to compute the alignment with the current scan. After a verification step, pose graph optimization is performed and used to update the surfel map. Some approaches [X. Chen et al., 2019; 2021; Jeong et al., 2018; L. Sun et al., 2018] have also injected semantic information into LiDAR based SLAM systems, to improve pose estimation, for example by masking our moving objects. SuMa++ [X. Chen et al., 2019] improves on [Behley and Stachniss, 2018] by integrating a CNN to segment LiDAR scans [Milioto et al., 2019]. This allows them to obtain a higher level map. Furthermore semantic information is used to detect and remove surfels belonging to dynamic objects. It also used to guide the ICP by weighting associated points.

6.3 TwistSLAM++

Following the idea of the algorithms TwistSLAM [Gonzalez et al., 2022b] and S^{3}LAM [Gonzalez, Marchand, et al., 2021] we use a panoptic neural network [Y. Wu et al., 2019] to create a map of clusters corresponding to objects in the scene. Using points extracted from static clusters we track the camera. Then, we use the points from remaining potentially dynamic clusters to track the objects. As we estimate the geometry of some objects (e.g. a plane for the road) we are able to constrain the velocity of tracked clusters with mechanical links. To improve this approach we chose to use LiDAR scans in several ways as we can see in the pipeline, figure 6.2. First we feed them to a 3D object detection network that estimates the pose and size of objects in the scene. Second we use successive LiDAR scans corresponding to objects and register them to compute their relative pose. We inject both detected and registered poses as constraints in the BA, the first one being free from any

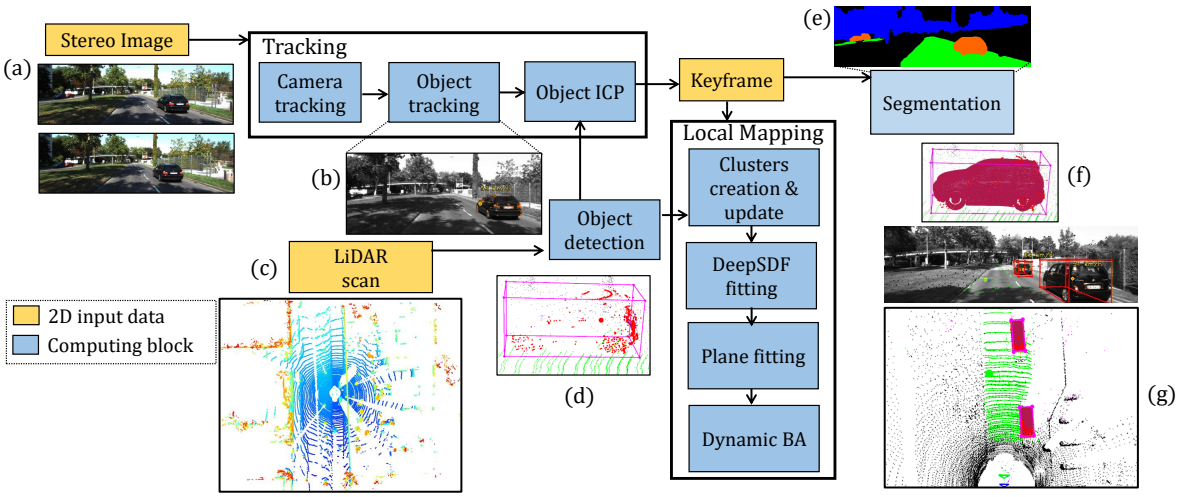

Figure 6.2 – The pipeline of our approach: (a) keypoints are extracted from stereo images and used for camera tracking and (b) object tracking. (c) LiDAR scans are fed toa 3D object detector, allowing us to obtain (d) the 3D bounding box of objects in the scene and to cluster LiDAR points, which are then used in an ICP algorithm. (e) selected keyframes are segmented to create clusters that are augmented with clustered LiDAR points. (f) Object LiDAR points are then used to fit a per object sdf that constrains the geometry of clusters. (g) The trajectory and geometry of clusters is refined in the BA.

drift and the second one more accurate. Third, we follow the work of [J. Wang et al., 2021] and use DeepSDF [J. J. Park et al., 2019] to fit an SDF to objects using LiDAR points. The SDF is then used in the BA to constrain the SLAM map points to lie on the object surface.

6.3.1 A recall of TwistSLAM

In the previous chapter we introduced TwistSLAM. Our algorithm uses the panoptic segmentation of keyframes to create a map containing clusters of 3D points that uniquely correspond to objects in the scene. We use a priori static clusters (e.g. the road, the buildings) to robustly track the camera. Other potentially dynamic clusters are tracked by minimizing their reprojection error. We use the structure of the scene, represented with planes to constrain the movement of objects by projecting twists, that should follow mechanical linkages. Doing so we obtain physically plausible displacements. Finally we optimize all object and camera poses, as well as all map points in a single BA with a constant velocity constraint on objects.

6.3.2 Injecting LiDAR scans in TwistSLAM

To improve the accuracy of object tracking we propose to use LiDAR scans, taken at each timestamp and processed in multiple ways. We first process the scans by using a 3D object detection network (namely 3DSSD [Z. Yang et al., 2020]). For each scan this network yields a set of estimated 3D bounding boxes, with their corresponding 6 DoF pose and size, denoted respectively for the estimations at the i^{th} frame: ${}^{c_i}\mathbf{T}^{d}_{o_i}$ where d stands for detection and $\mathbf{s}_i = (l_i, w_i, h_i)^\top$, where l, w and h denote the length, width and height of the bounding box. We associate clusters created in our SLAM system to object detections by minimizing their 3D distance. A detection is valid if its distance to the cluster centroid is lower than 2 meters. We then use consecutive detections to compute the relative twist ${}^{w}\boldsymbol{\xi}^{d}_{o_i}$ linking two poses:

$$ {}^{w}\boldsymbol{\xi}^{d}_{o_i} = \log({}^{c_{i+1}}\mathbf{T}^{d}_{o_{i+1}} ({}^{c_i}\mathbf{T}^{d}_{o_i})^{-1}) \tag{6.1} $$

The estimation of this twist has the advantage of being free from any drift. It can thus be used to limit the drift accumulated during tracking, similarly to the action of a loop closing step for camera tracking. The drawback however is that it is more noisy than keypoints based tracking. Indeed, the detector was trained to detect 3D objects rather to accurately estimate their pose. We show in section 6.3.4 how to inject it in the BA.

One of the main drawback of TwistSLAM [Gonzalez et al., 2022b] was the lack of canonical pose for objects. Indeed, when an object is first created, its pose is initialized with an identity matrix for the rotation, and the centroid of the cluster for the translation. Thus, this pose does not relate to the pose of objects in their canonical coordinate frame. Using 3D object detection we can estimate the initial object pose. This initial pose is then updated using object tracking. We also fuse the estimated dimensions of the detection using the median of each dimension for robustness. Both the pose and the dimensions allow us to estimate a 3D bounding box for clusters. Using the bounding box we can associate 3D LiDAR points to clusters. We thus obtain at each timestamp a precise 3D scan of each object in the scene. We use the generalized ICP algorithm [Segal et al., 2009] to compute the transformation between consecutive timestamps, that we denote ${}^{o_{i+1}}\mathbf{T}^{r}_{o_i}$ where r stands for registration.

This transformation can be decomposed as:

$$ {}^{o_{i+1}}\mathbf{T}^{r}_{o_i} = {}^{o_{i+1}}\mathbf{T}^{r}_{w} ({}^{o_i}\mathbf{T}^{r}_{w})^{-1} \tag{6.2} $$

which allows us to compute the corresponding twist in the world coordinate frame:

$$ {}^{w}\boldsymbol{\xi}^{r}_{o_i} = \log({}^{o_{i+1}}\mathbf{T}^{r}_{o_i}) \tag{6.3} $$

This twist as the advantage of being accurate compared to keypoint based twists, particularly when keypoints are difficult to extract (e.g. on small, far or textureless objects). We show in section 6.3.4 how to inject it in the global BA.

Finally, note that we also associate LiDAR points to some clusters (e.g. the road) as they can improve the estimation of their geometry. To do so we project LiDAR points in the image and measure the probability that they belong to the class *road* in the segmented image. They are then associated to the cluster if this probability is higher than a threshold.

6.3.3 Estimating clusters geometry

The geometry of objects is an important property that we can inject in the SLAM to improve the accuracy of 3D mapping. To estimate it, we use clustered LiDAR points, that are precise and apply the method developed in DSP-SLAM [J. Wang et al., 2021]. DSP-SLAM uses DeepSDF [J. J. Park et al., 2019] to represent the geometry of an object using an SDF generated from its latent code vector:

$$ G({}^{o}\mathbf{T}_{w}\,{}^{w}\bar{\mathbf{X}}, \mathbf{z}) = s \tag{6.4} $$

where s is the SDF value computed at the 3D points position ${}^{o}\bar{\mathbf{X}} = {}^{o}\mathbf{T}_{w}\,{}^{w}\bar{\mathbf{X}}$ and $\mathbf{z} \in \mathbb{R}^{64}$ is the latent code representing the object shape. An example of a SDF from [J. J. Park et al., 2019] is visible figure 6.3.

They optimize the latent code, object pose and scale so that the generated geometry tightly fits the object LiDAR points. Doing so they can reconstruct a realistic watertight mesh and use the object poses as constraints in the BA. As we already have a good estimate of the objects poses we propose to apply this algorithm on LiDAR points not to refine the pose but rather to refine the SLAM 3D points. We use clustered LiDAR scans to fit the latent code $\mathbf{z}$ exactly as in DSP-SLAM. However contrary to DSP-SLAM we keep the object pose and scale fixed, their values being set using our own estimate of the object pose and length. Then, we seek to constrain 3D map points so that they lie on the object estimated surface. As we have an estimate of the SDF value and of its derivative with respect to the latent code and 3D points position we can apply a gradient descent

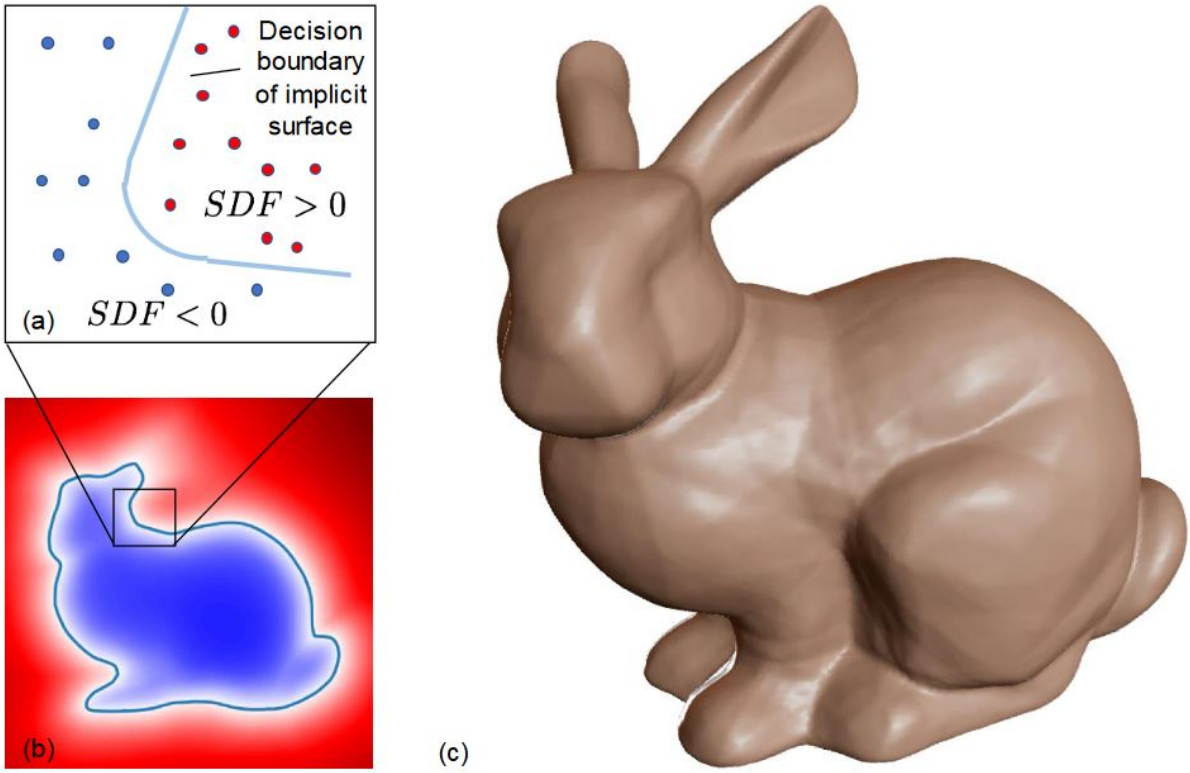

Figure 6.3 – Illustration of the SDF on the Stanford Bunny. (a) Points outside of the mesh have positive SDF values, points inside the mesh have negative SDF values. The surface is represented by the level set SDF = 0. (b) 2D cross section of the SDF for the Stanford Bunny. (c) Rendered surface from the 0 level set of DeepSDF. Image courtesy of DeepSDF [J. J. Park et al., 2019]

algorithm to project points on the surface, which can be written at the k^{th} step:

$$ {}^{o}\mathbf{X}^{(k+1)} = {}^{o}\mathbf{X}^{(k)} - \alpha^{(k)} \frac{\partial G({}^{o}\mathbf{X}^{(k)}, \mathbf{z})}{\partial {}^{o}\mathbf{X}^{(k)}} \tag{6.5} $$

where $\alpha^{(k)}$ is the step size, the point initial value is ${}^{o}\mathbf{X}^{(0)} = {}^{o}\mathbf{X}$ and the derivative of G is obtained through back propagation. This process is repeated for 5 to 10 steps to obtain projected points that we denote ${}^{o}\tilde{\mathbf{X}}$. Projected points will then be used as anchors in the bundle adjustment to constrain 3D points to be coherent with the estimated geometry. The use of LiDAR data is particularly interesting to constrain map points as LiDAR points are usually more precise than points triangulated from stereo images. We show in figure 6.4 and example of rendered mesh, LiDAR points and projected points.

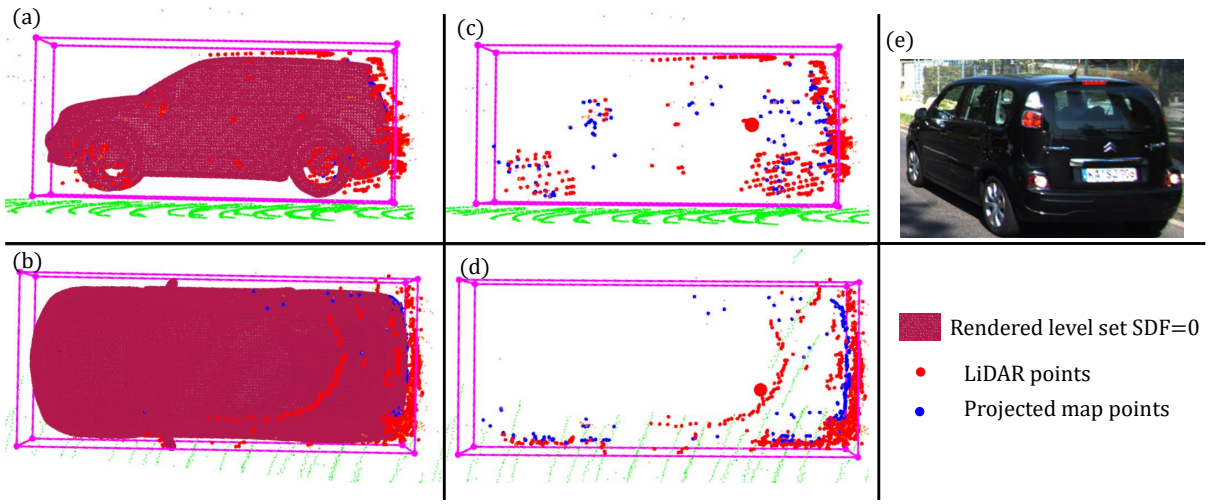

Figure 6.4 – Rendered SDF surface seen from the (a) top and (b) side. 3D LidAR points (red) and projected points (blue) seen from the (c) top and (d) side. (e) RGB image of the reconstructed cars.

6.3.4 Dynamic Bundle Adjustment

In this section we use the BA to constrain object trajectories to be coherent with twists estimated by the registration and detection steps. We also constrain object map points to be coherent with the estimated SDF of each object.

Our bundle adjustment cost function can be written as follows:

$$E(\{{}^{w}\tilde{\boldsymbol{\xi}}_o, {}^{c}\mathbf{T}_w, {}^{w}\mathbf{X}, {}^{o}\mathbf{X}\}) = \sum_{i,j} e^{i,j}_{stat} + \sum_{i,j} e^{i,j}_{dyna} + \sum_{i} e^{i}_{const} + \sum_{i} e^{i}_{det} + \sum_{i} e^{i}_{reg} + \sum_{j} e^{j}_{geo} \quad (6.6)$$

where $e^{i,j}_{stat}$ is the classical static reprojection error:

$$e^{i,j}_{stat} = \rho(||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w, {}^{w}\mathbf{X}_j)||_{\Sigma^{-1}_{i,j}})$$

$e^{i,j}_{dyna}$ is the dynamic reprojection error:

$$e^{i,j}_{dyna} = \rho(||{}^{i}\mathbf{x}_j - p({}^{c_i}\mathbf{T}_w \exp(\mathbf{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{o_i}\mathbf{T}_w, {}^{o}\mathbf{X})||_{\Sigma^{-1}_{i,j}})$$

where $\Sigma^{-1}_{i,j}$ is estimated using the MAD.

e^{i}_{const} is the constant velocity model that penalizes twists variations by linking 3 consecutive poses:

$$e^i_{const} = \rho(||\mathbf{\Pi}^w\boldsymbol{\xi}_{o_i} - \mathbf{\Pi}^w\boldsymbol{\xi}_{o_i}||_{\mathbf{W}}) \tag{6.7}$$

where $\mathbf{W}$ is a diagonal weight matrix used to balance the errors, ${}^w\boldsymbol{\xi}_{o_{i+1}}$ is the twist linking the poses $\exp(\mathbf{\Pi}^{o_i}\tilde{\boldsymbol{\xi}}_w){}^w\mathbf{T}_{o_i}$ and $\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i+1}}){}^w\mathbf{T}_{o_{i+1}}$ and ${}^w\boldsymbol{\xi}_{o_i}$ is the twist linking the poses $\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}){}^w\mathbf{T}_{o_{i-1}}$ and $\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i}$.

The error e^i_{det} penalizes the difference with twists estimated from the object detection network:

$$e^i_{det} = \rho(||\mathbf{\Pi}^w\boldsymbol{\xi}^d_{o_i} - \mathbf{\Pi}^w\boldsymbol{\xi}_{o_i}||_{\mathbf{W}_{det}})$$

where $\mathbf{W}_{det}$ is a diagonal weight matrix and ${}^w\boldsymbol{\xi}_{o_i} = \log(\exp(\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_i}){}^w\mathbf{T}_{o_i}{}^{o_{i-1}}\mathbf{T}_w\exp(-\mathbf{\Pi}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}))$ is the twist linking two consecutive poses.

Similarly, the error e^i_{reg} penalizes the difference with twists estimated by registering consecutive point clouds:

$$e^i_{reg} = \rho(||\mathbf{\Pi}^w\boldsymbol{\xi}^r_{o_i} - \mathbf{\Pi}^w\boldsymbol{\xi}_{o_i}||_{\mathbf{W}_{reg}})$$

where $\mathbf{W}_{reg}$ is a diagonal weight matrix.

Finally, the residual e^i_{geo} constrains points to lie on the estimated geometry surface by penalizing the difference between the position of 3D points and of their projected counterpart:

$$e^j_{geo} = \rho(||{}^o\tilde{\mathbf{X}}_j - {}^o\mathbf{X}_j||_{\mathbf{W}}) \tag{6.8}$$

Note that we could also directly use the DeepSDF function $G({}^o\mathbf{X}, \mathbf{z})$ to compute the value of the residual and to obtain the Jacobian via back propagation.

6.3.5 Computing the Jacobians

The Jacobians of the registration and detection errors are the same as the only changes between both do not depend on the optimized twists.

$$\mathbf{J}_{reg} = \begin{pmatrix} \frac{\partial e^i_{reg}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_{i-1}}} & \frac{\partial e^i_{reg}}{\partial {}^w\tilde{\boldsymbol{\xi}}_{o_i}} \end{pmatrix} \tag{6.9}$$

The derivative with respect to ${}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}$ can be written:

$$\frac{\partial e^i_{reg}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}} = \boldsymbol{\Pi}\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}(\mathbf{I}_4 \otimes \mathbf{R})\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} \tag{6.10}$$

with $\mathbf{T} = \exp((\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{w}\mathbf{T}_{o_i})(\exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})\,{}^{w}\mathbf{T}_{o_{i-1}})^{-1}$, $\mathbf{R}$ is the rotation matrix of $\exp((\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{w}\mathbf{T}_{o_i})({}^{w}\mathbf{T}_{o_{i-1}})^{-1}$. The derivative with respect to ${}^{w}\tilde{\boldsymbol{\xi}}_{o_i}$ can be written:

$$\frac{\partial e^i_{reg}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}} = -\boldsymbol{\Pi}\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}({\mathbf{T}_{\mathbf{B}}}^{\top} \otimes \mathbf{I}_3)\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi} \tag{6.11}$$

with $\mathbf{T} = \exp((\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{w}\mathbf{T}_{o_i})(\exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})\,{}^{w}\mathbf{T}_{o_{i-1}})^{-1}$ and $\mathbf{T_B} = {}^{w}\mathbf{T}_{o_i}\,{}^{o_{i-1}}\mathbf{T}_{w}\exp(-\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})$.

Proof. First we show how to compute the derivative with respect to ${}^{w}\boldsymbol{\xi}_{o_{i-1}}$. We can write:

$$\begin{aligned}
\boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i} &= \boldsymbol{\Pi}\log(\exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{w}\mathbf{T}_{o_i}(\exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})\,{}^{w}\mathbf{T}_{o_{i-1}})^{-1}) && (6.12)\\
&= \boldsymbol{\Pi}\log(\exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{w}\mathbf{T}_{o_i}({}^{o_{i-1}}\mathbf{T}_{w}\exp(-\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}))) && (6.13)\\
&= \boldsymbol{\Pi}\log(f_1(f_2({}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}))) && (6.14)
\end{aligned}$$

where $f_1(\mathbf{T}) = \exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{w}\mathbf{T}_{o_i}\,{}^{o_{i-1}}\mathbf{T}_{w}\mathbf{T}$ is the left multiplication by the pose $\exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{w}\mathbf{T}_{o_i}\,{}^{o_{i-1}}\mathbf{T}_{w}$ and $f_2({}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}) = \exp(-\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})$. Using the chain rule it follows that:

$$\left.\frac{\partial \boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0} = \boldsymbol{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}\left.\frac{\partial f_1(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{B}}\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0}(-\boldsymbol{\Pi}) \tag{6.15}$$

where $\mathbf{A} = \exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_i})\,{}^{w}\mathbf{T}_{o_i}(\exp(\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})\,{}^{w}\mathbf{T}_{o_{i-1}})^{-1}$ and $\mathbf{B} = \exp(-\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})$ Using the derivative of left multiplication transformation, we can write:

$$\left.\frac{\partial \boldsymbol{\Pi}\,{}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0} = \boldsymbol{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}(\mathbf{I}_4 \otimes \mathbf{R})\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\boldsymbol{\Pi}\,{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0}(-\boldsymbol{\Pi}) \tag{6.16}$$

where $\mathbf{R}$ is the rotation part of $\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i})^{w}\mathbf{T}_{o_i}{}^{o_{i-1}}\mathbf{T}_{w}$. We thus have:

$$\left.\frac{\partial e^i_{reg}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0} = -\left.\frac{\partial \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0} \tag{6.17}$$

$$= \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}} (\mathbf{I}_4 \otimes \mathbf{R})\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}=0}\mathbf{\Pi} \tag{6.18}$$

We then show how to compute the derivative with respect to ${}^{w}\tilde{\boldsymbol{\xi}}_{o_i}$. We can write:

$$\mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i} = \mathbf{\Pi}\log(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i})^{w}\mathbf{T}_{o_i}(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})^{w}\mathbf{T}_{o_{i-1}})^{-1}) \tag{6.19}$$

$$= \mathbf{\Pi}\log(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i})^{w}\mathbf{T}_{o_i}({}^{o_{i-1}}\mathbf{T}_{w}\exp(-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}}))) \tag{6.20}$$

$$= \mathbf{\Pi}\log(f_1(f_2({}^{w}\tilde{\boldsymbol{\xi}}_{o_i}))) \tag{6.21}$$

where $f_1(\mathbf{T}) = \mathbf{T}{}^{w}\mathbf{T}_{o_i}(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})^{w}\mathbf{T}_{o_{i-1}})^{-1}$ is the right multiplication by the pose ${}^{w}\mathbf{T}_{o_i}(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})^{w}\mathbf{T}_{o_{i-1}})^{-1}$ and $f_2({}^{w}\tilde{\boldsymbol{\xi}}_{o_i}) = \exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i})$. Using the chain rule it follows that:

$$\left.\frac{\partial \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} = \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}\left.\frac{\partial f_1(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{B}}\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0}\mathbf{\Pi} \tag{6.22}$$

where $\mathbf{A} = \exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i})^{w}\mathbf{T}_{o_i}(\exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})^{w}\mathbf{T}_{o_{i-1}})^{-1}$ and $\mathbf{B} = \exp(\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i})$ Using the derivative of right multiplication transformation, we can write:

$$\left.\frac{\partial \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} = \mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}(\mathbf{T_B}^{\top} \otimes \mathbf{I}_4)\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0}\mathbf{\Pi} \tag{6.23}$$

where $\mathbf{T_B} = {}^{w}\mathbf{T}_{o_i}{}^{o_{i-1}}\mathbf{T}_{w}\exp(-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_{i-1}})$. We thus have:

$$\left.\frac{\partial e^i_{reg}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} = -\left.\frac{\partial \mathbf{\Pi}^{w}\boldsymbol{\xi}_{o_i}}{\partial {}^{w}\tilde{\boldsymbol{\xi}}_{o_i}}\right|_{{}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0} \tag{6.24}$$

$$= -\mathbf{\Pi}\left.\frac{\partial \log(\mathbf{T})}{\partial \mathbf{T}}\right|_{\mathbf{T}=\mathbf{A}}(\mathbf{T_B}^{\top} \otimes \mathbf{I}_3)\left.\frac{\partial \exp(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=-\mathbf{\Pi}^{w}\tilde{\boldsymbol{\xi}}_{o_i}=0}\mathbf{\Pi} \tag{6.25}$$

■

As the residual e_{geo} is linear in ${}^{o}\mathbf{X}$ its derivative with respect to ${}^{o}\mathbf{X}$ is immediate:

$$\frac{\partial e^{j}_{geo}}{\partial {}^{o}\mathbf{X}_{j}} = -\mathbf{I}_{3} \tag{6.26}$$

6.4 Experiments

In this section we present the experiments we conducted to test our approach. We evaluate both the accuracy of the camera pose estimation and of the object pose estimation.

6.4.1 Experiments details

Similarly to the previous chapter we evaluate our system on the KITTI [Geiger et al., 2012] tracking dataset using the ATE, RPE and MOTP [J. Sturm et al., 2012; Z. Zhang and Scaramuzza, 2018]. Object detection is performed using the implementation of 3DSSD [Z. Yang et al., 2020] in the framework mmdetection3d [Contributors, 2020]. The ICP is performed using the implementation of open3d in C++ [Zhou et al., 2018].

6.4.2 Camera pose estimation

In this subsection we evaluate the accuracy of our camera pose estimation. We report the results in table 6.1. As we obtain almost exactly the same results as TwistSLAM we only show here the results on some sequences. This is not surprising as the sequences only exhibit a mild amount of dynamicity, that is well dealt even by non dynamic approaches [Mur-Artal and Tardós, 2017]. Furthermore in this approach we rather focused on the accuracy of object tracking, that we improve compared to state of the art, as shown in the following paragraph.

6.4.3 Object pose estimation.

In this paragraph we evaluate the accuracy of object pose estimation. We report the results in table 6.2 and table 6.3. As we can see we often obtain better results in terms of object tracking accuracy for the ATE and RPE compared to [Bescos et al., 2021] and [Gonzalez et al., 2022b]. The TP and MOTP metrics on the other hand vary. On most sequences they show very similar scores to TwistSLAM, which can be expected

Table 6.1 – Camera pose estimation comparison on the Kitti tracking dataset.

seq	ORB-SLAM2 [Mur-Artal and Tardós, 2017]		DynaSLAM [Bescos et al., 2018]		DynaSLAM2 [Bescos et al., 2021]		TwistSLAM [Gonzalez et al., 2022b]		TwistSLAM++	
	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)	RPE_t (m/f)	RPE_R (°/f)
00	**0.04**	0.06	**0.04**	0.06	**0.04**	0.06	**0.04**	**0.05**	**0.04**	**0.05**
02	0.04	0.03	0.04	0.03	0.04	**0.02**	0.03	0.03	0.03	**0.02**
03	0.07	0.04	0.07	0.04	0.06	0.04	0.06	**0.02**	0.06	0.03
04	0.07	0.06	0.07	0.06	0.07	0.06	0.06	**0.04**	0.06	**0.04**
05	0.06	0.03	0.06	0.03	0.06	0.03	0.06	**0.02**	0.06	**0.02**
10	**0.07**	0.04	**0.07**	0.04	**0.07**	0.03	**0.07**	0.03	**0.07**	**0.02**
11	0.04	0.03	0.04	0.03	0.04	0.03	**0.03**	**0.02**	**0.03**	**0.02**
14	**0.03**	0.08	**0.03**	0.08	**0.03**	0.08	**0.03**	**0.06**	**0.03**	**0.06**
18	0.05	0.03	0.05	0.03	0.05	**0.02**	0.04	**0.02**	0.04	**0.02**
mean	0.052	0.044	0.052	0.046	0.051	0.041	**0.041**	**0.032**	**0.041**	**0.032**

Table 6.2 – Object pose estimation comparison on the Kitti tracking dataset. ATE is in m, RPE_t in m/m, RPE_R in °/m

	DynaSLAM 2 [Bescos et al., 2021]			TwistSLAM			TwistSLAM++		
seq / obj. id / class	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R
03 / 1 / car	0.69	0.34	1.84	0.31	**0.10**	0.28	**0.23**	0.11	**0.19**
05 / 31 / car	0.51	0.26	13.5	0.58	0.35	**0.19**	**0.09**	**0.07**	0.28
10 / 0 / car	0.95	0.40	2.84	0.77	0.21	1.98	**0.05**	**0.10**	**0.96**
11 / 0 / car	1.05	0.43	12.51	0.17	**0.23**	0.23	**0.15**	0.28	**0.21**
11 / 35 car	1.25	0.89	16.64	0.10	0.03	0.11	0.11	**0.02**	**0.09**
18 / 2 / car	1.10	0.30	9.27	**0.21**	0.27	0.66	**0.29**	**0.09**	**0.32**
18 / 3 / car	1.13	0.55	20.05	0.15	0.21	0.56	**0.13**	**0.10**	**0.37**
19 / 63 / car	0.86	1.45	48.80	**0.28**	2.17	**1.08**	0.34	**0.21**	**0.31**
19 / 72 / car	0.99	1.12	3.36	0.16	0.05	**0.34**	**0.09**	**0.03**	0.37
20 / 0 / car	0.56	0.45	1.30	**0.17**	0.20	0.72	0.30	**0.21**	**0.35**
20 / 12 / car	1.18	0.40	6.19	**0.24**	**0.20**	1.54	0.80	0.54	**0.64**
20 / 122 / car	0.87	0.72	5.75	0.17	**0.02**	0.07	**0.16**	**0.02**	**0.06**
mean	0.93	0.61	11.83	0.26	0.32	0.68	**0.23**	**0.15**	**0.35**

as the MOTP metrics only require an overlap of 0.25 for a detection to be positive, thus an improvement of a few centimeters on the pose does not translate to new positive detections. A way to do so would be to decrease the number of points required for tracking to track objects for longer periods. This however is out of scope of our work as we focus on improving tracking accuracy. On some sequences however we obtain lower 3D and birdview MOTP for a similar 2D MOTP. Those differences are mainly due to wrong pose estimates that happen when the cluster is first created far from the camera. Those differences can also be explained by the fact that TwistSLAM uses the groundtruth initial bounding box of objects, which is noise-free while we use a 3D object detector. Finally on some sequences (such as 11/35 and 20/0) the additional LiDAR information allows us to improve tracking stability and thus to track on longer trajectories, increasing the MOTP without increasing the ATE or RPE.

6.4.4 Ablation study

We propose in table 6.4 an ablation study comparing the impact of each constraint on the accuracy of object pose estimation. As we can see we obtain the best results when all constraints are activated. However the improvement brought by each constraint varies. On the one hand the SDF has a low impact on the accuracy and the improvements brought by the 3D detector and the ICP are mild. However we argue that this can be caused by

Table 6.3 – Object pose estimation comparison on the Kitti tracking dataset. TP and MOTP are in %.

	DynaSLAM 2 [Bescos et al., 2021]						TwistSLAM						TwistSLAM++					
seq / obj. id / class	2D TP	2D MOTP	BV TP	BV MOTP	3D TP	3D MOTP	2D TP	2D MOTP	BV TP	BV MOTP	3D TP	3D MOTP	2D TP	2D MOTP	BV TP	BV MOTP	3D TP	3D MOTP
03 / 1 / car	50.0	**71.79**	39.34	56.61	38.53	48.20	**58.02**	60.00	**58.02**	**60.00**	**58.02**	**45.00**	**56.79**	60.00	**56.79**	**60.00**	**56.79**	**60.00**
05 / 31 / car	28.96	**60.30**	14.48	**46.84**	11.45	34.20	**30.84**	35.00	**30.84**	35.00	**30.84**	**35.00**	30.00	26.64	16.32	14.95	16.10	14.49
10 / 0 / car	**81.63**	**73.51**	**70.41**	**47.60**	**68.37**	**40.28**	7.20	3.70	6.10	3.10	5.80	2.80	6.48	10.00	6.48	10.00	6.48	10.00
11 / 0 / car	**72.65**	**74.78**	**61.66**	**50.74**	**52.28**	**47.35**	29.61	32.50	29.61	32.50	29.61	32.50	26.82	30.00	26.82	30.00	26.82	30.00
11 / 35 car	53.17	65.25	19.05	31.95	6.35	26.02	**65.00**	**67.50**	**65.00**	**67.50**	**65.00**	**67.50**	**73.75**	**77.50**	**73.75**	**77.50**	**73.75**	**77.50**
18 / 2 / car	**86.36**	74.81	67.05	45.47	62.12	34.80	84.67	**87.50**	**84.67**	**87.50**	**84.67**	**87.50**	85.18	**87.50**	**85.18**	**87.50**	**85.18**	**87.50**
18 / 3 / car	**53.33**	**70.94**	21.75	**41.45**	16.84	**35.80**	28.19	30.00	**28.19**	30.00	**28.19**	30.00	21.83	25.00	21.83	25.00	21.83	25.00
19 / 63 / car	35.26	63.50	29.48	45.69	26.48	33.89	**65.93**	**70.00**	**65.93**	**70.00**	36.26	20.64	**65.93**	**70.00**	**65.93**	**70.00**	**65.93**	**70.00**
19 / 72 / car	**29.11**	**62.59**	**29.43**	**55.48**	**29.43**	**39.81**	16.92	20.00	16.92	20.00	16.92	20.00	5.38	10.00	5.38	10.00	5.38	10.00
20 / 0 / car	63.68	78.54	43.78	45.00	31.84	46.15	84.75	87.50	84.75	87.50	84.75	87.50	**93.22**	**97.50**	**93.22**	**97.50**	**93.22**	**97.50**
20 / 12 / car	**42.77**	**76.77**	**37.64**	**49.29**	**36.23**	**40.81**	14.24	17.5	13.91	17.45	13.04	17.25	32.75	37.5	32.75	37.5	32.75	37.5
20 / 122 / car	34.90	78.76	34.51	48.05	29.02	44.43	**84.94**	**87.50**	**84.94**	**87.50**	**84.94**	**87.50**	**84.94**	**87.50**	**84.94**	**87.50**	**84.94**	**87.50**
mean	**55.15**	**70.96**	39.05	47.01	34.08	39.31	45.53	49.89	47.41	49.84	44.84	43.18	49.42	51.6	**47.45**	**50.62**	**47.43**	**50.58**

Table 6.4 – Ablation study of different constraints.

Seq. 3 obj. 0	All constraint	No detection constraint	No joint constraint	No ICP	No sdf constraint	No soft trajectory constraint	No constraint
ATE	0.43	0.47	0.52	0.45	0.49	0.46	0.48
RPE_t	0.05	0.06	0.15	0.06	0.06	0.11	0.07
RPE_R	0.17	0.17	0.63	0.17	0.19	0.70	0.40
Seq. 3 obj. 1	All constraints	No detection constraint	No joint constraint	No ICP	No sdf constraint	No soft trajectory constraint	No constraint
ATE	0.26	0.33	0.31	0.34	0.30	0.32	0.39
RPE_t	0.06	0.06	0.11	0.15	0.06	0.14	0.11
RPE_R	0.26	0.23	0.44	0.21	0.21	0.56	0.57

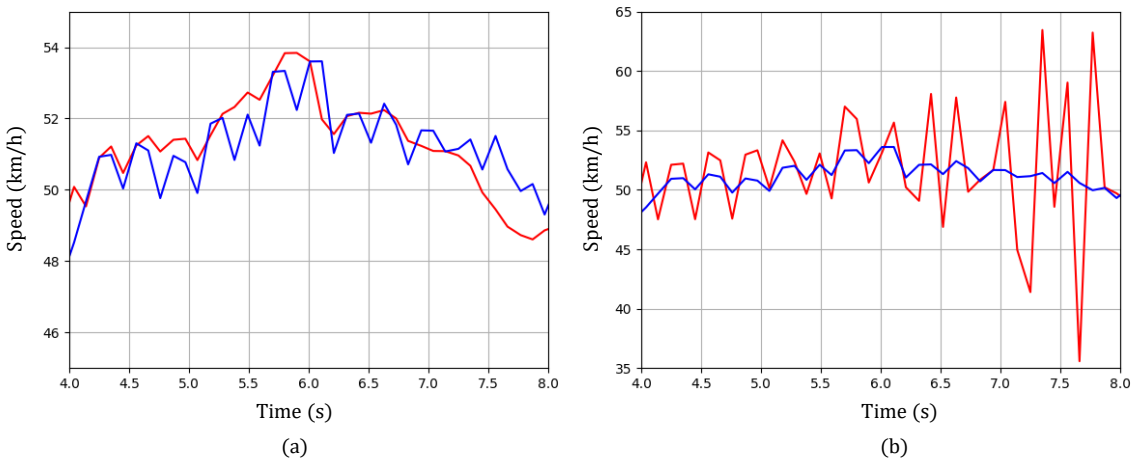

Figure 6.5 – Comparison of tracking speed evolution with (a) and without (b) soft constraints on twists. Please note that the scale of the y axis changes from one graph to the other.

the lack of accuracy of the ground truth, which we discuss below. Furthermore the object detection network yields noisy estimates as it was not trained to predict precise object pose estimations. As we track most objects on short durations the accumulated drift is not high enough to see the benefits of the constraints from the 3D detector. On the other hand the joint constraints and soft trajectory constraints improves the accuracy of object pose estimation by a large margin. We show in figure 6.5 a comparison of the evolution of speed with and without soft constraints to illustrate their importance to obtain physically plausible results.

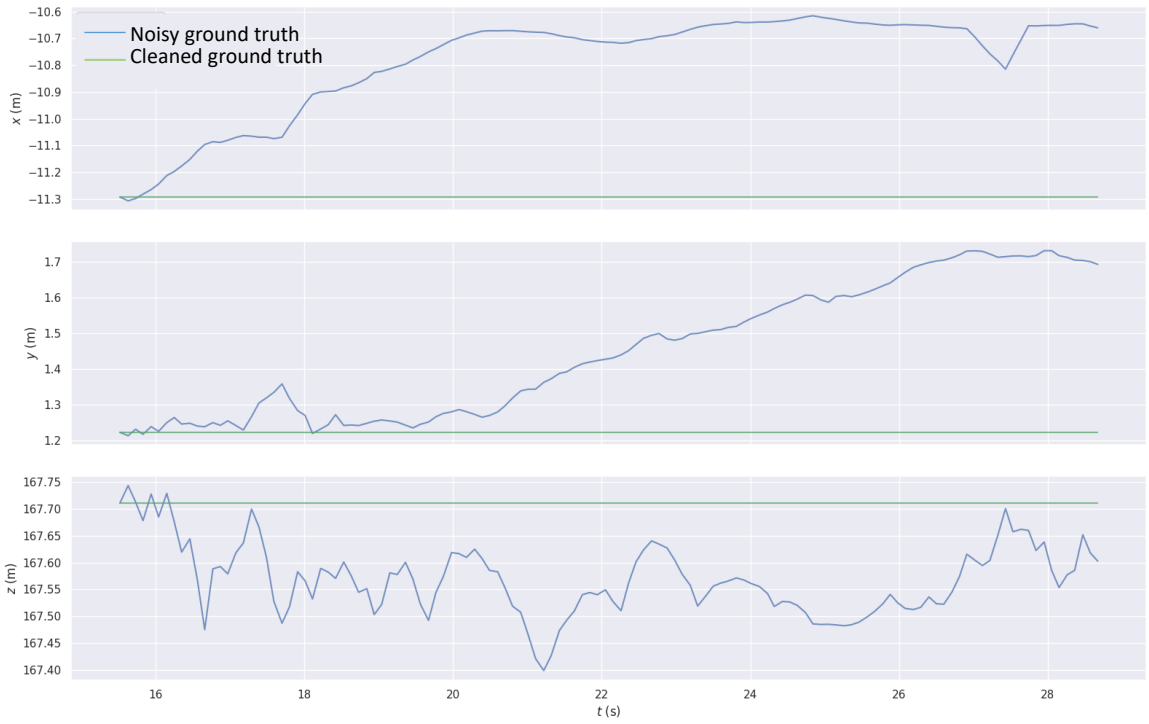

Figure 6.6 – Illustration of groundtruth pose noise

A note on the accuracy of the ground truth: As the ground truth for object poses was obtained by manual annotations using mechanical turk we expect its accuracy to be limited. Hence we chose to evaluate it. To do so we chose a static car in sequence 11 and repeated the first groundtruth pose for all its trajectory, thus obtaining a more coherent ground truth. We then computed the APE and RPE: the RMSE of the APE was 0.27 m, the median of the translational RPE 0.03 m/f and the median of the rotational RPE 0.08°/f. We show in figure 6.6 a comparison of the noisy and cleaned ground truth trajectories decomposed on three axis. As we can see the static car moves by about 0.5 m in each direction. This limits the evaluation we are able to do, thus the results should be analyzed cautiously. We argue that for future research it may be necessary to improve the accuracy of this dataset. The Oxford multimotion dataset [Judd and Gammell, 2019] is an interesting recent dataset that contains accurate ground truth for camera and object poses thanks to a vicon motion capture system. However it also shows some limitations, such as a limited variability in terms of scenes and available objects.

6.5 Qualitative results

In addition to the first figure, we also show here qualitative results. Figures 6.7 and 6.8 show examples of tracked and reconstructed cars on sequences 11 and 0 of the kitti tracking dataset. As we can see most cars have been well detected, their pose correctly estimated and their shape recovered, even when the cars are partially occluded.

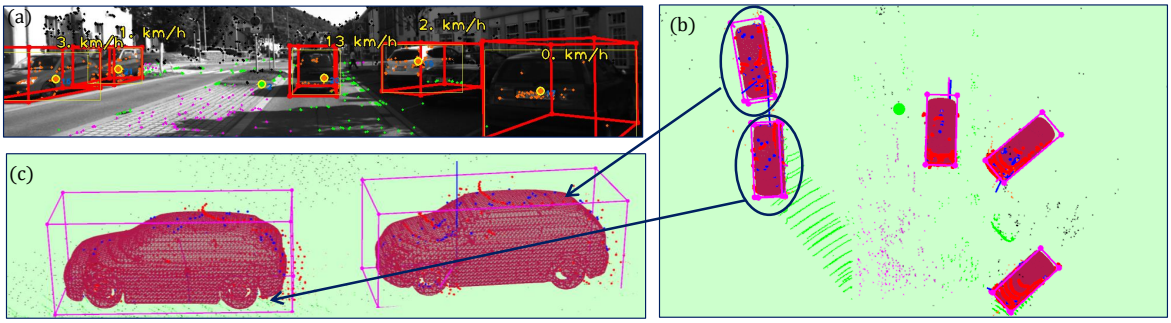

Figure 6.7 – (a) Frame with detected objects, bounding boxes and speed.(b) Map with tracked objects, seen from above. (c) Mesh of reconstructed cars with bounding boxes, LiDAR points (red) and projected points on the mesh (blue).

We also show in figure 6.9 an example of ICP alignment on a car scan. As we can see the two consecutive point clouds in red and blue are correctly aligned. A comparison of pointclouds before and after alignment for this car is visible in figure 6.10. As wee can see the scans are more tightly aligned after the ICP alignment, particularly on the side of the car.

We show in figure 6.4 an example of reconstruction and projected points. As we can see the projected points lie on the object surface. However we can also see that the mesh of the object does not perfectly fit the LiDAR scan, improving the fitting step may lead to better performances. Furthermore we note that the object mesh is not supported by the road plane. Adding a support constraint could prove to be useful.

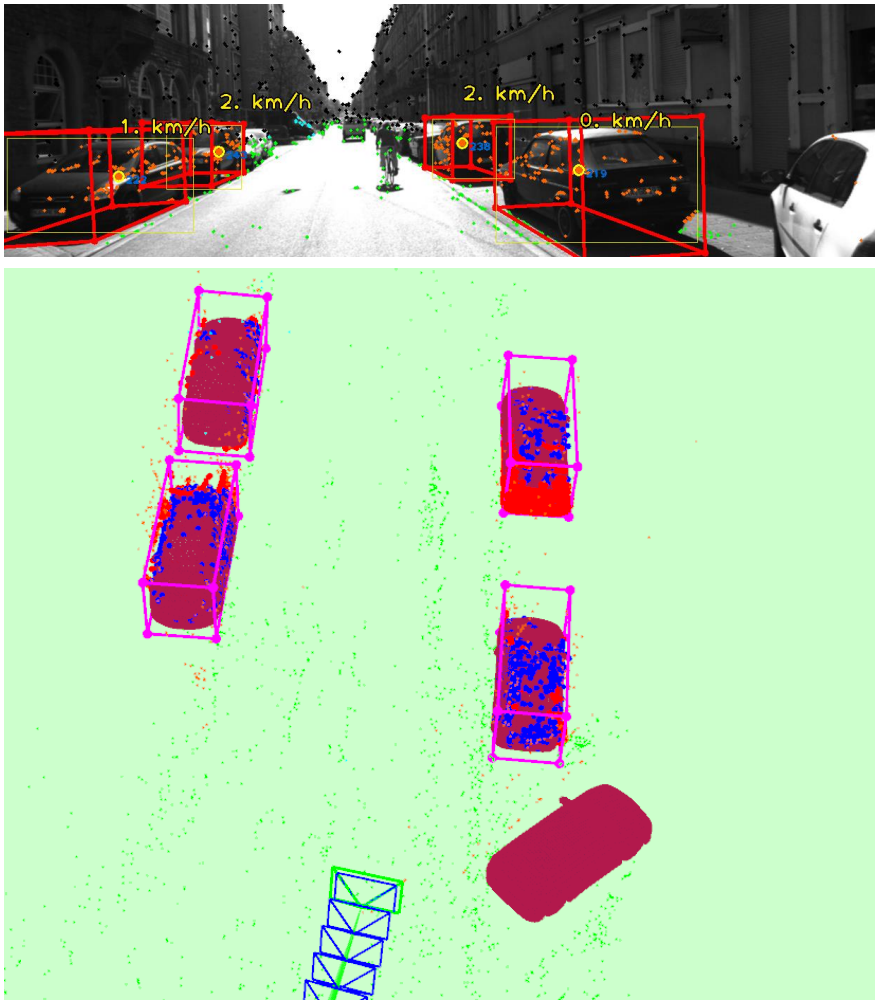

Figure 6.8 – (top) Frame with detected objects, bounding boxes and speed.(bottom) Map with tracked objects, seen from above, with bounding boxes, LiDAR points (red) and projected points on the mesh (blue). Note that the rightmost car (in white) tracking has been lost due to camera motion, however we still show its mesh in the map.

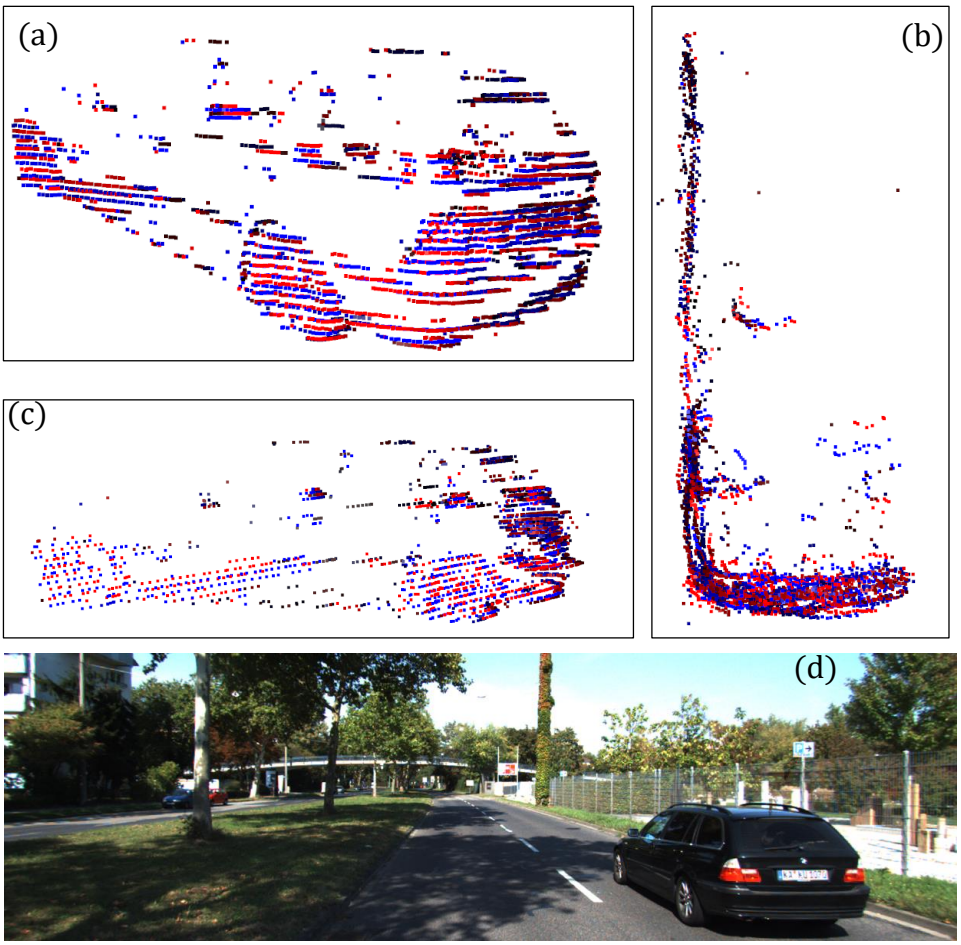

Figure 6.9 – Aligned LiDAR pointclouds (red and blue) seen from (a) ¾, (b) top, (c) side and (d) corresponding car seen in the RGB frame.

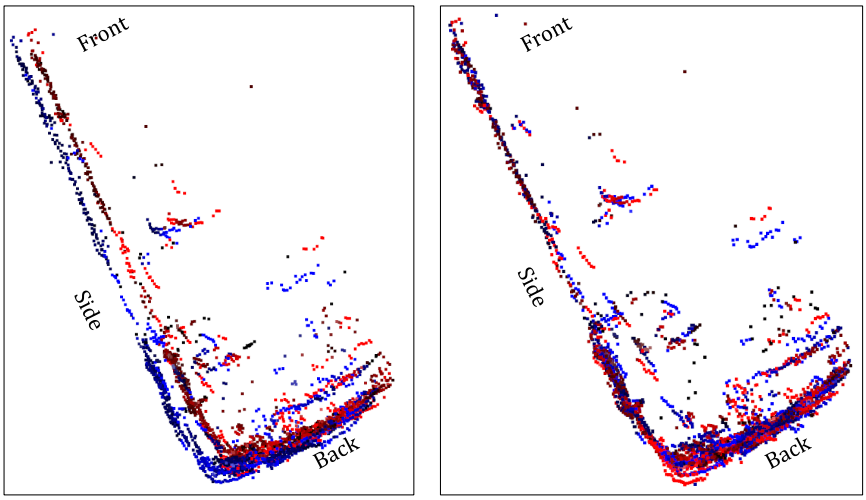

Figure 6.10 – (left) Consecutive scans aligned using RGB based tracking. (right) Scans aligned using the generalized ICP algorithm.

6.6 Conclusion

In this chapter we proposed an improvement over our previous work, that we call TwistSLAM++. Our approach integrates 3D LiDAR scans into a dynamic SLAM system to improve object tracking accuracy and obtain an estimation of object poses in their canonical coordinate frame. To do so we use a 3D object detector that estimates aligned 3D bounding boxes corresponding to objects in the scene. Following, we associate detections to tracked clusters to obtain their canonical pose and cluster LiDAR scans. We use consecutive LiDAR scans to estimate their relative pose and constrain their trajectory in the BA. Finally, we fit a 3D SDF for each object using LiDAR points and use it to constrain map points to lie on the object surface. We show that this approach allows us to improve object tracking while estimating their canonical pose. However we also show that the groundtruth pose of objects is noisy, which limits our evaluations.

Chapter 7

Conclusion and perspectives

7.1 Conclusion

In this manuscript we have investigated the introduction of semantic information into SLAM systems. SLAM is a fundamental problem that is the keystone of many popular fields such as augmented reality, robotics and self-driving vehicles. Many works have pushed the boundaries of SLAM accuracy and scale and it is possible today to precisely track a camera in many scenarios. However it is still limited by one of its main assumption, which is that the world is a single rigid body made of low-level 3D points. This assumption can prevent the SLAM from yielding precise estimations in some scenarios (e.g. dynamic scenes). It can also limit its use cases, due to the *semantic gap* between the map and the environment in which the system evolves. To close this gap we propose to represent the map as a scene graph. In our graph, objects correspond to vertices and have different properties depending on their class. They are also linked to each other by edges that also depend on semantic classes. Using this higher-level, more realistic representation, we show that we can further push the boundaries of SLAM.

In summary, in this manuscript we first presented the fundamentals required to understand our work: how 3D points can be expressed in the camera coordinate frame to be projected in images, associated with 2D keypoints and robustly optimized with camera poses. We also gave a short introduction to semantic information in the age of deep learning. Second we developed the pipeline of a SLAM system starting with the bundle adjustment. Then we summarized the state-of-the-art of semantic SLAM according to the way they use semantic information. Afterwards we developed our vision to go from a low level geometrical, homogeneous, rigid map to a scene graph made of objects with properties and relationships. To do so we first developed an object pose estimation system **L6DNet**. This deep learning based algorithm can estimate the relative pose between the camera and an object in the scene, which can serve as a high level landmark for a

classical SLAM system. We predict 3D offsets from local patches to estimate the 3D position of points that are used to recover the object pose. This strategy allows us to train a light network using little resources while reaching state-of-the-art performances. The main limit of this network is that it needs to be trained for specific objects. Even if the training is short this can be problematic for some applications. Hence we chose to tackle more generic objects by building **S^{3}LAM**. Our SLAM system can create a map of clusters that correspond to objects in the scene. For some a priori chosen objects we estimate their geometry using planes. This allows us to obtain a high level semantic map and to improve camera pose estimation by constraining cluster points to lay on planes. However this approach cannot deal with dynamic objects in the scene. To solve this problem we use our structured map and show with **TwistSLAM** that we can accurately estimate camera pose in dynamic scenes while tracking all moving objects at the same time. We use mechanical links to constrain their movement, which improve their pose estimation. To further improve our work we propose **TwistSLAM++** by injecting LiDAR scans into **TwistSLAM**. We use a 3D object detection network to estimate object canonical poses and estimate relative object transformations with an ICP algorithm. We inject both those estimations in the BA to improve object tracking. Furthermore, using a deep learnt SDF we estimate the shape of objects that we use to constrain object 3D points in the BA. We showed that this approach further improved object tracking accuracy.

7.2 Limitations and perspectives

In this manuscript we successfully developed a SLAM system with the ability to track objects within the scene, improving at the same time camera pose estimation in dynamic environments. While efficient, our approach still shows some limitations that could be fixed on the short-term:

- **Far and small object tracking.** While experimenting on our system TwistSLAM we have seen that tracking far objects was not giving accurate results. We argue that this is due to the nature of our feature based approach. When objects are far or small they do not exhibit large highly textured areas and thus only few keypoints can be extracted and used for tracking. To solve this problem we propose to use instead a direct approach, minimizing the photometric error on pixels. An interesting approach is DOT [Ballester et al., 2021] that compares camera and ob-

ject relative poses to decide whether objects are dynamic or static. However this approach does not refine object trajectories in the bundle adjustment and does not focus on estimating object trajectories but rather on masking moving objects.

- **Continuous trajectory tracking.** In TwistSLAM we estimate the twist and pose of objects at each timestamp. However those estimates are samples of a continuous and differentiable function. Thus, it may be interesting to rather estimate a continuous function [Cioffi et al., 2022; Tirado and Civera, 2022]. We argue it could improve object tracking by removing the need of a regularizer to obtain a smooth trajectory. Indeed, instead of estimating a twist at each timestamp we could decompose its evolution on a well chosen basis of functions to reduce its number of degrees of freedom and give it properties such as continuity and differentiability.

- **Non-rigid object tracking.** In addition to being able to differentiate static and dynamic objects, semantic information can differentiate rigid and non-rigid objects. Furthermore semantic information may allow to estimate mechanical properties for different classes of objects and improve non rigid tracking. For example for a deforming sports ball or a deforming pillow.

- **Human tracking.** We have shown that our approach can accurately track pedestrians. However it works only because pedestrians are partly rigid. This does not work when humans are going under more sophisticated motions or are too small in the image. However being able to accurately track humans is an important task. Indeed they are often present in many scenes and should be detected for security reasons. An interesting strategy would be to integrate a human pose estimation network into a SLAM system to constrain the deformation of humans 3D points and improve both SLAM and human pose estimation. The very recent work BodySLAM [Henning et al., 2022] is a first interesting step towards that goal.

- **Texture generation.** Some recent work [Siddiqui et al., 2022] shows that a network can be trained to generate textures and apply them on 3D models. This approach could be used to generate texture of reconstructed objects and add a photometric consistency error to track objects.

We now introduce some long-term perspectives that could improve the accuracy of semantic SLAM in dynamic environments. The past couple years have witnessed the rise of two major new approaches in computer vision: *RAFT: Recurrent All Pairs Field Transforms for Optical Flow* [Teed and Deng, 2020] and *NeRF: Neural Radiance Fields* [Mildenhall et al., 2020]. Both those papers have gained an important popularity to solve two different problems. The first one extracts features from images using a CNN. Those features are then used to build correlation volumes corresponding to the inner products of features. A recurrent neural network then uses the volumes to iteratively build the outputs of the system which correspond the optical flow of images.

The second one, NeRF uses a Multi Layer Perceptron to predict, for each 3D point in a scene, the color and volume density of this point. By estimating those values on a line and integrating them along it they can predict pixels values which allows them to synthesize new images of the scene from unseen viewpoints.

While very different both those approaches have been modified to produce new SLAM systems: Droid-SLAM [Teed and Deng, 2021a] and iMap [Sucar et al., 2021]. Droid-SLAM uses the correlation volumes of features to iteratively estimate camera poses and depth images using an RNN. This approach works outstandingly well, it is precise and robust, even in challenging environments. However as the correlation volumes are built for all pairs of images in the covisibility graph they are very large and require a large amount of memory and computational resources for both training and inference. The second one, iMap uses a fully connected network as the only map representation, to optimize the required space. A NeRF is trained with keyframes and used to generate images corresponding to new frames. By minimizing the photometric error between generated images and captured frames the camera pose is estimated. This allows them to build in real-time and without training a dense and complete map of the environment. However this approach happens to be less precise than classical approaches such as ORB-SLAM 2 [Mur-Artal and Tardós, 2017]. We believe that both those approaches are promising for future research. Thus we propose the following research directions, in line with our work:

- Modifying DroidSLAM to enable camera pose estimation in dynamic scenes and object tracking. Indeed, as DroidSLAM is not suited for dynamic scene, the accuracy of camera pose estimation can suffer from moving objects. Furthermore it is not able to estimate the trajectory of moving objects. RAFT3D [Teed and Deng, 2021b] is a first interesting step towards this direction, able to estimate twists

fields from consecutive images. This approach opens many possibilities as it does not yet track objects over multiple frames and is not able to take into account semantic information to improve tracking. Furthermore it is particularly interesting for object tracking as points are extracted densely and processed with a CNN it may suffer less from small, far objects, from the problem of keypoints matching on moving objects or from the problem of being unable to extract enough keypoints for accurate tracking.

- Using deformable NerF approaches to represent the map in a SLAM system. Some recent approaches (e.g. NeRFies [K. Park, Sinha, Barron, et al., 2021], HyperNeRF [K. Park, Sinha, Hedman, et al., 2021]) can represent deformable scenes, some papers can also represent articulated objects [Tseng et al., 2022]. Thus they may be used as a changing map representation for SLAM systems based on implicit representation such as iMap. NiceSLAM [Z. Zhu et al., 2022] is a first step towards that direction but they chose to detect and ignore dynamic objects than to represent and track them.
- Using semantic information to improve the convergence of DroidSLAM and iMap. As we have shown in this manuscript, semantic information is valuable to improve the accuracy of mapping and tracking. A possibility could be for example to use semantic information to guide the depth estimated by the network to be coherent with geometric properties of objects. We can cite iLabel [Zhi et al., 2021] as a first step towards that direction that allows a NeRF based SLAM to segment a scene without any training. However this algorithm does not introduce prior information in the rendering process that could improve camera pose estimation.

Bibliography

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J., (2016), NetVLAD: CNN architecture for weakly supervised place recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5297–5307.

Arandjelović, R., & Zisserman, A., (2014), Visual vocabulary with a semantic twist, *Asian Conf. on Computer Vision*, 178–195.

Arndt, C., Sabzevari, R., & Civera, J., (2020), From points to planes - adding planar constraints to monocular SLAM factor graphs, *2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 4917–4922.

Arun, K. S., Huang, T. S., & Blostein, S. D., (1987), Least-squares fitting of two 3-D point sets, *IEEE Trans. on PAMI*, (5), 698–700.

Atanasov, N., Zhu, M., Daniilidis, K., & Pappas, G. J., (2014), Semantic localization via the matrix permanent., *Robotics: Science and Systems*, **2**.

Axler, S., (2014), *Linear algebra done right*, Springer.

Aygun, M., Osep, A., Weber, M., Maximov, M., Stachniss, C., Behley, J., & Leal-Taixé, L., (2021), 4d panoptic LiDAR segmentation, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5527–5537.

Ballester, I., Fontán, A., Civera, J., Strobl, K. H., & Triebel, R., (2021), DOT: dynamic object tracking for visual SLAM, *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 11705–11711.

Barath, D., Matas, J., & Noskova, J., (2019), MAGSAC: marginalizing sample consensus, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10197–10205.

Barath, D., & Matas, J., (2018), Graph-cut RANSAC, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6733–6741.

Barath, D., Noskova, J., Ivashechkin, M., & Matas, J., (2020), MAGSAC++, a fast, reliable and accurate robust estimator, *Conference on Computer Vision and Pattern Recognition*.

Bârsan, I. A., Liu, P., Pollefeys, M., & Geiger, A., (2018), Robust dense mapping for large-scale dynamic environments, *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 7510–7517.

Bay, H., Tuytelaars, T., & Van Gool, L., (2006), SURF: speeded up robust features, *European conference on computer vision*, 404–417.

Behley, J., & Stachniss, C., (2018), Efficient surfel-based SLAM using 3d laser range data in urban environments., *Robotics: Science and Systems*, **2018**, 59.

Bescos, B., Campos, C., Tardós, J. D., & Neira, J., (2021), DynaSLAM II: tightly-coupled multi-object tracking and SLAM, *IEEE Robotics and Automation Letters*, **6**(3), 5191–5198.

Bescos, B., Fácil, J. M., Civera, J., & Neira, J., (2018), DynaSLAM: tracking, mapping, and inpainting in dynamic scenes, *IEEE Robotics and Automation Letters*, **3**(4), 4076–4083.

Besl, P. J., & McKay, N. D., (1992), Method for registration of 3-D shapes, *Sensor fusion IV: control paradigms and data structures*, **1611**, 586–606.

Blanco, J.-L., (2010), A tutorial on se (3) transformation parameterizations and on-manifold optimization, *University of Malaga, Tech. Rep*, **3**, 6.

Bogoslavskyi, I., & Stachniss, C., (2016), Fast range image-based segmentation of sparse 3d laser scans for online operation, *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 163–169.

Bolya, D., Zhou, C., Xiao, F., & Lee, Y. J., (2019), Yolact: real-time instance segmentation, *Proceedings of the IEEE/CVF international conference on computer vision*, 9157–9166.

Bolya, D., Zhou, C., Xiao, F., & Lee, Y. J., (2020), Yolact++: better real-time instance segmentation, *IEEE transactions on pattern analysis and machine intelligence*.

Bowman, S. L., Atanasov, N., Daniilidis, K., & Pappas, G. J., (2017), Probabilistic data association for semantic SLAM, *2017 IEEE international conference on robotics and automation (ICRA)*, 1722–1729.

Brasch, N., Bozic, A., Lallemand, J., & Tombari, F., (2018), Semantic monocular SLAM for highly dynamic environments, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 393–400.

C. Hertzberger, L. S., Udo Freze, (2008), A framework for sparse, non-linear least squares problems on manifolds, *Master's thesis, University of Bremen*.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., & Leonard, J. J., (2016), Past, present, and future of simultaneous localization and mapping: toward the robust-perception age, *IEEE Trans. on robotics*, **32**(6), 1309–1332.

Calonder, M., Lepetit, V., Strecha, C., & Fua, P., (2010), Brief: binary robust independent elementary features, *European conference on computer vision*, 778–792.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S., (2020), End-to-end object detection with transformers, *European Conference on Computer Vision*, 213–229.

Castaldo, F., Zamir, A., Angst, R., Palmieri, F., & Savarese, S., (2015), Semantic cross-view matching, *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 9–17.

Chaumette, F., & Hutchinson, S., (2006), Visual servo control, Part I: basic approaches, *IEEE Robotics and Automation Magazine*, **13**(4), 82–90.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L., (2017), Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, *IEEE Trans. on pattern analysis and machine intelligence*, **40**(4), 834–848.

Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H., (2017), Rethinking atrous convolution for semantic image segmentation, *arXiv preprint arXiv:1706.05587*.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H., (2018), Encoder-decoder with atrous separable convolution for semantic image segmentation, *Proceedings of the European conference on computer vision (ECCV)*, 801–818.

Chen, X., Li, S., Mersch, B., Wiesmann, L., Gall, J., Behley, J., & Stachniss, C., (2021), Moving object segmentation in 3d lidar data: a learning-based approach exploiting sequential data, *IEEE Robotics and Automation Letters*, **6**(4), 6529–6536.

Chen, X., Milioto, A., Palazzolo, E., Giguere, P., Behley, J., & Stachniss, C., (2019), Suma++: efficient lidar-based semantic SLAM, *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4530–4537.

Cheng, B., Misra, I., Schwing, A. G., Kirillov, A., & Girdhar, R., (2022), Masked-attention mask transformer for universal image segmentation, *Computer Vision Pattern Recognition*.

Cheng, B., Schwing, A., & Kirillov, A., (2021), Per-pixel classification is not all you need for semantic segmentation, *Advances in Neural Information Processing Systems*, **34**.

Cheng, Y., (1995), Mean shift, mode seeking, and clustering, *IEEE Trans. on PAMI*, **17**(8), 790–799.

Cioffi, G., Cieslewski, T., & Scaramuzza, D., (2022), Continuous-time vs. discrete-time vision-based slam: a comparative study, *IEEE Robotics and Automation Letters*, **7**(2), 2399–2406.

Civera, J., Gálvez-López, D., Riazuelo, L., Tardós, J. D., & Montiel, J. M. M., (2011), Towards semantic SLAM using a monocular camera, *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1277–1284.

Comaniciu, D., & Meer, P., (2002), Mean shift: a robust approach toward feature space analysis, *IEEE Trans. on PAMI*, **24**(5), 603–619.

Comport, A. I., Marchand, É., & Chaumette, F., (2007), Kinematic sets for real-time robust articulated object tracking, *Image and Vision Computing*, **25**(3), 374–391.

Concha, A., & Civera, J., (2015), Dpptam: dense piecewise planar tracking and mapping from a monocular sequence, *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5686–5693.

Concha, A., Hussain, M. W., Montano, L., & Civera, J., (2014), Manhattan and piecewise-planar constraints for dense monocular mapping., *Robotics: Science and systems*.

Contributors, M., (2020), MMDetection3D: OpenMMLab next-generation platform for general 3D object detection.

Cui, L., & Ma, C., (2019), Sof-slam: a semantic visual SLAM for dynamic environments, *IEEE Access*, **7**, 166528–166539.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., & Nießner, M., (2017), Scannet: richly-annotated 3d reconstructions of indoor scenes, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5828–5839.

Dai, A., & Nießner, M., (2018), 3dmv: joint 3d-multi-view prediction for 3d semantic scene segmentation, *Proceedings of the European Conference on Computer Vision (ECCV)*, 452–468.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O., (2007), MonoSLAM: real-time single camera SLAM, *IEEE transactions on pattern analysis and machine intelligence*, **29**(6), 1052–1067.

Dellaert, F., (2014), Visual SLAM tutorial: bundle adjustment, *CVPR'14 tutorial*.

Dellaert, F., Seitz, S. M., Thorpe, C. E., & Thrun, S., (2000), Structure from motion without correspondence, *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*, **2**, 557–564.

DeTone, D., Malisiewicz, T., & Rabinovich, A., (2018), Superpoint: self-supervised interest point detection and description, *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 224–236.

Do, T.-T., Cai, M., Pham, T., & Reid, I., (2018), Deep-6DPose: recovering 6D object pose from a single RGB image, *arXiv preprint arXiv:1802.10367*.

Doherty, K., Fourie, D., & Leonard, J., (2019), Multimodal semantic SLAM with probabilistic data association, *2019 international conference on robotics and automation (ICRA)*, 2419–2425.

Doherty, K. J., Baxter, D. P., Schneeweiss, E., & Leonard, J. J., (2020), Probabilistic data association via mixture models for robust semantic SLAM, *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 1098–1104.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al., (2020), An image is worth 16x16 words: transformers for image recognition at scale, *International Conference on Learning Representa- tions*.

Duong, N.-D., (2019), *Hybrid machine learning and geometric approaches for single rgb camera relocalization* [Doctoral dissertation, CentraleSupélec].

Engel, J., Koltun, V., & Cremers, D., (2017), Direct sparse odometry, *IEEE transactions on pattern analysis and machine intelligence*, **40**(3), 611–625.

Engel, J., Schöps, T., & Cremers, D., (2014), LSD-SLAM: large-scale direct monocular SLAM, *European Conf. on computer vision*, 834–849.

Engels, C., Stewénius, H., & Nistér, D., (2006), Bundle adjustment rules, *Photogrammetric computer vision*, **2**(32).

Fan, Y., Zhang, Q., Liu, S., Tang, Y., Jing, X., Yao, J., & Han, H., (2020), Semantic SLAM with more accurate point cloud map in dynamic environments, *IEEE Access*, **8**, 112237–112252.

Fanelli, G., Gall, J., & Van Gool, L., (2011), Real time head pose estimation with random regression forests, *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 617–624.

Fioraio, N., & Di Stefano, L., (2013), Joint detection, tracking and mapping by semantic bundle adjustment, *IEEE Conf. on Computer Vision and Pattern Recognition*, 1538–1545.

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

Fitzgibbon, A. W., (2003), Robust registration of 2D and 3D point sets, *Image and vision computing*, **21**(13-14), 1145–1153.

Forster, C., Pizzoli, M., & Scaramuzza, D., (2014), Svo: fast semi-direct monocular visual odometry, *2014 IEEE international conference on robotics and automation (ICRA)*, 15–22.

Forsyth, D., & Ponce, J., (2011), *Computer vision: a modern approach.*, Prentice hall.

Frey, K. M., Steiner, T. J., & How, J. P., (2019), Efficient constellation-based map-merging for semantic SLAM, *2019 International Conference on Robotics and Automation (ICRA)*, 1302–1308.

Frost, D., Prisacariu, V., & Murray, D., (2018), Recovering stable scale in monocular SLAM using object-supplemented bundle adjustment, *IEEE Transactions on Robotics*, **34**(3), 736–747.

Frost, D. P., Kähler, O., & Murray, D. W., (2016), Object-aware bundle adjustment for correcting monocular scale drift, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 4770–4776.

Gall, J., Yao, A., Razavi, N., Van Gool, L., & Lempitsky, V., (2011), Hough forests for object detection, tracking, and action recognition, *IEEE Trans. on PAMI*, **33**(11), 2188–2202.

Gálvez-López, D., Salas, M., Tardós, J. D., & Montiel, J., (2016), Real-time monocular object SLAM, *Robotics and Autonomous Systems*, **75**, 435–449.

Gálvez-López, D., & Tardos, J. D., (2012), Bags of binary words for fast place recognition in image sequences, *IEEE Trans. on Robotics*, **28**(5), 1188–1197.

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P., & Garcia-Rodriguez, J., (2018), A survey on deep learning techniques for image and video semantic segmentation, *Applied Soft Computing*, **70**, 41–65.

Garg, S., Jacobson, A., Kumar, S., & Milford, M., (2017), Improving condition-and environment-invariant place recognition with semantic place categorization, *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6863–6870.

Garg, S., Suenderhauf, N., & Milford, M., (2018), Lost? appearance-invariant place recognition for opposite viewpoints using visual semantics, *Robotics: Science and Systems XIV*, 1–10.

Garg, S., Suenderhauf, N., & Milford, M., (2019), Semantic–geometric visual place recognition: a new perspective for reconciling opposing views, *The International Journal of Robotics Research*, 0278364919839761.

Gaudillière, V., Simon, G., & Berger, M.-O., (2019), Camera relocalization with ellipsoidal abstraction of objects, *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 8–18.

Gaudillière, V., Simon, G., & Berger, M.-O., (2020), Perspective-2-ellipsoid: bridging the gap between object detections and 6-dof camera pose, *IEEE Robotics and Automation Letters*, **5**(4), 5189–5196.

Gawel, A., Del Don, C., Siegwart, R., Nieto, J., & Cadena, C., (2018), X-view: graph-based semantic multi-view localization, *IEEE Robotics and Automation Letters*, **3**(3), 1687–1694.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R., (2013), Vision meets robotics: the kitti dataset, *The Int.Journal of Robotics Research*, **32**(11), 1231–1237.

Geiger, A., Lenz, P., & Urtasun, R., (2012), Are we ready for autonomous driving? the kitti vision benchmark suite, *2012 IEEE Conf. on computer vision and pattern recognition*, 3354–3361.

Geyer, C., & Daniilidis, K., (2000), A unifying theory for central panoramic systems and practical implications, *European conference on computer vision*, 445–461.

Girshick, R., (2015), Fast r-cnn, *Proceedings of the IEEE international conference on computer vision*, 1440–1448.

Girshick, R., Donahue, J., Darrell, T., & Malik, J., (2015), Region-based convolutional networks for accurate object detection and segmentation, *IEEE transactions on pattern analysis and machine intelligence*, **38**(1), 142–158.

Gonzalez, M., Kacete, A., Murienne, A., & Marchand, E., (2021), L6dnet: light 6 DoF network for robust and precise object pose estimation with small datasets, *IEEE Robotics and Automation Letters*, **6**(2), 2914–2921.

Gonzalez, M., Marchand, E., Kacete, A., & Royan, J., (2021), S^3LAM: structured scene SLAM, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'22*.

Gonzalez, M., Marchand, E., Kacete, A., & Royan, J., (2022a), S3LAM: SLAM à scène structurée, *GRETSI'22-XXVIIIème Colloque Francophone de Traitement du Signal et des Images*.

Gonzalez, M., Marchand, E., Kacete, A., & Royan, J., (2022b), TwistSLAM: constrained SLAM in dynamic environment, *IEEE Robotics and Automation Letters*, **7**(3), 6846–6853.

Gonzalez, M., Marchand, E., Kacete, A., & Royan, J., (2022c), TwistSLAM++: using multiple modalities for accurate dynamic semantic SLAM, *arXiv preprint arXiv:2209.07888*.

Grabner, A., Roth, P. M., & Lepetit, V., (2018), 3d pose estimation and 3d model retrieval for objects in the wild, *IEEE Conf. on Computer Vision and Pattern Recognition*, 3022–3031.

Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W., (2010), A tutorial on graph-based SLAM, *IEEE Intelligent Transportation Systems Magazine*, **2**(4), 31–43.

Grisetti, G., Kümmerle, R., Strasdat, H., & Konolige, K., (2011), G2o: a general framework for (hyper) graph optimization, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 9–13.

Guo, Y., Liu, Y., Georgiou, T., & Lew, M. S., (2018), A review of semantic segmentation using deep neural networks, *International journal of multimedia information retrieval*, **7**(2), 87–93.

Hachiuma, R., Pirchheim, C., Schmalstieg, D., & Saito, H., (2020), Detectfusion: detecting and segmenting both known and unknown dynamic objects in real-time SLAM, *30th British Machine Vision Conference, BMVC 2019*.

Han, S., & Xi, Z., (2020), Dynamic scene semantics SLAM based on semantic segmentation, *IEEE Access*, **8**, 43563–43570.

Harris, C., Stephens, M., et al., (1988), A combined corner and edge detector, *Alvey vision conference*, **15**(50), 10–5244.

Hartley, R., & Zisserman, A., (2003), *Multiple view geometry in computer vision*, Cambridge university press.

Hartley, R. I., & Sturm, P., (1997), Triangulation, *Computer vision and image understanding*, **68**(2), 146–157.

He, K., Gkioxari, G., Dollár, P., & Girshick, R., (2017), Mask R-CNN, *IEEE Int. Conf. on computer vision*, 2961–2969.

He, K., Zhang, X., Ren, S., & Sun, J., (2016), Identity mappings in deep residual networks, *European conference on computer vision*, 630–645.

He, Y., Sun, W., Huang, H., Liu, J., Fan, H., & Sun, J., (2020), Pvn3d: a deep point-wise 3d keypoints voting network for 6dof pose estimation, *IEEE Conf. on Computer Vision and Pattern Recognition*, 11632–11641.

Henein, M., Zhang, J., Mahony, R., & Ila, V., (2020), Dynamic SLAM: the need for speed, *2020 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2123–2129.

Henning, D., Laidlow, T., & Leutenegger, S., (2022), BodySLAM: joint camera localisation, mapping, and human motion tracking, *arXiv preprint arXiv:2205.02301*.

Hermans, A., Floros, G., & Leibe, B., (2014), Dense 3d semantic mapping of indoor scenes from RGB-D images, *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2631–2638.

Himri, K., Ridao, P., Gracias, N., Palomer, A., Palomeras, N., & Pi, R., (2018), Semantic SLAM for an auv using object recognition from point clouds, *IFAC-PapersOnLine*, **51**(29), 360–365.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., & Navab, N., (2012), Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes, *Asian Conf. on Computer Vision*, 548–562.

Hong, W., Guo, Q., Zhang, W., Chen, J., & Chu, W., (2021), Lpsnet: a lightweight solution for fast panoptic segmentation, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16746–16754.

Horn, B. K., (1987), Closed-form solution of absolute orientation using unit quaternions, *Josa a*, **4**(4), 629–642.

Hosseinzadeh, M., Latif, Y., Pham, T., Suenderhauf, N., & Reid, I., (2018), Structure aware SLAM using quadrics and planes, *Asian Conf. on Computer Vision*, 410–426.

Hosseinzadeh, M., Li, K., Latif, Y., & Reid, I., (2019), Real-time monocular object-model aware sparse SLAM, *2019 Int. Conf. on Robotics and Automation (ICRA)*, 7123–7129.

Hou, B., Miolane, N., Khanal, B., Lee, M. C., Alansary, A., McDonagh, S., Hajnal, J. V., Rueckert, D., Glocker, B., & Kainz, B., (2018), Computing cnn loss and gradients for pose estimation with riemannian geometry, *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, 756–764.

Hou, Y., Zhang, H., Zhou, S., & Zou, H., (2017), Use of roadway scene semantic information and geometry-preserving landmark pairs to improve visual place recognition in changing environments, *IEEE Access*, **5**, 7702–7713.

Hsiao, M., Westman, E., Zhang, G., & Kaess, M., (2017), Keyframe-based dense planar SLAM, *2017 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 5110–5117.

Huang, J., Yang, S., Mu, T.-J., & Hu, S.-M., (2020), ClusterVO: clustering moving instances and estimating visual odometry for self and surroundings, *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2168–2177.

Huang, J., Yang, S., Zhao, Z., Lai, Y.-K., & Hu, S.-M., (2019), ClusterSLAM: a SLAM backend for simultaneous rigid body clustering and motion estimation, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5875–5884.

Huber, P. J., (2011), Robust statistics. *In International encyclopedia of statistical science* (pp. 1248–1251), Springer.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al., (2011), Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera, *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 559–568.

Jégou, H., Douze, M., Schmid, C., & Pérez, P., (2010), Aggregating local descriptors into a compact image representation, *2010 IEEE computer society conference on computer vision and pattern recognition*, 3304–3311.

Jeong, J., Yoon, T. S., & Park, J. B., (2018), Multimodal sensor-based semantic 3d mapping for a large-scale environment, *Expert Systems with Applications*, **105**, 1–10.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., & Qu, R., (2019), A survey of deep learning-based object detection, *IEEE access*, **7**, 128837–128868.

Judd, K. M., & Gammell, J. D., (2019), The oxford multimotion dataset: multiple SE(3) motions with ground truth, *IEEE Robotics and Automation Letters*, **4**(2), 800–807.

Kacete, A., Royan, J., Seguier, R., Collobert, M., & Soladie, C., (2016), Real-time eye pupil localization using Hough regression forest, *IEEE Winter Conf. on Applications of Computer Vision*.

Kaess, M., (2015), Simultaneous localization and mapping with infinite planes, *2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 4605–4611.

Kaneko, M., Iwami, K., Ogawa, T., Yamasaki, T., & Aizawa, K., (2018), Mask-slam: robust feature-based monocular SLAM by masking using semantic segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 258–266.

Kehl, W., Manhardt, F., Tombari, F., Ilic, S., & Navab, N., (2017), SSD-6D: making RGB-based 3D detection and 6D pose estimation great again, *IEEE Int. Conf. on Computer Vision*, 1521–1529.

Kehl, W., Milletari, F., Tombari, F., Ilic, S., & Navab, N., (2016), Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation, *European Conf. on Computer Vision*, 205–220.

Kendall, A., Grimes, M., & Cipolla, R., (2015), Posenet: a convolutional network for real-time 6-dof camera relocalization, *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Kirillov, A., Girshick, R., He, K., & Dollár, P., (2019a), Panoptic feature pyramid networks, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6399–6408.

Kirillov, A., Girshick, R., He, K., & Dollár, P., (2019b), Panoptic feature pyramid networks, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6399–6408.

Klein, G., & Murray, D., (2007), Parallel tracking and mapping for small AR workspaces, *2007 6th IEEE and ACM int. symposium on mixed and augmented reality*, 225–234.

Knorr, S. B., & Kurz, D., (2016), Leveraging the user's face for absolute scale estimation in handheld monocular SLAM, *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 11–17.

Kobyshev, N., Riemenschneider, H., & Van Gool, L., (2014), Matching features correctly through semantic understanding, *2014 2nd Int. Conf. on 3D Vision*, **1**, 472–479.

Krizhevsky, A., Sutskever, I., & Hinton, G. E., (2012), Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, **25**.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W., (2011), G2o: a general framework for graph optimization, *2011 IEEE Int. Conf. on Robotics and Automation*, 3607–3613.

Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., & Kleiner, A., (2009), On measuring the accuracy of SLAM algorithms, *Autonomous Robots*, **27**(4), 387–407.

Kundu, A., Li, Y., Dellaert, F., Li, F., & Rehg, J. M., (2014), Joint semantic segmentation and 3d reconstruction from monocular video, *European Conference on Computer Vision*, 703–718.

Landrieu, L., & Simonovsky, M., (2018), Large-scale point cloud semantic segmentation with superpoint graphs, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4558–4567.

Larsson, M., Stenborg, E., Hammarstrand, L., Pollefeys, M., Sattler, T., & Kahl, F., (2019), A cross-season correspondence dataset for robust semantic segmentation,

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9532–9542.

Larsson, M., Stenborg, E., Toft, C., Hammarstrand, L., Sattler, T., & Kahl, F., (2019), Fine-grained segmentation networks: self-supervised segmentation for improved long-term visual localization, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 31–41.

Li, J., Meger, D., & Dudek, G., (2019), Semantic mapping for view-invariant relocalization, *2019 International Conference on Robotics and Automation (ICRA)*, 7108–7115.

Li, P., Qin, T., et al., (2018), Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving, *Proceedings of the European Conf. on Computer Vision (ECCV)*, 646–661.

Li, S., & Lee, D., (2017), RGB-D SLAM in dynamic environments using static point weighting, *IEEE Robotics and Automation Letters*, **2**(4), 2263–2270.

Li, X., & Belaroussi, R., (2016), Semi-dense 3D semantic mapping from monocular SLAM, *arXiv preprint arXiv:1611.04144*.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B., (2018), Pointcnn: convolution on x-transformed points, *Advances in neural information processing systems*, **31**, 820–830.

Li, Y., Zhao, H., Qi, X., Wang, L., Li, Z., Sun, J., & Jia, J., (2021), Fully convolutional networks for panoptic segmentation, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 214–223.

Li, Y., Wang, G., Ji, X., Xiang, Y., & Fox, D., (2018), DeepIM: Deep Iterative Matching for 6D pose estimation, *European Conf. on Computer Vision*, 683–698.

Li, Z., Wang, G., & Ji, X., (2019), CDPN: Coordinates-Based Disentangled Pose Network for real-time RGB-Based 6-DoF object pose estimation, *IEEE Int. Conf. on Computer Vision*, 7678–7687.

Lianos, K.-N., Schonberger, J. L., Pollefeys, M., & Sattler, T., (2018), Vso: visual semantic odometry, *Proceedings of the European Conference on Computer Vision (ECCV)*, 234–250.

Liao, Z., Shi, J., Qi, X., Zhang, X., Wang, W., He, Y., Liu, X., & Wei, R., (2020), Coarse-to-fine visual localization using semantic compact map, *2020 3rd International Conference on Control and Robots (ICCR)*, 30–37.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P., (2017), Focal loss for dense object detection, *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C., (2016), Ssd: single shot multibox detector, *European conference on computer vision*, 21–37.

Liu, W., Sun, J., Li, W., Hu, T., & Wang, P., (2019), Deep learning on point clouds and its application: a survey, *Sensors*, **19**(19), 4188.

Liu, Y., & Miura, J., (2021), Rds-slam: real-time dynamic SLAM using semantic segmentation methods, *IEEE Access*, **9**, 23772–23785.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B., (2021), Swin transformer: hierarchical vision transformer using shifted windows, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S., (2022), A convnet for the 2020s, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11976–11986.

Long, J., Shelhamer, E., & Darrell, T., (2015), Fully convolutional networks for semantic segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

Lowe, D. G., (1999), Object recognition from local scale-invariant features, *Proceedings of the seventh IEEE international Conference on Computer Vision*, **2**, 1150–1157.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., & Milford, M. J., (2015), Visual place recognition: a survey, *IEEE Transactions on Robotics*, **32**(1), 1–19.

Luiten, J., Fischer, T., & Leibe, B., (2020), Track to reconstruct and reconstruct to track, *IEEE Robotics and Automation Letters*, **5**(2), 1803–1810.

Ma, J., Jiang, X., Fan, A., Jiang, J., & Yan, J., (2021), Image matching from handcrafted to deep features: a survey, *International Journal of Computer Vision*, **129**(1), 23–79.

Ma, W.-C., Tartavull, I., Bârsan, I. A., Wang, S., Bai, M., Mattyus, G., Homayounfar, N., Lakshmikanth, S. K., Pokrovsky, A., & Urtasun, R., (2019), Exploiting sparse semantic hd maps for self-driving vehicle localization, *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5304–5311.

Ma, W.-C., Wang, S., Brubaker, M. A., Fidler, S., & Urtasun, R., (2017), Find your way by observing the sun and other semantic cues, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 6292–6299.

Ma, Y., Soatto, S., Košecká, J., & Sastry, S., (2004), *An invitation to 3-D vision: from images to geometric models* (Vol. 26), Springer.

Mahendran, S., Ali, H., & Vidal, R., (2017), 3D pose regression using convolutional neural networks, *IEEE Int. Conf. on Computer Vision*, 2174–2182.

Malis, E., & Marchand, E., (2006), Experiments with robust estimation techniques in real-time robot vision, *2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 223–228.

Marchand, E., Uchiyama, H., & Spindler, F., (2016), Pose estimation for augmented reality: a hands-on survey, *IEEE Trans. on Visualization and Computer Graphics*, **22**(12), 2633–2651.

Martıén Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, ... Xiaoqiang Zheng, (2015), TensorFlow: large-scale machine learning on heterogeneous systems [Software available from tensorflow.org], http://tensorflow.org/

Masone, C., & Caputo, B., (2021), A survey on deep visual place recognition, *IEEE Access*, **9**, 19516–19547.

McCormac, J., Clark, R., Bloesch, M., Davison, A., & Leutenegger, S., (2018), Fusion++: volumetric object-level SLAM, *2018 international conference on 3D vision (3DV)*, 32–41.

McCormac, J., Handa, A., Davison, A., & Leutenegger, S., (2017), Semanticfusion: dense 3D semantic mapping with convolutional neural networks, *2017 IEEE Int. Conf. on Robotics and automation (ICRA)*, 4628–4635.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R., (2020), Nerf: representing scenes as neural radiance fields for view synthesis, *European conference on computer vision*, 405–421.

Milioto, A., Behley, J., McCool, C., & Stachniss, C., (2020), Lidar panoptic segmentation for autonomous driving, *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8505–8512.

Milioto, A., Vizzo, I., Behley, J., & Stachniss, C., (2019), Rangenet++: fast and accurate lidar semantic segmentation, *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 4213–4220.

Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D., (2021), Image segmentation using deep learning: a survey, *IEEE transactions on pattern analysis and machine intelligence*.

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., & Sayd, P., (2006), Real time localization and 3D reconstruction, *IEEE Conf. on Computer Vision and Pattern Recognition*, **1**, 363–370.

Mousavian, A., Košecká, J., & Lien, J.-M., (2015), Semantically guided location recognition for outdoors scenes, *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 4882–4889.

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D., (2015), ORB-SLAM: a versatile and accurate monocular SLAM system, *IEEE Trans. on robotics*, **31**(5), 1147–1163.

Mur-Artal, R., & Tardós, J. D., (2017), ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras, *IEEE Trans. on Robotics*, **33**(5), 1255–1262.

Naseer, T., Oliveira, G. L., Brox, T., & Burgard, W., (2017), Semantics-aware visual localization under challenging perceptual conditions, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2614–2620.

Nicholson, L., Milford, M., & Sünderhauf, N., (2018), QuadricSLAM: dual quadrics from object detections as landmarks in object-oriented SLAM, *IEEE Robotics and Automation Letters*, **4**(1), 1–8.

Nocedal, J., & Wright, S., (2006), *Numerical optimization*, Springer Science & Business Media.

Nocedal, J., & Wright, S. J., (1999), *Numerical optimization*, Springer.

Ok, K., Liu, K., Frey, K., How, J. P., & Roy, N., (2019), Robust object-based SLAM for high-speed autonomous navigation, *2019 International Conference on Robotics and Automation (ICRA)*, 669–675.

Park, C., Moghadam, P., Kim, S., Elfes, A., Fookes, C., & Sridharan, S., (2018), Elastic lidar fusion: dense map-centric continuous-time SLAM, *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 1206–1213.

Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S., (2019), Deepsdf: learning continuous signed distance functions for shape representation, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 165–174.

Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., & Martin-Brualla, R., (2021), Nerfies: deformable neural radiance fields, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5865–5874.

Park, K., Sinha, U., Hedman, P., Barron, J. T., Bouaziz, S., Goldman, D. B., Martin-Brualla, R., & Seitz, S. M., (2021), Hypernerf: a higher-dimensional representation for topologically varying neural radiance fields, *arXiv preprint arXiv:2106.13228*.

Park, K., Patten, T., & Vincze, M., (2019), Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation, *IEEE Int. Conf. on Computer Vision*, 7668–7677.

Pavlakos, G., Zhou, X., Chan, A., Derpanis, K. G., & Daniilidis, K., (2017), 6-DoF object pose from semantic keypoints, *IEEE Int. Conf. on Robotics and Automation*, 2011–2018.

Peng, J., Shi, X., Wu, J., & Xiong, Z., (2019), An object-oriented semantic SLAM system towards dynamic environments for mobile manipulation, *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 199–204.

Peng, S., Liu, Y., Huang, Q., Zhou, X., & Bao, H., (2019), PVNet: Pixel-wise Voting Network for 6DoF pose estimation, *IEEE Conf. on Computer Vision and Pattern Recognition*, 4561–4570.

Pham, Q.-H., Hua, B.-S., Nguyen, T., & Yeung, S.-K., (2019), Real-time progressive 3d semantic segmentation for indoor scenes, *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1089–1098.

Pham, Q.-H., Nguyen, T., Hua, B.-S., Roig, G., & Yeung, S.-K., (2019), Jsis3d: joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8827–8836.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., & Zuiderveld, K., (1987), Adaptive histogram equalization and its variations, *Computer vision, graphics, and image processing*, **39**(3), 355–368.

Pizer, S. M., Johnston, R. E., Ericksen, J. P., Yankaskas, B. C., & Muller, K. E., (1990), Contrast-limited adaptive histogram equalization: speed and effectiveness, *[1990]*

Proceedings of the First Conference on Visualization in Biomedical Computing, 337–338.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J., (2017), Pointnet: deep learning on point sets for 3d classification and segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J., (2017), Pointnet++: deep hierarchical feature learning on point sets in a metric space, *Advances in neural information processing systems*, **30**.

Rad, M., & Lepetit, V., (2017), BB8: a scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth, *IEEE Int. Conf. on Computer Vision*, 3828–3836.

Radwan, N., Valada, A., & Burgard, W., (2018), Vlocnet++: deep multitask learning for semantic visual localization and odometry, *IEEE Robotics and Automation Letters*, **3**(4), 4407–4414.

Ramadasan, D., Chevaldonné, M., Chateau, T., & Clermont-Ferrand, F., (2015), Mcslam: a multiple constrained slam., *BMVC*, 107–1.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A., (2016), You only look once: unified, real-time object detection, *IEEE Conf. on computer vision and pattern recognition*, 779–788.

Redmon, J., & Farhadi, A., (2018), Yolov3: an incremental improvement, *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., & Sun, J., (2015), Faster R-CNN: towards real-time object detection with region proposal networks, *Advances in neural information processing systems*, **28**.

Riegler, G., Ferstl, D., Rüther, M., & Bischof, H., (2013), Hough networks for head pose estimation and facial feature localization, *Journal of Computer Vision*, **101**(3), 437–458.

Ronneberger, O., Fischer, P., & Brox, T., (2015), U-net: convolutional networks for biomedical image segmentation, *International Conference on Medical image computing and computer-assisted intervention*, 234–241.

Rosinol, A., Abate, M., Chang, Y., & Carlone, L., (2020), Kimera: an open-source library for real-time metric-semantic localization and mapping, *IEEE Int. Conf. on Robotics and Automation*, 1689–1696.

Rosten, E., & Drummond, T., (2006), Machine learning for high-speed corner detection, *European conference on computer vision*, 430–443.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G., (2011), ORB: an efficient alternative to SIFT or SURF, *2011 Int. Conf. on computer vision*, 2564–2571.

Runz, M., Buffier, M., & Agapito, L., (2018), Maskfusion: real-time recognition, tracking and reconstruction of multiple moving objects, *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 10–20.

Runz, M., Li, K., Tang, M., Ma, L., Kong, C., Schmidt, T., Reid, I., Agapito, L., Straub, J., Lovegrove, S., et al., (2020), Frodo: from detections to 3d objects, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14720–14729.

Rünz, M., & Agapito, L., (2017), Co-fusion: real-time segmentation, tracking and fusion of multiple objects, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 4471–4478.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al., (2015), Imagenet large scale visual recognition challenge, *International journal of computer vision*, **115**(3), 211–252.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., & Davison, A. J., (2013), SLAM++: simultaneous localisation and mapping at the level of objects, *IEEE Conf. on computer vision and pattern recognition*, 1352–1359.

Schonberger, J. L., & Frahm, J.-M., (2016), Structure-from-motion revisited, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4104–4113.

Schönberger, J. L., Pollefeys, M., Geiger, A., & Sattler, T., (2018), Semantic visual localization, *IEEE Conf. on Computer Vision and Pattern Recognition*, 6896–6906.

Schorghuber, M., Steininger, D., Cabon, Y., Humenberger, M., & Gelautz, M., (2019), SLAMANTIC-leveraging semantics to improve VSLAM in dynamic environments, *IEEE/CVF Int. Conf. on Computer Vision Workshops*.

Scona, R., Jaimez, M., Petillot, Y. R., Fallon, M., & Cremers, D., (2018), Staticfusion: background reconstruction for dense RGB-D SLAM in dynamic environments, *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 3849–3856.

Segal, A., Haehnel, D., & Thrun, S., (2009), Generalized-icp., *Robotics: science and systems*, **2**(4), 435.

Sengupta, S., Greveson, E., Shahrokni, A., & Torr, P. H., (2013), Urban 3d semantic modelling using stereo vision, *2013 IEEE International Conference on robotics and Automation*, 580–585.

Seymour, Z., Sikka, K., Chiu, H.-P., Samarasekera, S., & Kumar, R., (2019), Semantically-aware attentive neural embeddings for long-term 2d visual localization, *British Machine Vision Conference (BMVC)*.

Shi, J., et al., (1994), Good features to track, *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, 593–600.

Shi, T., Shen, S., Gao, X., & Zhu, L., (2019), Visual localization using sparse semantic 3d map, *2019 IEEE international conference on image processing (ICIP)*, 315–319.

Siam, M., Elkerdawy, S., Jagersand, M., & Yogamani, S., (2017), Deep semantic segmentation for automated driving: taxonomy, roadmap and challenges, *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, 1–8.

Siddiqui, Y., Thies, J., Ma, F., Shan, Q., Nießner, M., & Dai, A., (2022), Texturify: generating textures on 3d shape surfaces, *arXiv preprint arXiv:2204.02411*.

Simonyan, K., & Zisserman, A., (2014), Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556*.

Singh, G., & Košecká, J., (2016), Semantically guided geo-location and modeling in urban environments. *In Large-scale visual geo-localization* (pp. 101–120), Springer.

Sivic, J., & Zisserman, A., (2003), Video google: a text retrieval approach to object matching in videos, *Computer Vision, IEEE International Conference on*, **3**, 1470–1470.

Stenborg, E., Toft, C., & Hammarstrand, L., (2018), Long-term visual localization using semantically segmented images, *2018 IEEE international Conference on robotics and automation (ICRA)*, 6484–6490.

Strasdat, H., Montiel, J. M., & Davison, A. J., (2012), Visual slam: why filter?, *Image and Vision Computing*, **30**(2), 65–77.

Strecke, M., & Stuckler, J., (2019), Em-fusion: dynamic object-level SLAM with probabilistic data association, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5865–5874.

Stückler, J., & Behnke, S., (2014), Multi-resolution surfel maps for efficient dense 3d modeling and tracking, *Journal of Visual Communication and Image Representation*, **25**(1), 137–147.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D., (2012), A benchmark for the evaluation of RGB-D SLAM systems, *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*.

Sturm, J., Jain, A., Stachniss, C., Kemp, C. C., & Burgard, W., (2010), Operating articulated objects based on experience, *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2739–2744.

Sturm, J., Stachniss, C., & Burgard, W., (2011), A probabilistic framework for learning kinematic models of articulated objects, *Journal of Artificial Intelligence Research*, **41**, 477–526.

Sucar, E., & Hayet, J.-B., (2017), Probabilistic global scale estimation for monoSLAM based on generic object detection, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 48–56.

Sucar, E., & Hayet, J.-B., (2018), Bayesian scale estimation for monocular SLAM based on generic object detection for correcting scale drift, *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 5152–5158.

Sucar, E., Liu, S., Ortiz, J., & Davison, A. J., (2021), Imap: implicit mapping and positioning in real-time, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6229–6238.

Sucar, E., Wada, K., & Davison, A., (2020), Nodeslam: neural object descriptors for multi-view shape reconstruction, *2020 International Conference on 3D Vision (3DV)*, 949–958.

Sun, L., Yan, Z., Zaganidis, A., Zhao, C., & Duckett, T., (2018), Recurrent-octomap: learning state-based map refinement for long-term semantic mapping with 3-d-lidar data, *IEEE Robotics and Automation Letters*, **3**(4), 3749–3756.

Sun, Y., Liu, M., & Meng, M. Q.-H., (2017), Improving RGB-D SLAM in dynamic environments: a motion removal approach, *Robotics and Autonomous Systems*, **89**, 110–122.

Sünderhauf, N., & Milford, M., (2017), Dual quadrics from object detection bounding-boxes as landmark representations in SLAM, *arXiv preprint arXiv:1708.00965*.

Sünderhauf, N., Pham, T. T., Latif, Y., Milford, M., & Reid, I., (2017), Meaningful maps with object-oriented semantic mapping, *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 5079–5085.

Sundermeyer, M., Marton, Z.-C., Durner, M., Brucker, M., & Triebel, R., (2018), Implicit 3D orientation learning for 6D object detection from RGB images, *European Conf. on Computer Vision*, 699–715.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A., (2015), Going deeper with convolutions, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Szeliski, R., (2010), *Computer vision: algorithms and applications*, Springer Science & Business Media.

Taira, H., Okutomi, M., Sattler, T., Cimpoi, M., Pollefeys, M., Sivic, J., Pajdla, T., & Torii, A., (2018), Inloc: indoor visual localization with dense matching and view synthesis, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7199–7209.

Taira, H., Rocco, I., Sedlar, J., Okutomi, M., Sivic, J., Pajdla, T., Sattler, T., & Torii, A., (2019), Is this the right place? geometric-semantic pose verification for indoor visual localization, *IEEE/CVF Int. Conf. on Computer Vision*, 4373–4383.

Tan, M., & Le, Q., (2019), Efficientnet: rethinking model scaling for convolutional neural networks, *International conference on machine learning*, 6105–6114.

Tang, J., Ericson, L., Folkesson, J., & Jensfelt, P., (2019), Gcnv2: efficient correspondence prediction for real-time SLAM, *IEEE Robotics and Automation Letters*, **4**(4), 3505–3512.

Tang, J., Folkesson, J., & Jensfelt, P., (2018), Geometric correspondence network for camera motion estimation, *IEEE Robotics and Automation Letters*, **3**(2), 1010–1017.

Tatarchenko, M., Park, J., Koltun, V., & Zhou, Q.-Y., (2018), Tangent convolutions for dense prediction in 3d, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3887–3896.

Tateno, K., Tombari, F., Laina, I., & Navab, N., (2017), CNN-SLAM: real-time dense monocular SLAM with learned depth prediction, *IEEE Conf. on Computer Vision and Pattern Recognition*, 6243–6252.

Teed, Z., & Deng, J., (2020), RAFT: recurrent all-pairs field transforms for optical flow, *European Conf. on computer vision*, 402–419.

Teed, Z., & Deng, J., (2021a), Droid-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras, *Advances in Neural Information Processing Systems*, **34**, 16558–16569.

Teed, Z., & Deng, J., (2021b), RAFT-3D: scene flow using rigid-motion embeddings, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8375–8384.

Tejani, A., Tang, D., Kouskouridas, R., & Kim, T.-K., (2014), Latent-class Hough forests for 3D object detection and pose estimation, *European Conf. on Computer Vision*, 462–477.

Tekin, B., Bogo, F., & Pollefeys, M., (2019), H+O: unified egocentric recognition of 3D hand-object poses and interactions, *IEEE Conf. on Computer Vision and Pattern Recognition*, 4511–4520.

Tekin, B., Sinha, S. N., & Fua, P., (2018), Real-time seamless single shot 6D object pose prediction, *IEEE Conf. on Computer Vision and Pattern Recognition*, 292–301.

Thoma, M., (2016), A survey of semantic segmentation, *arXiv preprint arXiv:1602.06541*.

Tirado, J., & Civera, J., (2022), Jacobian computation for cumulative b-splines on se (3) and application to continuous-time object tracking, *IEEE Robotics and Automation Letters*.

Toft, C., Maddern, W., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J., Pajdla, T., et al., (2020), Long-term visual localization revisited, *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Toft, C., Olsson, C., & Kahl, F., (2017), Long-term 3D localization and pose from semantic labellings, *IEEE Int. Conf. on Computer Vision Workshops*, 650–659.

Toft, C., Stenborg, E., Hammarstrand, L., Brynte, L., Pollefeys, M., Sattler, T., & Kahl, F., (2018), Semantic match consistency for long-term visual localization, *European Conf. on Computer Vision (ECCV)*, 383–399.

Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. W., (1999), Bundle adjustment—a modern synthesis, *Int. workshop on vision algorithms*, 298–372.

Tschopp, F., Nieto, J., Siegwart, R., & Cadena, C., (2021), Superquadric object representation for optimization-based semantic SLAM, *arXiv preprint arXiv:2109.09627*.

Tseng, W.-C., Liao, H.-J., Yen-Chen, L., & Sun, M., (2022), CLA-NeRF: category-level articulated neural radiance field, *arXiv preprint arXiv:2202.00181*.

Umeyama, S., (1991), Least-squares estimation of transformation parameters between two point patterns, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **13**(04), 376–380.

Valada, A., Radwan, N., & Burgard, W., (2018), Incorporating semantic and geometric priors in deep pose regression, *Workshop on learning and inference in robotics:*

Integrating structure, priors and models at robotics: Science and systems (RSS), **1**, 3.

Valentin, J. P., Sengupta, S., Warrell, J., Shahrokni, A., & Torr, P. H., (2013), Mesh based semantic modelling for indoor and outdoor scenes, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2067–2074.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I., (2017), Attention is all you need, *Advances in neural information processing systems*, **30**.

Vincent, J., Labbé, M., Lauzon, J.-S., Grondin, F., Comtois-Rivet, P.-M., & Michaud, F., (2020), Dynamic object tracking and masking for visual SLAM, *2020 IEEE/RSJ Int Conf. on Intelligent Robots and Systems (IROS)*, 4974–4979.

Vineet, V., Miksik, O., Lidegaard, M., Nießner, M., Golodetz, S., Prisacariu, V. A., Kähler, O., Murray, D. W., Izadi, S., Pérez, P., et al., (2015), Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction, *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 75–82.

Voodarla, M., Shrivastava, S., Manglani, S., Vora, A., Agarwal, S., & Chakravarty, P., (2021), S-bev: semantic birds-eye view representation for weather and lighting invariant 3-dof localization, *arXiv preprint arXiv:2101.09569*.

Wang, C., Xu, D., Zhu, Y., Martıén-Martıén, R., Lu, C., Fei-Fei, L., & Savarese, S., (2019), Densefusion: 6D object pose estimation by iterative dense fusion, *IEEE Conf. on Computer Vision and Pattern Recognition*, 3343–3352.

Wang, J., Rünz, M., & Agapito, L., (2021), DSP-SLAM: object oriented SLAM with deep shape priors, *2021 International Conference on 3D Vision (3DV)*, 1362–1371.

Wang, P., Yang, R., Cao, B., Xu, W., & Lin, Y., (2018), Dels-3d: deep localization and segmentation with a 3d semantic map, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5860–5869.

Wang, X., Christie, M., & Marchand, E., (2022), Binary graph descriptor for robust relocalization on heterogeneous data, *IEEE Robotics and Automation Letters*, **7**(2), 2008–2015.

Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., & Davison, A., (2015), ElasticFusion: dense SLAM without a pose graph, *Robotics: Science and Systems*.

Wilson, W., Hulls, C., & Bell, G., (1996), Relative end-effector control using cartesian position-based visual servoing, *IEEE Trans. on Robotics and Automation*, **12**(5), 684–696.

Wong, Y.-S., Li, C., Nießner, M., & Mitra, N. J., (2021), Rigidfusion: RGB-D scene reconstruction with rigidly-moving objects, *Computer Graphics Forum*, **40**, 511–522.

Wu, J., Xue, T., Lim, J. J., Tian, Y., Tenenbaum, J. B., Torralba, A., & Freeman, W. T., (2016), Single image 3d interpreter network, *European Conf. on Computer Vision*, 365–382.

Wu, W., Qi, Z., & Fuxin, L., (2019), Pointconv: deep convolutional networks on 3d point clouds, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9621–9630.

Wu, Y., Zhang, Y., Zhu, D., Feng, Y., Coleman, S., & Kerr, D., (2020), Eao-slam: monocular semi-dense object SLAM based on ensemble data association, *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4966–4973.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R., (2019), Detectron2.

Xiang, Y., & Fox, D., (2017), Da-rnn: semantic mapping with data associated recurrent neural networks, *Robotics: Science and Systems (RSS)*.

Xiang, Y., Schmidt, T., Narayanan, V., & Fox, D., (2018), Posecnn: a convolutional neural network for 6d object pose estimation in cluttered scenes, *Robotics: Science and Systems (RSS)*.

Xiao, L., Wang, J., Qiu, X., Rong, Z., & Zou, X., (2019), Dynamic-SLAM: semantic monocular visual localization and mapping based on deep learning in dynamic environment, *Robotics and Autonomous Systems*, **117**, 1–16.

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K., (2017), Aggregated residual transformations for deep neural networks, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Xu, B., Li, W., Tzoumanikas, D., Bloesch, M., Davison, A., & Leutenegger, S., (2019), Mid-fusion: octree-based object-level multi-instance dynamic SLAM, *2019 International Conference on Robotics and Automation (ICRA)*, 5231–5237.

Xu, H., Zhang, S., & Liu, P., (2019), Dos-slam: a real-time dynamic object segmentation visual SLAM system, *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, 85–90.

Yang, S., & Scherer, S., (2019a), CubeSLAM: monocular 3-D object SLAM, *IEEE Trans. on Robotics*, **35**(4), 925–938.

Yang, S., & Scherer, S., (2019b), Monocular object and plane SLAM in structured environments, *IEEE Robotics and Automation Letters*, **4**(4), 3145–3152.

Yang, S., Song, Y., Kaess, M., & Scherer, S., (2016), Pop-up SLAM: semantic monocular plane SLAM for low-texture environments, *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1222–1229.

Yang, Z., Sun, Y., Liu, S., & Jia, J., (2020), 3dssd: point-based 3d single stage object detector, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11040–11048.

Yu, C., Liu, Z., Liu, X.-J., Xie, F., Yang, Y., Wei, Q., & Fei, Q., (2018), DS-SLAM: a semantic visual SLAM towards dynamic environments, *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1168–1174.

Yu, X., Chaturvedi, S., Feng, C., Taguchi, Y., Lee, T.-Y., Fernandes, C., & Ramalingam, S., (2018), Vlase: vehicle localization by aggregating semantic edges, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3196–3203.

Zaffar, M., Khaliq, A., Ehsan, S., Milford, M., & McDonald-Maier, K., (2019), Levelling the playing field: a comprehensive comparison of visual place recognition approaches under changing conditions, *arXiv preprint arXiv:1903.09107*.

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., & Lee, B., (2022), A survey of modern deep learning based object detection models, *Digital Signal Processing*, 103514.

Zeng, Z., Zhou, Y., Jenkins, O. C., & Desingh, K., (2018), Semantic mapping with simultaneous object detection and localization, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 911–918.

Zhang, J., Henein, M., Mahony, R., & Ila, V., (2020), VDO-SLAM: a visual dynamic object-aware SLAM system, *arXiv preprint arXiv:2005.11052*.

Zhang, T., & Nakamura, Y., (2018), Posefusion: dense RGB-D SLAM in dynamic human environments, *International Symposium on Experimental Robotics*, 772–780.

Zhang, Z., & Scaramuzza, D., (2018), A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7244–7251.

Zhao, B., Feng, J., Wu, X., & Yan, S., (2017), A survey on deep learning-based fine-grained object classification and semantic segmentation, *International Journal of Automation and Computing*, **14**(2), 119–135.

Zhao, C., Sun, L., & Stolkin, R., (2017), A fully end-to-end deep learning approach for real-time simultaneous 3d reconstruction and material recognition, *2017 18th International Conference on Advanced Robotics (ICAR)*, 75–82.

Zhi, S., Sucar, E., Mouton, A., Haughton, I., Laidlow, T., & Davison, A. J., (2021), Ilabel: interactive neural scene labelling, *arXiv preprint arXiv:2111.14637*.

Zhou, Q.-Y., Park, J., & Koltun, V., (2018), Open3D: A modern library for 3D data processing, *arXiv:1801.09847*.

Zhu, H., Meng, F., Cai, J., & Lu, S., (2016), Beyond pixels: a comprehensive survey from bottom-up to semantic image segmentation and cosegmentation, *Journal of Visual Communication and Image Representation*, **34**, 12–27.

Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M. R., & Pollefeys, M., (2022), Nice-SLAM: neural implicit scalable encoding for SLAM, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12786–12796.

Zins, M., Simon, G., & Berger, M.-O., (2022), Object-based visual camera pose estimation from ellipsoidal model and 3d-aware ellipse prediction, *International Journal of Computer Vision*, **130**(4), 1107–1126.

UNIVERSITÉ DE RENNES 1

Titre : Optimisation du SLAM visuel par analyse sémantique de l'environnement réel.

Mot clés : SLAM, sémantique, cartographie, localisation

Résumé : Le but du SLAM (Simultaneous Localization And Mapping) est d'estimer la trajectoire d'une caméra en mouvement tout en cartograhiant l'espace. Les algorithmes classiques construisent généralement une cartographie purement géométrique et homogène, ainsi il y a un *écart sémantique* entre la représentation interne du SLAM et le monde réel dans lequel le système évolue. Notre but dans ce manuscript est de construire un système de SLAM pouvant exploiter l'information sémantique pour repousser les limites du SLAM. Dans ce but nous proposons un réseau de neurones léger pour estimer la pose d'objets dans la scène. Les objets peuvent servir de repères haut niveau pour un SLAM, améliorant la pose de la caméra et ajoutant de l'information dans la cartographie. Puis nous proposons un SLAM capable de créer des groupes de points 3D correspondant à des objets génriques dans la scène. En utilisant une connaissance a priori sur la classe des objets nous pouvons estimer leur géométrie pour améliorer la cartographie et la pose de la caméra. Enfin, nous proposons un SLAM capable d'estimer la pose de la caméra dans des scène dynamiques tout en estimant la trajectoire de tous les objets dans la scène. Un a priori sur les objets nous permet de contraindre leur mouvement afin qu'il soit cohérent avec la structure du monde. Nous proposons également d'améliorer le suivi des objets en injectant des données LiDAR dans notre SLAM.

Title: Visual SLAM optimization by semantic analysis of the environment.

Keywords: SLAM, semantic, mapping, localization

Abstract: The goal of SLAM (Simultaneous Localization and Mapping) is to estimate the trajectory of a moving camera while building a map of its environment. Classical algorithm usually build a purely geometrical and homogeneous map, thus there is a *semantic gap* between the internal representation and the real world in which the system is evolving. Our goal in this manuscript is to build a SLAM system that can harness semantic information to push forward the limits of SLAM. To this end, we first propose a light neural network to estimate the pose of objects in the scene. Objects can serve as high level landmarks for a SLAM system, improving camera pose and adding information into the map. This network however has to be trained for specific objects. We then propose a SLAM system that can create clusters of 3D points corresponding to generic objects in the scene. With some a priori knowledge about object classes we can estimate their geometry in real time to improve both the map and camera pose estimation. Finally we propose new SLAM able to robustly estimate camera pose in dynamic scenes and to estimate the trajectories of all moving objects in the scene. A priori knowledge allows us to constrain the movement of objects to be plausible with respect to the structure of the world. We also propose to improve object tracking by injecting LiDAR data into our SLAM system.