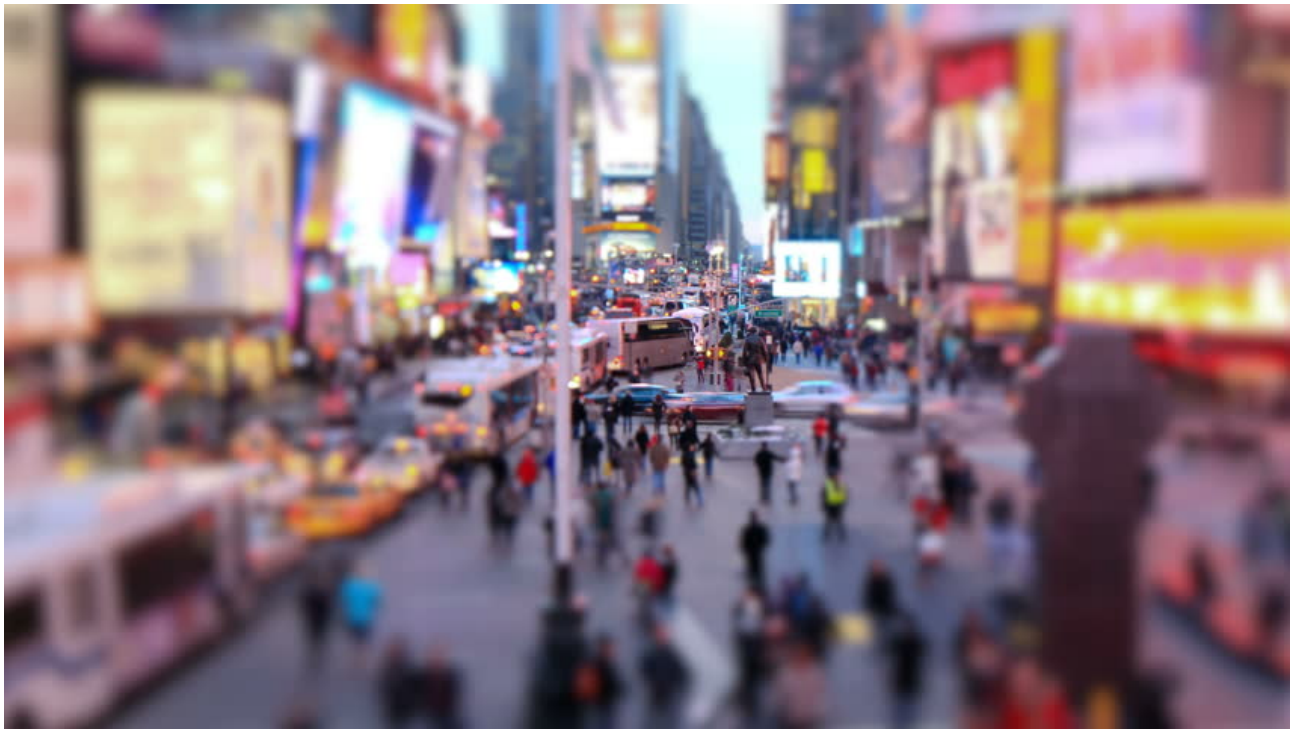
Australian National University

ACRV AUSTRALIAN CENTRE FOR ROBOTIC VISION

Doctoral Thesis

# Meta Information In Graph-based Simultaneous Localisation And Mapping




**Supervisors:**
Dr. Viorela Ila
Prof. Hongdong Li
Prof. Robert Mahony

**Author:**
Mina Henein

# Meta Information In Graph-based Simultaneous Localisation And Mapping

A thesis submitted to attain the degree of

DOCTOR OF PHILOSOPHY

of The Australian National University

Presented by

Mina Henein

MSc École Centrale de Nantes in Control and Robotics
MSc Università Degli Studi Di Genova in Robotics Engineering

accepted on the recommendation of

Dr. Viorela Ila, University of Sydney
Prof. Hongdong Li, Australian National University
Prof. Robert Mahony, Australian National University

June, 2020

Research School of Electrical, Energy and Materials Engineering
College of Engineering & Computer Science
Australian National University, Canberra
Australia

*To all those who wonder, all those who question, and all those who are always searching for meaning. Intelligence is curiosity, so learn to question and question to learn. Learn to be silent and listen, to close your eyes and watch, for in the most silent moments we hear the most beautiful songs, and in the darkest of places we see the brightest stars.*

Mina Henein

March $6^{th}$, 2020

# Declaration of Originality

I hereby confirm that I am the sole author of the written work entitled

**Meta Information In Graph-based Simultaneous Localisation And Mapping**

and that I alone have authored and written in my own words.

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Academic Integrity' https://www.anu.edu.au/students/academic-skills/academic-integrity.

The citation conventions usual to the discipline in question here have been respected.
The above written work may be tested electronically for plagiarism.

**Canberra, June 21$^{st}$, 2020**
Place and date

Signature

# Acknowledgements

First and foremost I would like to thank God. You have given me the power to believe in my passion and pursue my dreams. I could have never done this without the faith I have in you.

I would like to express my sincere gratitude to all of the faculty and staff of the School of Engineering and Computer Science at The Australian National University and The Australian Centre of Excellence for Robotic Vision.

I am much obliged to my supervisors Dr. Viorela Ila and Prof. Robert Mahony for their tremendous ongoing support.

My sincere thanks also go to my lab colleagues and collaborators, especially Montiel Abello, Gerard Kennedy, Jun Zhang, Zheyu Zhuang, Jean-Luc Stevens, Sean O'Brien, Klemen Istenič, Cristian Rodriguez, Rodrigo Santa Cruz, Cedric Scheerlink, Ehab Salahat and Pieter van Goor for their ongoing support and valuable ideas and recommendations.

Last but not the least, I would like to thank Magdalena Borkowska, Annachiara Arban and my family, without whom I could never imagine reaching this; my parents and sister, for believing in me, supporting me throughout my life and encouraging me during my studies.

# Abstract

Establishing the spatial and temporal relationships between a robot, and its environment serves as a basis for scene understanding. The established approach in the literature to simultaneously build a representation of the environment, and spatially and temporally localise the robot within the environment, is ***Simultaneous Localisation And Mapping*** (SLAM). SLAM algorithms in general, and in particular visual SLAM—where the primary sensors used are cameras—have gained a great amount of attention in the robotics and computer vision communities over the last few decades due to their wide range of applications. The advances in sensing technologies and image-based learning techniques provide an opportunity to introduce additional understanding of the environment to improve the performance of SLAM algorithms.

In this thesis, I utilise ***meta information*** in a SLAM framework to achieve a robust and consistent representation of the environment and challenge some of the most limiting assumptions in the literature. I exploit ***structural information*** associated with geometric primitives, making use of the significant amount of structure present in real world scenes where SLAM algorithms are normally deployed. In particular, I exploit planarity of a group of points and introduce higher-level information associated with orthogonality and parallelism of planes to achieve structural consistency of the returned map. Separately, I also challenge the static world assumption that severely limits the deployment of autonomous mobile robotic systems in a wide range of important real world applications involving highly dynamic and unstructured environments by utilising the ***semantic and dynamic information*** in the scene. Most existing techniques try to simplify the problem by ignoring dynamics, relying on a pre-collected database of objects 3D models, imposing some motion constraints or fail to estimate the full $\mathrm{SE}(3)$ motions of objects in the scene which makes it infeasible to deploy these algorithms in real life scenarios of unknown and highly dynamic environments. Exploiting semantic and dynamic information in the environment allows to introduce a model-free object-aware SLAM system that is able to achieve robust moving object tracking, accurate estimation of dynamic objects full $\mathrm{SE}(3)$ motion, and extract velocity information of moving objects in the scene, resulting in accurate robot localisation and *spatio-temporal map* estimation.

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

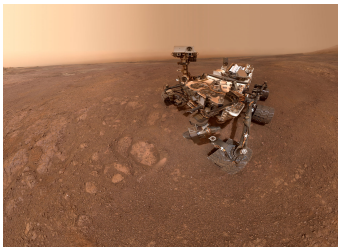
In a world where automation and intelligent systems are increasingly and rapidly having a huge impact in a number of various domains, robotics research is tremendously gaining attention. Among the areas that have gained a lot of attention, and continue to represent one of the most important building blocks in advanced robotics, is the ability to enable robots to perform tasks autonomously and independently. In order to achieve such capability, a robot must be able to reason about its surroundings and to build an understanding of the environment and of its location with respect to the environment. Robot localisation is the process of determining where a mobile robot is located with respect to its environment. Mapping is the ability of a robot to model the environment. Both localisation and mapping are two of the most fundamental competencies required by an autonomous robot. Simultaneous localisation and mapping (SLAM) algorithms are used for this purpose and are considered a core enabling technology for mobile robotics. SLAM is concerned with the problem of continually building a map of some unknown environment while being able at all times to determine the robot's location within this map.

## 1.1 Importance of SLAM for robotics

The ability to learn the location of a robot, as well as the environment around it, without knowing either beforehand is incredibly difficult. SLAM systems are however proving to be very effective at tackling this challenge.

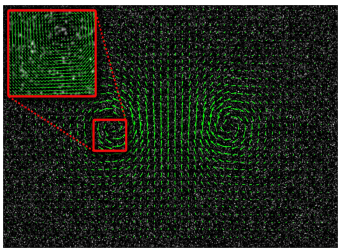
SLAM has many potential applications, and demand for this technology is tremendously increasing as it helps many products become more commercially viable such as augmented and virtual reality, and autonomous vehicles to name a few. SLAM systems are used in a wide variety of robots. From space rovers to drones and agriculture field robots, passing by vacuum cleaners, defect detection robots in mines, underwater reef monitory robots and nano-robots for

minimally invasive surgeries (MIS), SLAM represents a core building block to the deployment of autonomous robotic applications in real life.



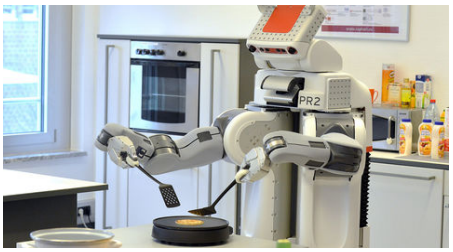
NASA curiosity rover.
https://mars.nasa.gov/resources/22273/curiositys-selfie-at-rock-hall/

Aqua reef monitoring robot.
https://robots.ieee.org/robots/aqua/



Particle image velocimetry.
http://wikiwand.com/en/Particle_image_velocimetry

Minimally invasive surgery robots.
https://sciencenewsforstudents.org/article/therapeutic-robots-may-soon-swim-within-body




PR2 service robot.
https://razorrobotics.com/robots/pr2

Google autonomous car.
https://waymo.com/press/

Figure 1.1: **Various SLAM applications.**

Visual SLAM systems, using visual inputs from a camera, are emerging as one of the most sophisticated computer vision technologies available. The advances in optical devices to capture *images* have given birth to various visual SLAM systems including but not limited to monocular, stereo, RGB-D, and event-based SLAM systems.

One major potential opportunity for visual SLAM is to replace Global Positioning System (GPS) tracking and navigation in certain applications. GPS-based systems are not useful indoors, and they suffer from degraded accuracy in locations with compromised sky views, such as "Urban Canyons" formed by high-rise buildings. Visual SLAM systems can help solve these problems. Autonomous driving in urban environments represents the biggest real application to visual SLAM algorithms nowadays, and one of the most important pillars to the deployment of autonomous vehicles. Other applications of SLAM include service, social and medical robots. This includes robot companions and assistants for elderly people, and other service robots that should interact and work closely with humans, which has been made possible after the advances in learning techniques to jointly perform action recognition and pose estimation [1, 2].

## 1.2 Meta information in SLAM

Based on the above areas of application, SLAM algorithms are deployed in almost every possible environment. From underwater, to space, passing by the human body, indoors, mines, and urban environments, SLAM algorithms should make use of the abundant prior information specific to each of these environments. Despite the wide range of applications of SLAM algorithms, indoor and urban environments remain to be the most common, and researched areas of application of SLAM techniques.

We use the prefix *meta-* from the Greek $\mu\epsilon\tau\alpha$, meaning "after" or "beyond" or "higher" and define meta information as the type of information that is concerned with higher-level and richer scene models, i.e. information *beyond* the directly measured data. Although the SLAM algorithms and techniques developed in this work are general enough to be deployed in any application, and form a framework for the integration of meta information to better constraint the problem, there is a clear focus on urban environments and autonomous driving as a primary application. Urban environments provide a large amount of important prior information that could be exploited into a SLAM framework. Meta information present in urban environments can provide cues to exploit the knowledge we have about the environments where SLAM algorithms are run, and to allow SLAM to challenge some of the most limiting assumptions made in the literature to solve the SLAM problem, such as the static world assumption. Challenging these assumptions is at the core of the deployment of SLAM algorithms to a wide range of real world scenarios.

Advances in learning techniques for processing information from visual data, enable high level sensing capabilities in robotic applications and provide rich information about the scene. Learning techniques are capable of providing: 1) geometric information about the scene in terms

of single image depth extraction from a monocular camera system [3], 2) structural information in the form of planes [4] present in the scene, 3) semantic information [5] to segment and classify −on instance-level− the objects present in the environment, and 4) dynamic information [6] to achieve motion segmentation. Utilising this information opens new doors for SLAM to achieve unprecedented results in environments that were always considered challenging. Examples of these environments include indoor environments with large featureless surfaces [7], environments represented at the object-level [8], dynamic environments where a significant part of the scene is moving [9], and others that have only become possible thanks to the rapid advances in learning algorithms to provide rich high-level information about the scene. Making use of meta information present in the environment has not only allowed deployment of SLAM algorithms in challenging environments [7, 8, 9] but also has increased the accuracy, consistency, and usability of the returned map [8, 9, 10, 11, 12, 13].

### 1.2.1 Structural information

All built environments, whether indoors or outdoors, contain significant amount of structure. Application of SLAM in urban environments, and especially to autonomous driving is probably the biggest monetary pillar and one of the main drivers of SLAM research nowadays. SLAM algorithms have to be run on-board the autonomous vehicle and provide a globally and structurally consistent map at every time step. Although applied to environments that contain significant structure, classical SLAM algorithms usually make no assumptions about the structure of the scene being analysed [14].

The first type of meta information that we explore in this thesis is the *structural information*. In particular, we exploit planar information of 3D points pertaining to planar surfaces. We also exploit parallelism and orthogonality between the detected planes. The plane parameters are incorporated into the estimation problem as latent variables that are not directly observed by the robot but rather inferred through the environment points, and/or other planes. We show that by utilising prior knowledge of the environment, more accurate and globally consistent solutions can be obtained.

### 1.2.2 Semantic information

The world is a complex and highly dynamic environment, and thus to allow robots to be part of our daily lives, research in autonomous robotics is rapidly departing from simple, controlled environments to environments that are more representative of the reality. SLAM is a well researched area in robotics, and while many efficient solutions to the problem exist, most of the existing techniques heavily rely on the static world assumption [15] to simplify the problem.

Such an assumption limits the deployment of SLAM algorithms in highly dynamic environments, where they are destined to fail due to the lack of reliable static structure. Developments in the field of 3D scene understanding to provide *semantic information* offer promising ways to challenge the static world assumption towards the deployment of SLAM algorithms in high dynamic environments.

The second type of meta information explored in this thesis is *semantic information*. We propose a feature-based algorithm that exploits semantic information to integrate rigid body motion of objects into a SLAM framework, without the need to estimate the object pose or to have any prior knowledge of the object's 3D model. Through a frame change of a pose transformation, we are able to describe the motion of rigid objects in the scene in terms of the points that belong to the object in a model-free manner. We show that, in highly dynamic environments, and by utilising semantic information and prior knowledge about the rigidity of the objects in the scene, our algorithm produces consistently better results compared to other algorithms that exclude moving features from the SLAM estimation problem. We also fully exploit the rigid object motion to extract velocity information of objects in the scene, an emerging task in autonomous driving which has not yet been thoroughly explored [16]. Such information is crucial to aid autonomous driving algorithms for tasks such as collision avoidance [17] or adaptive cruise control [18]. To the best of our knowledge, at the time of writing this document, this is the first work able to estimate, along with the camera poses, the static and dynamic structure, the full $\mathrm{SE}(3)$ pose change of every rigid object in the scene, extract object velocities and be demonstrable on a real-world outdoor dataset.

### 1.2.3 Dynamic information

Advances in deep learning have provided algorithms that can reliably detect and segment classes of objects at almost real time [19, 5]. This semantic information has not yet been fully exploited within the SLAM community [20]. Exploiting the capabilities of modern deep learning techniques in providing semantic scene understanding opens the doors towards integrating the dynamic information into SLAM algorithms.

In the last piece of work presented here, we propose a model-free, object-aware point-based dynamic full SLAM system that leverages image-based semantic information to integrate *dynamic information* of the scene into the estimation, and estimate the motion of dynamic objects in the environment jointly with performing SLAM. The proposed system is able to simultaneously localise the robot, map the static structure, estimate motions of dynamic objects and build a dynamic representation of the world, and is engineered to work in real world, and yield consistent results in indoor and outdoor challenging scenarios. To the best of our knowledge, at the time of writing this document, this is the first full object-aware dynamic SLAM system that is

able to achieve motion segmentation, dynamic object tracking, and estimate the camera poses along with the static and dynamic structure, the full $\mathrm{SE}(3)$ pose change of every rigid object in the scene, extract velocity information of moving objects, and outperforms every state of the art similar dynamic SLAM system in estimation accuracy.

## 1.3 Approach and contributions

The scope of this thesis is to develop, prototype and test algorithms that utilise meta information in order to challenge the most adopted assumptions for SLAM problems in challenging environments. We namely utilise structural, semantic, and dynamic information to improve the SLAM estimation accuracy. The main assumption we try to challenge is the static world assumption, that is highly adopted in the SLAM literature to simplify the problem and that hinders the deployment of autonomous robotic vehicles in a wide range of real-world applications.
We redefine the term *mapping* in SLAM to be concerned with a *spatio-temporal representation of the world*, as opposed to the concept of a static map that has long been the emphasis of classical SLAM algorithms, including SLAM systems that can robustly operate in dynamic environments by excluding the dynamics of the world.

### 1.3.1 Proposed approach

In this thesis, we propose a novel SLAM system that utilises meta information to improve the SLAM estimation accuracy and robustness, and produce more consistent maps. We aim to transfer the prior knowledge we have about the places we live in, particularly urban environments, to robots undertaking the SLAM estimation problem, with the aim to achieve a better understanding of the environment through exploiting the structural, semantic and dynamic prior information about the environment. This information is provided by other algorithms whose outputs could be thought of as sensor inputs to our system. We propose to jointly solve a classical SLAM problem augmented with latent variables that constitute entities of which one has some prior knowledge but whose states can not be directly measured by the robot.

We first propose to extract planes from a point cloud of 3D points. These planes are integrated into a graph-based SLAM algorithm and their parameters estimated. Although not directly measured, these planes are constrained by other environment points and detected planes.
We then utilise instance-level semantic segmentation to detect and segment objects in the scene. Our model-free object-aware feature-based visual SLAM algorithm is able to perform simultaneous localisation, mapping and moving object tracking in a single SLAM framework. The system is designed to handle the environment dynamics and estimate a full $\mathrm{SE}(3)$ pose change of every rigid object in the scene. Moreover, we exploit the estimated object motions to extract

velocity information of every detected object in the environment. Our system is proven to be robust in challenging indoor and outdoor scenarios, and is able to integrate information about dynamic and static structures in the environment into a single estimation framework resulting in accurate robot pose and spatio-temporal (time-varying) map estimation.

Our main argument is that SLAM and scene understanding algorithms are mutually beneficial as follows:

1. On one hand, object and layout understanding benefits the SLAM pose estimation and mapping. The high-level information such as planes and objects in the scene can provide additional semantic, geometric, and dynamic constraints to improve the SLAM estimates.

2. On the other hand, SLAM improves the accuracy and robustness of 3D scene understanding. Robust single image object segmentation is difficult due to occlusions. However, multi-view information and SLAM estimates can be used to refine and recover failure cases of object segmentation and tracking.

### 1.3.2 Contributions

The main contributions of this thesis can be summarised in the following:

- a graph-based SLAM framework that is able to integrate structural information about the environment, and an optimisation method that offers the possibility to optimise the layout planes present in the environment in addition to the camera poses, and the static structure (section 4.2.1 in chapter 4),

- a novel way to express motion of rigid objects in the scene in terms of points that belong to the object ((5.5) in chapter 5) and integrate information about dynamic and static structures in the environment into a single SLAM framework in a model-free manner (section 5.2.2 in in chapter 5) resulting in accurate robot pose and spatio-temporal map estimation, accurate dynamic objects full $\mathrm{SE}(3)$ pose change estimation, and a way to extract velocity information of moving objects in the scene ((5.7) in chapter 5),

- a full object-aware feature-based dynamic visual SLAM system that outperforms every state-of-the-art similar SLAM systems and yields object motion estimation results that are comparable to the camera pose estimation (section 6.3 in chapter 6),

- a robust method to enhance scene 3D understanding for moving object tracking exploiting image-based semantic information (section 6.2.2.3 in chapter 6), and the ability to deal with indirect occlusions resulting from the failure of semantic object segmentation (section 6.3.3.3 in chapter 6), and

- an open-source code, for each of the SLAM systems developed in this thesis, that is made available for the community to develop and extend including novel observation functions to integrate structural, semantic and dynamic information, and is made flexible to include any type of implicit information as long as there is an algorithm "sensor" that can provide this information and a function that can model it (chapter 7).

## 1.4 Thesis road map

The remainder of this thesis is structured as follows:

- Chapter 2 reviews the necessary mathematical preliminaries and problem formulation in the literature.
- Chapter 3 reviews the related work and describes the main components of a typical classical SLAM system, and details the off-the-shelf components in the literature that constitute the meta information provided by the front-end.
- Chapter 4 shows how to integrate structural information about the environment layout into the SLAM estimation problem and describes effects of adding such information on the global consistency of SLAM. This chapter is mostly based on the work published in "Exploring The Effect of Meta-Structural Information on the Global Consistency of SLAM".
- Chapter 5 describes a novel way to integrate dynamic information about the environment into the SLAM problem through exploiting image-based semantic information, and expressing the motion of rigid objects in the scene using points that belong to the object in a feature-based model-free dynamic SLAM system. This chapter is mostly based on the work published in "Dynamic SLAM: The Need for Speed".
- Chapter 6 details a model-free object-aware full dynamic SLAM system that exploits semantic information to enhance scene 3D understanding and moving object tracking, yielding results that outperform every state-of-the-art similar system. This chapter is based on a collaborative work with Jun Zhang, that resulted in the paper "VDO-SLAM: A Visual Dynamic Object-aware SLAM System".
- Chapter 7 describes the technical contribution of this thesis in terms of open-source code, and efforts made to simplify implementation, testing and extension of the current work.

- Chapter 8 summarises the outputs of this thesis and offers concluding remarks and open questions.

## 1.5 Publications

This thesis is based on the following peer-reviewed publications:

### 1.5.1 Journals

- **VDO-SLAM: A Visual Dynamic Object-aware SLAM System.**
  Jun Zhang*, Mina Henein*, Robert Mahony and Viorela Ila.
  * the two authors contributed equally to this work.
  To be submitted to IEEE Transactions on Robotics (T-RO).

### 1.5.2 Conferences

- **Exploring The Effect of Meta-Structural Information on the Global Consistency of SLAM.**
  Mina Henein, Montiel Abello, Viorela Ila and Robert Mahony.
  In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

- **Dynamic SLAM: The Need for Speed.**
  Mina Henein, Jun Zhang, Robert Mahony and Viorela Ila.
  In IEEE International Conference on Robotics and Automation (ICRA), 2020.

- **Robust Ego and Object 6-DoF Motion Estimation and Tracking**
  Jun Zhang, Mina Henein, Robert Mahony, and Viorela Ila.
  In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.
  (under review)

### 1.5.3 Workshops

- **Exploiting Rigid Body Motion for SLAM in Dynamic Environments.**
  Mina Henein, Gerard Kennedy, Robert Mahony and Viorela Ila.
  In IEEE International Conference on Robotics and Automation (ICRA), 2018.

Workshop paper: Inference and Learning for Joint Semantic, Geometric and Physical Understanding.

### 1.5.4 Misc

Publicly available source code:

- **Entity SLAM**
  https://github.com/MinaHenein/Entity-SLAM/wiki
  main developers and collaborators: Montiel Abello
- **DO SLAM**
  https://github.com/MinaHenein/do-slam
  main developers and collaborators: Gerard Kennedy & Yash Vyas
- **VDO-SLAM**
  https://github.com/halajun/VDO_SLAM
  main developers and collaborators: Jun Zhang

Chapter 2

# Preliminaries

The Simultaneous Localisation And Mapping is concerned with the problem of building a map of an unknown environment by a mobile robot while at the same time determining its location within this map. The term SLAM was originally coined by Hugh Durrant-Whyte and John J. Leonard [21] based on earlier work by Smith, Self and Cheeseman [22]. Durrant-Whyte and Leonard originally termed it SMAL but it was later changed to give a better impact.

In an attempt to make this document as self-contained as possible, a number of mathematical concepts such as multivariate differentiation, Taylor series, numerical optimization techniques, and probabilistic state estimation and its relation to least squares problems are presented in the below. Moreover, an introduction to Lie groups and some related concepts are presented as a generalisation over the Euclidean vectors space.

## 2.1 Multivariate differentiation

A function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ , which maps a vector onto a vector, is called *a vector field*.
The first derivative of a vector field $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is the Jacobian $\mathrm{J}_\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^{(m\times n)}$ which is defined as:

$$\mathrm{J}_\mathbf{f} := \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix} \tag{2.1}$$

The second derivative of a vector field $\mathbf{f}$ is called the Hessian $\mathrm{H}_\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^{(n\times m\times n)}$ which maps a vector onto a three dimensional array or a third-order tensor.

## 2.2 Taylor series

Let $f : \mathbb{R} \to \mathbb{R}$ be an infinitely differentiable function. The power series

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots = \sum_{k=0}^{\infty} \frac{f^k(a)}{k!}(x-a)^k \tag{2.2}$$

is called the Taylor series of $f$ at $a$, where $f^k(a)$ denotes the $k$th derivative of $f$ evaluated at the point $a$, and the derivative of order zero of $f$ is defined to be $f$ itself.
In practice, a function $f$ in the neighbourhood of a point $a$ is approximated using a finite series, the $n$th-order Taylor expansion:

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + \cdots + \frac{f^n(a)}{n!}(x-a)^n = \sum_{k=0}^{n} \frac{f^k(a)}{k!}(x-a)^k \tag{2.3}$$

## 2.3 Introduction to optimisation

SLAM is inherently an optimisation problem. In typical optimisation problems, one would like to find the minimum of $F$ :

$$\min_{\mathbf{x} \in \mathbb{R}^n} F(\mathbf{x}) \tag{2.4}$$

For a general scalar field $F : \mathbb{R}^n \to \mathbb{R}$, which maps a vector onto a scalar, and even if one assumes it is infinitely differentiable, there is no guarantee to find such a global minimum in a finite number of steps. Therefore, one often focuses on finding a local minimum in the neighbourhood of an initial guess $\mathbf{x_0}$ instead. If $\langle \bar{\mathbf{x}}, F(\bar{\mathbf{x}}) \rangle$ is a local minimum of $F$, then $\nabla F(\bar{\mathbf{x}}) = 0$, is the *necessary condition*. Furthermore, if $\nabla F(\bar{\mathbf{x}}) = 0$ and $\mathrm{H}_F(\bar{\mathbf{x}})$ is *positive definite*, thus $\forall_{y \in \mathbb{R}^n \setminus \{0\}} \mathrm{y}^\top \mathrm{H}_F(\bar{\mathbf{x}})\, \mathrm{y} > 0$, $\langle \bar{\mathbf{x}}, F(\bar{\mathbf{x}}) \rangle$ is a local minimum. This is the *sufficient condition*.

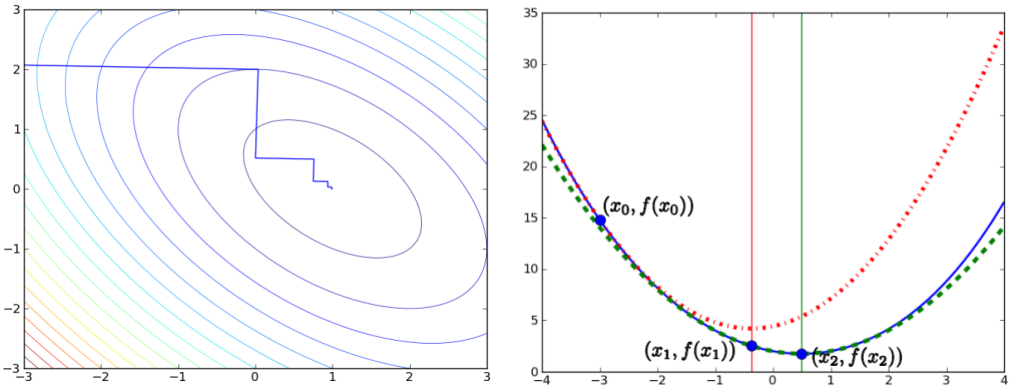
### 2.3.1 Gradient descent

The simplest approach to find the minimum of $F$ in the neighbourhood of $\mathbf{x}^{(0)}$ is the gradient descent method, which iteratively takes steps along the current negative gradient direction $-\nabla F(\mathbf{x^{(k)}})$, and the update rule is

$$\mathbf{x^{(k+1)}} = \mathbf{x^{(k)}} - \alpha_k \, \nabla F(\mathbf{x^{(k)}}) \tag{2.5}$$

Typically, the factor $\alpha_k > 0$ is selected in a way such that $F(\mathbf{x^{(k+1)}}) \ll F(\mathbf{x^{(k)}})$. If no such $\alpha_k$ exists, the minimum is reached. The gradient descent method is a first-order iterative algorithm, easy to implement, and is guaranteed to converge locally. However, the convergence rate can be

low, especially close to the minimum, as the gradient becomes quite small [23].




(a) Gradient descent. (b) Newton's method.

Figure 2.1: **Gradient descent vs Newton method.** (a) Method of gradient descent illustrated on a quadratic form. The method always takes steps along the direction of the steepest descent, which leads to a "zig-zagging" effect and thus a slow convergence close to the minimum. (b) The Newton method is illustrated on a higher-order polynomial (solid blue curve). The neighbourhood around the initial guess $x_0$ is approximated with a positive definite quadratic form (red dashed parabola). The initial update is performed by stepping to the minimum of this parabola (vertical red line). A second update is also shown (green parabola, green vertical line) which brings the estimate very close to the optimum [24].

### 2.3.2 Newton method

A more efficient approach is Newton's method, which requires $F$ to be twice differentiable and approximates it by a local quadratic function at each iteration and takes a step towards the minimum of this quadratic function. Since $\mathbf{x_0}$ is assumed to be in the neighbourhood of $\bar{\mathbf{x}}$, $\nabla F(\bar{\mathbf{x}})$ can be approximated using the first order Taylor expansion as

$$\nabla F(\bar{\mathbf{x}}) \approx \nabla F(\mathbf{x}_0) + \mathrm{H}_F(\mathbf{x}_0)\,(\bar{\mathbf{x}} - \mathbf{x}_0) \tag{2.6}$$

and since $\nabla F(\bar{\mathbf{x}}) = 0$ (the necessary condition), $\bar{\mathbf{x}} \approx \mathbf{x}_0 - \mathrm{H}_F^{-1}(\mathbf{x}_0)\,\nabla F(\mathbf{x_0})$.
The update rule is then

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathrm{H}_F^{-1}(\mathbf{x}^{(k)})\,\nabla F(\mathbf{x}^{(k)}) \tag{2.7}$$

Defining the incremental update as $\boldsymbol{\delta} := \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ , the Newton method is performed by repetitively solving the linear system

$$\mathrm{H}_F(\mathbf{x}^{(k)})\,\boldsymbol{\delta} = -\nabla F(\mathbf{x}^{(k)}) \tag{2.8}$$

To clearly see that the Newton's method approximates the function $F$ at $\mathbf{x}^{(k)}$ with a quadratic function, let us consider the quadratic form,

$$\frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} - \mathbf{b}^\top \mathbf{x} + c \tag{2.9}$$

If $\mathbf{A}$ is symmetric and positive semi-definite, the quadratic form is minimal for $\mathbf{A}\mathbf{x} = \mathbf{b}$ [25]. From (2.8), it becomes clear that the Newton's method approximates the function $F$ at $\mathbf{x}^{(k)}$ with a quadratic form with $\mathbf{A} = \mathrm{H}_F(\mathbf{x}^{(k)})$ and $\mathbf{b} = -\nabla F(\mathbf{x}^{(k)})$. Hence, if $F$ is a quadratic form, the Newton's method will converge in one iteration. Note that any function is approximately quadratic around its minimum if it is twice differentiable. Thus in contrast to the gradient descent method, Newton's method converges especially fast in the neighbourhood of the minimum as shown in Fig. 2.1. For high-dimensional problems, however, it is often intractable to compute the Hessian $\mathrm{H}_F(a)$.

### 2.3.3 Gauss-Newton method

An efficient variant of the Newton method is the Gauss-Newton method, which requires that $F$ to be of the following class:

$$F(\mathbf{x}) = a\,\mathbf{d}(\mathbf{x})^\top \Lambda \mathbf{d}(\mathbf{x}) \tag{2.10}$$

with $a > 0$, $\mathbf{d} : \mathbb{R}^n \to \mathbb{R}^m$ a twice differentiable vector field, and $\Lambda \in \mathbb{R}^{m\times m}$ being a symmetric, positive semi-definite matrix.
Even though this optimisation method seems to be limited to a specific class of problems, it covers a large number of applications. Namely, it covers least squares optimisation problems, where the function to minimise is a quadratic cost, and of which SLAM takes part.
The first derivative of $F$, with $a$ taken as $\frac{1}{2}$ which is an arbitrary choice that does not change the location of the function minima, and using the fact that $\Lambda$ is symmetric, becomes:

$$\nabla F = \frac{1}{2}(\mathbf{d}(\mathbf{x})^\top \Lambda \mathrm{J}_\mathbf{d}(\mathbf{x}))^\top + \frac{1}{2}(\mathrm{J}_\mathbf{d}(\mathbf{x})^\top \Lambda \mathbf{d}(\mathbf{x})) = \mathrm{J}_\mathbf{d}(\mathbf{x})^\top \Lambda \mathbf{d}(\mathbf{x}) \tag{2.11}$$

And the second derivative of $F$ is then:

$$\mathrm{H}_F(\mathbf{x}) = \mathrm{J}_\mathbf{d}(\mathbf{x})^\top \Lambda \mathrm{J}_\mathbf{d}(\mathbf{x}) + \mathrm{H}_\mathbf{d} \Lambda \mathbf{d}(\mathbf{x}) \tag{2.12}$$

with $\mathrm{H}_\mathbf{d}$ being the Hessian tensor of $\mathbf{d}$.
The Gauss-Newton method approximates the Hessian of $F$ as

$$\mathrm{H}_F(\mathbf{x}) \approx \mathrm{J}_\mathbf{d}(\mathbf{x})^\top \Lambda \mathrm{J}_\mathbf{d}(\mathbf{x}) \tag{2.13}$$

with the second-order derivative terms being negligible.
The linear system in (2.8) is approximated by the so-called *normal equation*

$$(\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathrm{J}_{\mathbf{d}})\,\boldsymbol{\delta} = -\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathbf{d} \tag{2.14}$$

This linear equation is then solved

$$\boldsymbol{\delta} = -(\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathrm{J}_{\mathbf{d}})^{-1}\,\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathbf{d} \tag{2.15}$$

and the update rule is applied as:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathrm{J}_{\mathbf{d}})^{-1}\,\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathbf{d} \tag{2.16}$$

Global convergence of the Gauss-Newton method is not guaranteed, and it might only converge to a local minimum that depends on the initial solution. In practice, if the objective function is locally well-approximated by a quadratic form, then convergence is quadratic. However, the curvature of the error surface of a non-linear observation model can vary significantly over the parameter space. The Levenberg-Marquardt method is a refinement to the Gauss-Newton procedure that increases the chance of convergence and prohibits divergence. Results still depend on the starting point [26].

### 2.3.4 Levenberg-Marquardt method

The Gauss-Newton method will fail in the degenerate case when $\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathrm{J}_{\mathbf{d}}$ is singular.
Moreover, as mentioned above, the gradient descent method is guaranteed to converge but suffers poor performance close to the minimum. On the contrary, the Gauss-Newton method works particularly well close to the minimum, but elsewhere the cost may not decrease at each iteration as the quadratic approximation may not always be a good approximation at this specific point.
An alternative method to combine the advantages of both is the Levenberg-Marquadt algorithm which interpolates gradient descent and Gauss-Newton by altering the normal equations as follows:

$$(\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathrm{J}_{\mathbf{d}} + \mu\mathrm{I})\,\boldsymbol{\delta} = -\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathbf{d} \tag{2.17}$$

where $\mu$ is a damping ratio and $I$ is the identity matrix. The parameter $\mu > 0$ steers the update vector $\delta$ towards the direction of the steepest descent. As $\mu$ approaches zero, Levenberg-Marquardt method approaches the standard Gauss-Newton method. On the other hand, if $\mu$ approaches infinity, the matrix $(\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathrm{J}_{\mathbf{d}} + \mu\mathrm{I})$ approaches a diagonal matrix with infinite trace. Thus, as $\mu \to \infty$, $\boldsymbol{\delta} = -\frac{1}{\mu}\mathrm{J}_{\mathbf{d}}^{\top}\Lambda\mathbf{d}$, and the Levenberg-Marquardt method approaches a gradient

descent update. The parameter $\mu$ is adjusted in each optimisation iteration. Only if the update $\mathbf{x}^{(k)} + \boldsymbol{\delta}$ reduces the cost ($F(\mathbf{x}^{(k)} + \boldsymbol{\delta}) \ll F(\mathbf{x}^{(k)})$), the update is accepted ($\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}$) indicating the algorithm is approaching the local minimum and hence $\mu$ is reduced to strengthen the influence of Gauss-Newton. However, if the update does not reduce the cost, it is rejected, and a larger $\mu$ (smaller step size) and an update more oriented towards the steepest descent direction is attempted.

## 2.4 SLAM problem formulation

### 2.4.1 SLAM as Bayesian belief net

The SLAM problem can be formulated as a Bayesian *belief network*. [27, 28, 29, 30, 31]. A belief net is a directed acyclic graph that encodes the conditional independence structure of a set of variables, where each variable only directly depends on its predecessors in the graph [32].

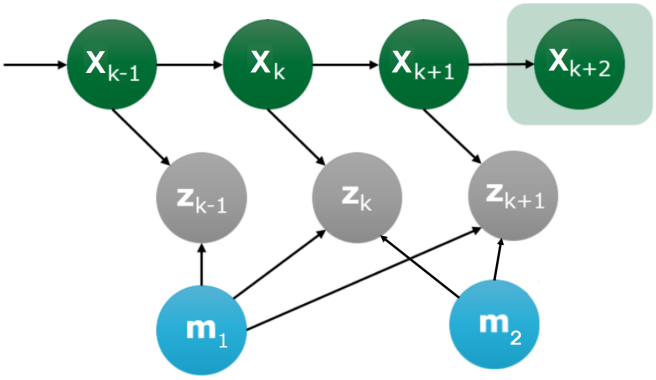



Figure 2.2: **SLAM as a Bayesian belief net.**

Fig. 2.2 shows a Bayesian belief network representation of a SLAM problem. The state of the robot at the $i$-th time step is denoted by $X_k$, an environment landmark by $\boldsymbol{m}^i$, and a measurement by $\boldsymbol{z}_k$. The joint probability model corresponding to this network is

$$P(\boldsymbol{\theta}_X, \boldsymbol{\theta}_M, \boldsymbol{\theta}_Z) = P(X_0) \prod_{k=1}^{m_k} P(\boldsymbol{u}_k | X_{k-1}, X_k) \prod_{z[1]}^{z[m_i]} P(\boldsymbol{z}_k^i | X_k, \boldsymbol{m}_k^i) \tag{2.18}$$

where $\boldsymbol{\theta}_X, \boldsymbol{\theta}_M, \boldsymbol{\theta}_Z$ are the sets of all robot states, environment landmarks and measurements respectively, $P(X_0)$ is a prior on the robot initial state, $P(\boldsymbol{u}_k | X_{k-1}, X_k)$ is the motion model, parameterised by a control input $\boldsymbol{u}_k$, and $P(\boldsymbol{z}_k^i | X_k, \boldsymbol{m}_k^i)$ is the landmark measurement model. The above equation assumes a uniform prior over all the landmarks.

Gaussian process and measurement models are assumed as is standard in the SLAM literature [22, 33, 34, 35, 36]:

$$\boldsymbol{u}_k = f_k(X_{k-1}, X_k) + v_k \qquad \Leftrightarrow \qquad P(\boldsymbol{u}_k|X_{k-1}, X_k) \propto \exp -\frac{1}{2}\|f_k(X_{k-1}, X_k) - \boldsymbol{u}_k\|^2_{\Sigma_{v_k}} \tag{2.19}$$

where $f_k(.)$ is the process model and $v_k \sim \mathcal{N}(0, \Sigma_{v_k})$ is normally distributed zero-mean process noise with co-variance matrix $\Sigma_{v_k}$.

$$\boldsymbol{z}^i_k = h^i_k(X_k, \boldsymbol{m}^i_k) + w^i_k \qquad \Leftrightarrow \qquad P(\boldsymbol{z}^i_k|X_k, \boldsymbol{m}^i_k) \propto \exp -\frac{1}{2}\|h^i_k(X_k, \boldsymbol{m}^i_k) - \boldsymbol{z}^i_k\|^2_{\Sigma_{w^i_k}} \tag{2.20}$$

where $h^i_k(.)$ is the landmark measurement model, and $w^i_k \sim \mathcal{N}(0, \Sigma_{w^i_k})$ is normally distributed zero-mean measurement noise with covariance $\Sigma_{w^i_k}$.
In the equations above, $\|e\|^2_\Sigma \triangleq e^\top \Sigma^{-1} e$ is defined as the squared Mahalanobis distance given a covariance matrix $\Sigma$.

### 2.4.2 SLAM as a factor graph

While belief nets are a very natural way to represent the generative aspect of the SLAM problem, *factor graphs* have a much tighter connection with the underlying optimisation problem [32]. As the measurements are known, one is allowed to eliminate them as variables, and consider them as parameters of the joint probability factors over the true unknowns. This naturally leads to the well known factor graph representation (such as the one shown in Fig.3.2), a class of bipartite graphical models that can be used to represent such factored densities [37]. In a factor graph there are nodes for unknowns and nodes for the probability factors defined on them, and the graph structure expresses which unknowns are involved in each factor. More details about factor graphs are presented in section 3.1.1 of chapter 3. A third way to express the SLAM problem in terms of graphical models is via *Markov random fields*, in which the factor nodes themselves are eliminated.

### 2.4.3 SLAM as a least squares problem

This section is concerned with the inference, i.e., obtaining an optimal estimate for the set of unknowns given all measurements.

The maximum a posteriori (MAP) estimates the entire trajectory $\boldsymbol{\theta}_X \triangleq X \triangleq \{X_k\}$ and the map $\boldsymbol{\theta}_M \triangleq M \triangleq \{\boldsymbol{m}^i_k\}$, given the measurements $Z \triangleq \{\boldsymbol{z}^i_k\}$ and control inputs $U \triangleq \{\boldsymbol{u}_k\}$. Let

$\boldsymbol{\theta} \triangleq \boldsymbol{\theta}_X \cup \boldsymbol{\theta}_M$ be the total system state. The MAP is

$$\boldsymbol{\theta}^* \triangleq \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P(X, M|Z) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P(X, M, Z) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} -\log(P(X, M, Z)) \tag{2.21}$$

which leads via (2.19) and (2.20) to the non-linear least squares:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \sum_{k=1}^{m_i} \|f_k(X_{k-1}, X_k) - \boldsymbol{u}_k\|^2_{\Sigma_{v_k}} + \sum_{z[1]}^{z[m_k]} \|h^i_k(X_k, \boldsymbol{m}^i_k) - \boldsymbol{z}^i_k\|^2_{\Sigma_{w_k}} \right\} \tag{2.22}$$

Non-linear optimisation methods such as Gauss-Newton or the Levenberg-Marquardt method are used to solve a succession of linear approximations to (2.22) in order to reach the minimum [38]. The process terms in (2.19) are linearised using the first order Taylor expansion as:

$$\begin{aligned} f_k(X_{k-1}, X_k) - \boldsymbol{u}_k &\approx \{f_k(X^0_{k-1}, X^0_k) + F^{k-1}_k \delta X_{k-1} + G^k_k \delta X_k\} - \boldsymbol{u}_k \\ &= F^{k-1}_k \delta X_{k-1} + G^k_k \delta X_k - \boldsymbol{a}_k \end{aligned} \tag{2.23}$$

where $F^{k-1}_k$ and $G^k_k$ are the Jacobians of $f_k(.)$ at the linearisation point $(X^0_{k-1}, X^0_k)$ defined as $F^{k-1}_k \triangleq \frac{\partial f_k(X_{k-1}, X_k)}{\partial X_{k-1}}|_{(X^0_{k-1}, X^0_k)}$ and $G^k_k \triangleq \frac{\partial f_k(X_{k-1}, X_k)}{\partial X_k}|_{(X^0_{k-1}, X^0_k)}$, and $\boldsymbol{a}_k \triangleq \boldsymbol{u}_k - f_k(X^0_{k-1}, X^0_k)$ is the odometry prediction error.

The measurement terms in (2.20) are similarly linearised as:

$$h^i_k(X_k, \boldsymbol{m}^i_k) - \boldsymbol{z}_k \approx \{h^i_k(X^0_k, {\boldsymbol{m}^i_k}^0) + H^k_k \delta X_k + J^i_k \delta \boldsymbol{m}^i_k\} - \boldsymbol{z}_k = H^k_k \delta X_k + J^i_k \delta \boldsymbol{m}^i_k - \boldsymbol{c}_k \tag{2.24}$$

where $H^k_k$ and $J^i_k$ are the Jacobians of $h^i_k(.)$ with respect to $X_k$ and $\boldsymbol{m}^i_k$ respectively at the linearisation point $(X^0_k, {\boldsymbol{m}^i_k}^0)$ defined as

$H^k_k \triangleq \frac{\partial h^i_k(X_k, \boldsymbol{m}^i_k)}{\partial X_k}|_{(X^0_k, {\boldsymbol{m}^i_k}^0)}$ and $J^i_k \triangleq \frac{\partial h^i_k(X_k, \boldsymbol{m}^i_k)}{\partial \boldsymbol{m}^i_k}|_{(X^0_k, {\boldsymbol{m}^i_k}^0)}$, and $\boldsymbol{c}_k \triangleq \boldsymbol{z}_k - h^i_k(X^0_k, {\boldsymbol{m}^i_k}^0)$ is the measurement prediction error.

Using the linearised process and measurement models in (2.23) and (2.24), a linear least squares in $\boldsymbol{\delta}$ is obtained:

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \left\{ \sum_{k=1}^{m_i} \|F^{k-1}_k \delta X_{k-1} + G^k_k \delta X_k - \boldsymbol{a}_k\|^2_{\Sigma_{v_k}} + \sum_{z[1]}^{z[m_k]} \|H^k_k \delta X_k + J^i_k \delta \boldsymbol{m}^i_k - \boldsymbol{c}_k\|^2_{\Sigma_{w_k}} \right\} \tag{2.25}$$

The Mahalanobis distance can be rewritten as $\|e\|^2_\Sigma \triangleq e^\top \Sigma^{-1} e = (\Sigma^{-\top/2} e)^\top (\Sigma^{-\top/2} e) = \|\Sigma^{-\top/2} e\|^2_2$ which allows us to rewrite (2.25), after collecting the Jacobian matrices into a matrix $A$, and the vectors $\boldsymbol{a}_k$ and $\boldsymbol{c}_k$ into a right-hand

side vector $\boldsymbol{b}$, the following standard least-squares problem is obtained

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \frac{1}{2}\|A\boldsymbol{\delta} - \mathbf{b}\|_2^2 \,, \tag{2.26}$$

## 2.5 Lie groups

The optimisation methods presented in section 2.3 are applicable for scalar fields which are defined on Euclidean vector spaces $\mathbb{R}^n$ and do not apply to non-Euclidean spaces such as 3D rotations. When performing an optimisation, an incremental update $\boldsymbol{\delta} \in \mathbb{R}^n$ is calculated and added to the current estimate $\mathbf{x}^{(k)} \in \mathbb{R}^n$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta} \tag{2.27}$$

Performing a rotation by $\boldsymbol{\omega}$ and then by $\boldsymbol{\delta}$ is in general not equivalent to performing a rotation of $\boldsymbol{\omega} + \boldsymbol{\delta}$. Here think of $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$ as being any parametrisation of rotation in 3D. Thus, rotations cannot be modelled as a Euclidean vector space, but as a *Lie group*.
Every Lie group has an associated *Lie algebra*, which is the tangent space around the identity element of the group, generated by differentiating the group transformations along chosen directions in the space, at the identity transformation. The exponential map converts any element of the tangent space (Lie algebra) exactly into a transformation in the Lie group. This section aims at revisiting the general update rule for non-Euclidean spaces. For more details about the mathematical definition of group, manifold, tangent space, Lie algebra, we refer the reader to [24, 39, 40].

### 2.5.1 Special Orthogonal group $\mathrm{SO}(3)$

The special orthogonal group $\mathrm{SO}(3)$ represents the group of 3D rotations, and its associated Lie algebra $\mathrm{so}(3)$ has three basis *generator* matrices which correspond to the derivatives of rotation around each of the standard axes, evaluated at the identity:

$$G_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad G_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \quad G_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

A skew-symmetric matrix $[\boldsymbol{\omega}]_\times \in \mathrm{so}(3)$ is then represented as a linear combination of the generators:

$$[\boldsymbol{\omega}]_{\mathrm{x}} = \omega_1 \, G_1 + \omega_2 \, G_2 + \omega_3 \, G_3 = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{2.28}$$

The exponential map that takes skew symmetric matrices to rotation matrices is simply the matrix exponential over a linear combination of the generators:

$$\exp : \mathrm{so}(3) \mapsto \mathrm{SO}(3) \tag{2.29}$$

and has the closed form solution using Rogridues rotation formula

$$\exp([\boldsymbol{\omega}]_{\mathrm{x}}) = I_3 + \frac{\sin\theta}{\theta}[\boldsymbol{\omega}]_{\mathrm{x}} + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_{\mathrm{x}}^2 \tag{2.30}$$

The exponential map yields a rotation by $\theta$ radians around the axis given by $\boldsymbol{\omega}$, and $\theta = \|\boldsymbol{\omega}\|$. The inverse of the exponential map is the logarithm map that maps an element $R$ in $\mathrm{SO}(3)$ to an element in the Lie algebra $\mathrm{so}(3)$.

$$\theta = \arccos\left(\frac{\mathrm{tr}(R)-1}{2}\right) \tag{2.31}$$

where tr() is the trace of a matrix.

$$\ln(R) = \frac{\theta}{2\sin\theta}(R - R^{\top}) \tag{2.32}$$

The vector $\boldsymbol{\omega}$ is then taken as the unique off-diagonal elements of $\ln(R)$.

### 2.5.2 Special Euclidean group $\mathrm{SE}(3)$

The special Euclidean group $\mathrm{SE}(3)$ represents the group of rigid transformations in 3D space.

$$T = \begin{pmatrix} R & \boldsymbol{t} \\ \mathbf{0}^{\top} & 1 \end{pmatrix} \tag{2.33}$$

with $R \in \mathrm{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$.
The Lie algebra $\mathrm{se}(3)$ is the set of 4×4 matrices corresponding to differential translations and rotations. There are thus six generators of the algebra:

$$G_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad G_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad G_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$G_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad G_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad G_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

An element $\boldsymbol{\delta}$ of $\mathrm{se}(3)$ is then represented as multiples of the generators:

$$\boldsymbol{\delta} = v_1\, G1 + v_2\, G2 + v_3\, G3 + \omega_1\, G_4 + \omega_2\, G_5 + \omega_3\, G_6 = \begin{pmatrix} [\boldsymbol{\omega}]_{\mathrm{x}} & \boldsymbol{v} \\ \mathbf{0}^\top & 0 \end{pmatrix} \tag{2.34}$$

with $(\boldsymbol{v}\ \boldsymbol{\omega})^\top \in \mathbb{R}^6$.
The exponential map is the matrix exponential over a linear combination of the generators:

$$\exp : \mathrm{se}(3) \mapsto \mathrm{SE}(3) \tag{2.35}$$

and has the closed form solution

$$\exp(\boldsymbol{\delta}) = \exp \begin{pmatrix} [\boldsymbol{\omega}]_{\mathrm{x}} & \boldsymbol{v} \\ \mathbf{0}^\top & 0 \end{pmatrix} = \begin{pmatrix} \exp([\boldsymbol{\omega}]_{\mathrm{x}}) & V\boldsymbol{v} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tag{2.36}$$

where $V = I_3 + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_{\mathrm{x}} + \frac{\theta-\sin\theta}{\theta^3}[\boldsymbol{\omega}]_{\mathrm{x}}^2$.
The logarithm map of $\mathrm{SE}(3)$ is then

$$t = V^{-1}v \tag{2.37}$$

and the vector $\boldsymbol{\omega}$ is again taken as the unique off-diagonal elements of $\ln(R)$ as in (2.32).

### 2.5.3 Optimisation on $\mathrm{SE}(3)$

One is now able to solve the $\mathrm{SE}(3)$ SLAM optimisation problem. The SLAM variables are normally the camera/robot poses and point positions in a classical SLAM problem. A camera pose lies in the non-Euclidean $\mathrm{SE}(3)$ space, therefore, one needs to utilise the Lie algebra explained in the previous section. The update is still denoted as $\boldsymbol{\delta}$ around the current variable $\mathbf{x}$. The variable update formula now changes to a general form:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta} \implies \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \boxplus \boldsymbol{\delta} \tag{2.38}$$

More specifically, for an $\mathrm{SE}(3)$ variable $\mathbf{x}$, $\boldsymbol{\delta}$ is in the tangent space $\mathrm{se}(3)$, and the update rule becomes:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \boxplus \boldsymbol{\delta} \implies \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \,.\, \exp(\boldsymbol{\delta}) \tag{2.39}$$

Similarly, the Jacobian of a cost function $\mathrm{e}$ also needs to extend to the general form:

$$\mathrm{J}_{\mathbf{e}}(\mathbf{x}) = \frac{\partial \mathbf{e}(\mathbf{x} \boxplus \boldsymbol{\delta})}{\partial \boldsymbol{\delta}}|_{\boldsymbol{\delta}=0} \tag{2.40}$$

The update rule in (2.39) and Jacobian general form in (2.40) are at the basis of SLAM optimisation on non-Euclidean $\mathrm{SE}(3)$ space that will be used in the work presented in this thesis to perform the optimisation later described in section 3.2.2.1.

It is worth mentioning that the above-presented optimisation methods represent the *back-end* component of a visual SLAM system, and will require a *front-end* component to abstract the sensor data into models usable for the optimisation. More details on the SLAM front and back-end components will be discussed in the following chapter.

Chapter 3

# Background And Overview

Our work lies at the intersection of geometric classical SLAM, motion and semantics as shown in Fig. 3.1.

## 3.1 Literature review

We first review the different categories of the classic geometric visual SLAM. For completeness and as an attempt to cover all the related work, we then review the semantic scene understanding literature, followed by the motion detection and tracking literature. We then focus on the intersection of geometry and semantics, and discuss the use of structural information and layout understanding in SLAM. Semantic SLAM algorithms are then covered and finally, the most relevant dynamic SLAM literature representing the intersection of geometry and motion is discussed.

### 3.1.1 Geometric classical SLAM

Based on the formulation, approaches to solve the SLAM problem in the literature can be either classified as filtering or optimisation-based methods. Based on the measurement functions defined, visual SLAM can also be classified as feature based (indirect) or direct approaches.

**Filtering vs. Optimisation-based SLAM** Filtering approaches model the problem as an online state estimation where the state of the system consists of the current robot pose and the map. The estimate is augmented and refined by incorporating the new measurements as they become available. The earliest works on SLAM focused on the Bayesian filtering approaches and were based on the extended Kalman filter (EKF) approach [41, 35, 42]. Other techniques

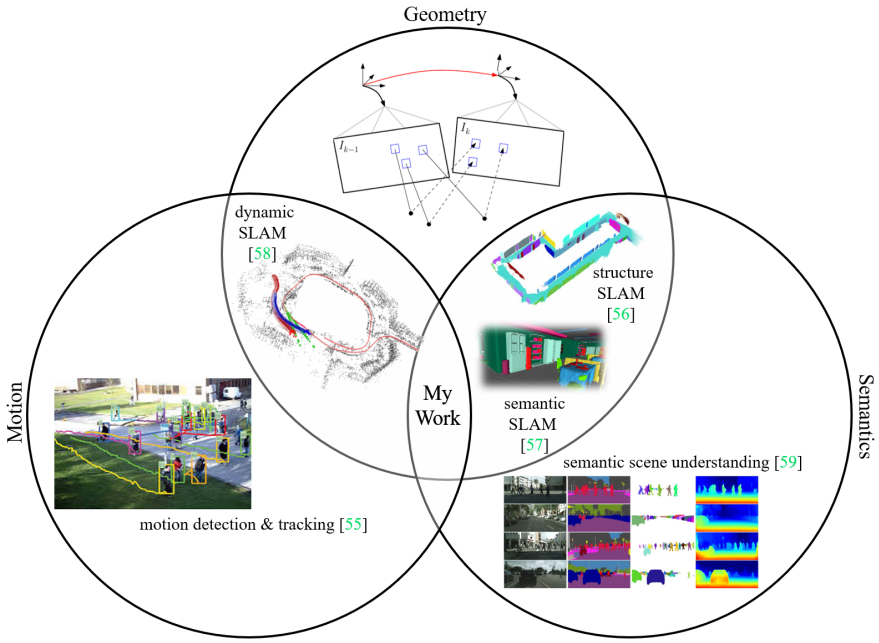



Figure 3.1: **Overview of the related work to this thesis.** Our work lies at the intersection of geometric classical SLAM, motion, and semantics. Note that papers cited in this figure are only for illustration and do not represent the full literature in each category.

that fall into the filtering category include particle filters [27, 43, 44], information filters [45, 46] or unscented Kalman filters (UKF) [47, 48]. To highlight their incremental nature, the filtering approaches are usually referred to as online SLAM methods. Filtering approaches estimate and continuously update a joint probability distribution of the map and robot poses. They comprise two steps: a prediction step to propagate the probability distribution, and an update step to correct the distribution based on measurements.
Although filtering approaches are starting to regain popularity with the advances in the field of event cameras, high frequency sensors and the use of direct methods, however, it has been shown that filtering approaches tend to have large drift in the long term, and are inconsistent when applied to the inherently non-linear SLAM problem [49] when the problem size grows, due to the accumulation of linearisation errors.

Optimisation-based approaches, conversely, estimate the full trajectory of the robot from the full set of measurements [50, 32, 51]. They jointly optimise all the camera poses and map points, which is also referred to as bundle adjustment (BA) [52] or smoothing [32] techniques. These approaches address the so-called full SLAM problem, and they typically rely on non-linear least-squares error minimisation techniques and can be solved by Gauss-Newton or Levenberg-Marquardt algorithms using available libraries such as iSAM [53] or g2o [54].

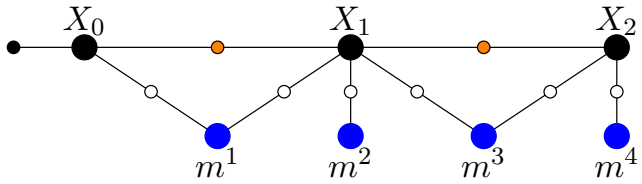
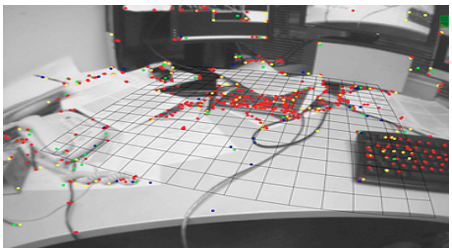



Figure 3.2: **Factor graph representation of a SLAM problem.** Larger circular nodes represent random variables; robot/camera poses $X_k$ and point features $m^i$. Factors, represented as smaller circles here, correspond to measurement functions. Factors can be unary such as a pose-prior factor, binary such as pose-pose and pose-point factors or n-ary.

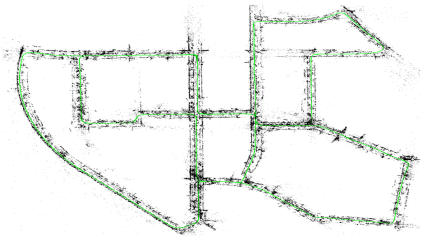
One intuitive way of formulating SLAM is to use a graph representation. Lu and Milios [50] first proposed the graph-based formulation of the SLAM problem in 1997 where they refine the map by globally optimising the system of equations to reduce the error introduced by constraints. Solving a graph-based SLAM problem involves constructing a graph whose nodes represent random variables; robot poses and/or landmark positions and in which an edge between two nodes encodes functions of those variables, typically a sensor measurement that constrains the connected nodes. Once such a graph is constructed, the goal is to find a configuration of the nodes that is maximally consistent with the measurements [55]. Approaching SLAM as a non-linear optimisation on graphs has been shown to offer very efficient solutions to moderate scale SLAM applications [54, 56]. Factor graphs [37], such as the one in Fig.3.2, are graphical models that have been used for representing the SLAM problem [32, 53, 57] as a graph-based problem. This is due to the fact that, in factor graphs, the functions are made explicit and such a bipartite graph is directly connected to the solutions of the optimisation problem [57].
However the memory and computational requirements can easily go unbound in large scale environments, therefore in practice, the concept of key-frames; only a subset of camera poses, has been introduced and used in the optimisation [58].
A comparison of filtering and optimisation-based approaches can be found in [59].

**Feature-based vs. Direct SLAM** Featured based or indirect methods pre-process the raw sensor measurements into intermediate representations such as feature points, lines or planes to be used in the SLAM estimation as landmarks. The optimisation then only depends on the extracted features and a geometric cost is normally used. These approaches greatly simplify the raw high dimensional image measurement to the low-dimensional geometry features. Different types of features have been considered; Features from Accelerated Segment Test (FAST [60]) corners in PTAM [58], Oriented FAST and rotated BRIEF (ORB [61]) features in ORB SLAM [62], Scale Invariant Feature Transform (SIFT [63]) in FOVIS [64] and Speeded Up Robust Features (SURF [65]) in LIBVISO [66]. In addition to points, other primitives such as lines and

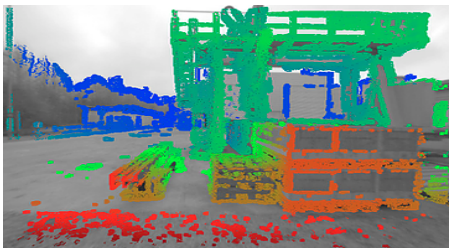
planes [11, 12, 67, 68] have been considered in SLAM.



PTAM indirect sparse map. [58]



ORB-SLAM indirect sparse map. [62]



LSD-SLAM direct semi-dense map. [69]

DTAM direct dense map. [70]

Figure 3.3: **Different feature-based and direct SLAM systems.**

The second category is direct methods which skip the pre-processing feature extraction step and directly utilise the raw sensor input such as image pixel intensities. Therefore the optimisation cost used is the photometric (intensity) error. Compared to the geometric error, the photometric error is highly non-linear and non-convex, which results in a small convergence basin and requires good initialisation of map and camera poses. DTAM [70] is the first real-time dense and direct SLAM method and relies on GPU acceleration. Recently, Engel *et al.* proposed two real-time SLAM systems running on CPU: LSD-SLAM [69] and DSO [71]. Different from DTAM, LSD and DSO only utilise some high gradient pixels to speed up the process.

### 3.1.2 Semantic scene understanding

Scene understanding is concerned with the problem of recognising, analysing and understanding the objects and surfaces in context with respect to the 3D structure of the scene, its layout, and their spatial, functional, and semantic relationships. Scene understanding includes object detection, scene classification and semantic segmentation. The object detection problem will later be described as part of the motion tracking problem.

**Scene classification** Traditional scene classification methods were based on hand-crafted fea-

tures such as SIFT [72] and SURF. A Spatial Pyramid Matching (SPM) model was proposed in [73]. In recent years, deep learning techniques have made a huge progress in image classification [74, 75], object detection [76, 77], and other computer vision related tasks. A scene classification model was presented in [78] based on deep CNN features.

**Semantic segmentation** Traditional methods for semantic segmentation were based on handcrafted and carefully engineered features such as SIFT [79] or Histogram of Oriented Gradients (HoG) [80, 81], along with flat classifiers such as Boosting [82], Support Vector Machine (SVM) [83, 84], and Random Decision Forests [85, 86]. Substantial improvements have been achieved by incorporating richer information from context [87] and structured prediction techniques [88, 89, 90, 91]. Image segmentation has often been modelled as an energy minimisation problem. Conditional Random Fields (CRF) are rich probabilistic graphical models; pixels of an image can be viewed as variable nodes in a CRF. Krähenbühl and Koltun [89] presented an efficient, approximate inference algorithm for fully-connected CRFs where pairwise edge potentials are defined by a linear combination of Gaussian kernels. Successful deep neural network architectures for image classification such as AlexNet [92], VGG net [93], GoogLeNet [74], and ResNet [75] are a natural precursor to, and often a direct part of semantic segmentation architectures [94]. All state-of-the-art models [5, 95, 96] involve convolutional neural networks. Modern semantic segmentation architectures can be categorised in two main categories. The first contains architectures primarily influenced by the Fully Convolutional Network (FCN [97]), and rely on an encoder-decoder architecture [98]. The second division contains architectures that are also influenced by the FCN architecture but additionally employ dilated convolutions [99, 96].

### 3.1.3 Motion detection and tracking

The problems of motion detection and multiple object tracking (MOT) have attracted a lot of attention over the years. Challenges in MOT can be mainly grouped into the motion and appearance information, the data association problem, and the object detection [100].

**Motion segmentation** Significant research has been made in motion segmentation in the past two decades. Current state-of-the-art algorithms seek to combine methods based on affine assumption [101] and those based on epipolar geometry [102] into a single framework that leverages their advantages [103, 104]. In a recent approach, a multi-frame spectral clustering framework is introduced by [105] with joint integration of an affine model, a homography model and a fundamental matrix. Making use of semantic information has proven to help deal with the issues of degenerate motions and partial occlusions increasing the motion segmentation accuracy [106, 107].

**Motion tracking** Multiple object tracking algorithms have moved over the last few years from classical inference/filtering based [108, 109] to data-driven (deep learning) approaches [110, 111]. One of the earliest works in motion estimation use the Kalman filter [112] to predict the state of the target at the current time step given its state at the previous time step. Recently, and with the development of deep learning, motion models based on Recurrent Neural Networks (RNN) and Long Short Term Memory (LSTM) [110, 113] have been developed. The state-of-the-art online multi-object tracking STAM-MOT [114] applies spatial and temporal attention map to handle the partial occlusion problem in tracking. To find the optimal location of objects, the proposed algorithm employs the dense searching strategies, which are utilised commonly in tracking single object.
Optical flow has been an effective way to describe motion between image frames within a video. The traditional Lucas–Kanade algorithm [115] has been widely used for sparse optical flow estimation. With the rapid advances of convolutional neural networks (CNN), new methods of estimating the optical flow have been proposed. Fischer and Ilg *et al.* successively propose FlowNet [116] and FlowNet2.0 [117], which can be used for dense optical flow estimation using a well-trained encoder-decoder network. More recently, Sun *et al.* propose PWC-Net [118], an optical flow network fusing pyramidal processing, warping, and a cost volume, achieving better and faster optical flow estimation.

**Appearance feature** The appearance feature is a discriminative way to represent an object, which is essential for MOT in cluttered scenes. Colour histograms [119, 120] and hand-crafted features [121, 122] were commonly used as objects appearance descriptors in earlier works. With the popularity of deep neural networks, deep feature-based appearance representations are increasingly used to enhance the discriminative power of appearance features [100]. The work by Wojke *et al.* [123] employs a wide residual network to extract object features and measure their similarity using cosine distances. The network architecture in [124] is used by Chen *et al.* [125] to extract object features, which use Euclidean distance as a similarity metric. Siamese networks have also been used in [126] to learn discriminative features of detected objects.

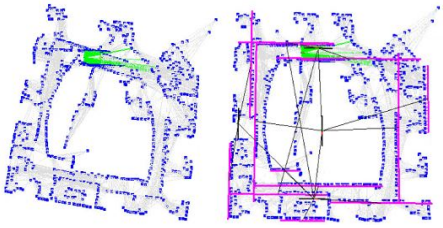
**Object detection** As a part of the commonly-used tracking-by-detection paradigm, object detection has a great impact on the performance of trackers that fall under this category. In earlier times, object detection based on discriminatively trained part-based models [127] played an important role in MOT. Recently, deep learning based object detection methods have far surpassed the traditional methods. Faster-RCNN [128] has become one of the most commonly used object detectors. Some more recent object detection algorithms include [129, 130, 131, 132, 133].

**Data association** Data association is a key step in tracking-by-detection based MOT methods, and represents the second stage of tracking-by-detection algorithms. Many offline MOT methods [134, 135, 136] treat data associations as a graph-based optimisation problem. The Hungarian algorithm [137] is also a commonly used data association optimisation method. More recently, a differentiable operator to build a deep Hungarian network is introduced in [138].

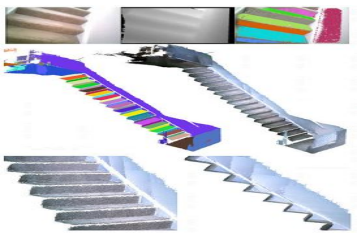
### 3.1.4 Structural information in SLAM

SLAM algorithms are normally applied to scenes that contain significant structure that could be exploited. Integrating structure of the environment into the estimation problem has shown to improve the quality of the estimation [11, 139, 140].

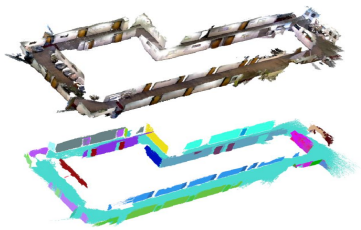
Lee *et al.* [140]



De La Puente *et al.* [67]



Michael Kaess [11]



Hsiao *et al.* [68]

Figure 3.4: **Different structure-aware SLAM systems.**

One of the earliest works that consider adding geometrical constraints to improve the quality of the reconstruction is presented by Szeliski and Torr [14] where they hallucinate additional point matches based on image homographies which are either given directly or computed between a collection of *a-priori* known co-planar points. Other approaches to explicitly incorporate planes in the estimation include the work by Weingarten and Siegwart [141] where they use 3D laser data and odometry derived from a 2D laser into an EKF formulation. Servant *et al.* [142] also use an EKF formulation starting with partial knowledge and tracking a monocular camera. In their work [143, 144], an EKF is also used for monocular mapping. The main

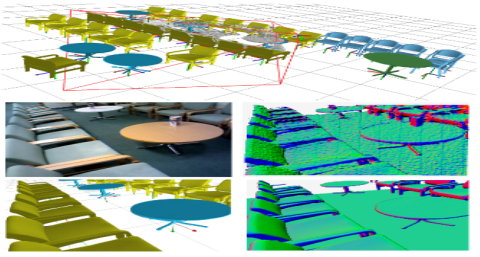
problem of using an EKF formulation is the computational cost caused by maintaining the dense covariance matrix, which limits the application to a small number of planes. [11] Zucchelli *et al.* [139] showed how linear constraints among feature points (e.g. co-linearity and co-planarity) can be incorporated in the minimisation process to improve the structure from motion estimates from optical flow using least-squares minimisation of the differential epipolar constraints. Several plane parameterisations are used in the literature. Lee *et al.* [140] use a graph formulation in combination with a spherical parameterisation to extract planes and use them to correct odometry between consecutive frames. Another approach presented by Triebel and Burgard [145], extracts constraints from 3D range scans and uses them for pose estimation in a graph-based SLAM framework. Another common approach to represent a plane by four parameters is to use its normal and distance to the origin. Trevor *et al.* [146] uses this over-parameterised representation for a smoothing solution. The over-parameterised formulation is also used by [147] for real-time mapping, combining both mapping of points and planes. Most recently, De la Puente and Rodriguez-Losada [67] presented an approach to perform landmark SLAM optimisation, along with different level structure detection (points, segments, lines and circles) in an Expectation-Maximisation algorithm, where only the last robot pose and the features belong to the graph. Kaess [11] also shows how to formulate SLAM that directly estimates infinite planes instead of the 3D landmarks in the environment. It uses homogeneous plane parameterisation with a corresponding minimal representation for the optimisation which is suitable for use with non-linear least-squares incremental solvers. Hsiao *et al.* [148] extend on [11] and use the same plane parameterisation into a dense planar-inertial SLAM with added structural constraints to achieve better performance. Similar to [11], the work by Hsiao *et al.* only consider planar surfaces and their constraints, and does not estimate for non-planar points. A recent work by Geneva *et al.* [149] use a closest point (CP) representation which is defined by the closest point on the plane to the origin of a given reference frame, which captures the geometric information of the plane, and minimally represents the plane with an additive error state operation, yielding numerical advantages. It is important to note that the last two mentioned works appeared after our work in the area of structural SLAM, however we include them here for the sake of completeness of literature review.
Our work on the other hand, estimates for both, structure points and planes in graph-based SLAM formulation, this allows for 3D reconstruction of more realistic environments where planar and non-planar structures coexist.

### 3.1.5 Semantic information in SLAM

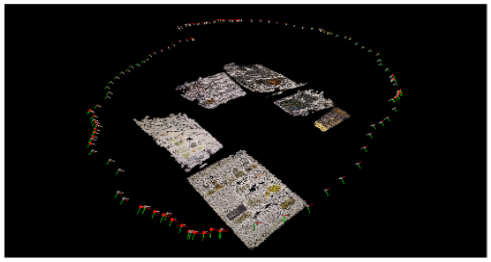
While the system presented in this thesis is a feature-based SLAM system, it still exploits various semantic information to be able to consistently and robustly map the environment and estimate

the robot poses. It is then important to review the semantic information literature in SLAM. This can be categorised into decoupled and coupled approaches.

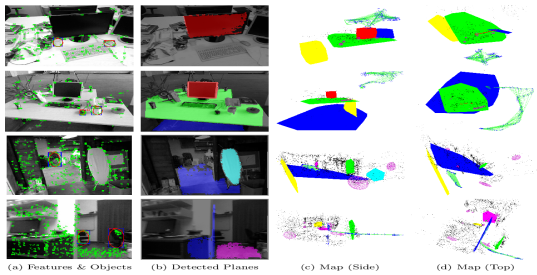
**Decoupled approaches** The decoupled semantic SLAM approaches first build a point cloud map of the environment then label the points, or detect objects within the map. Sengupta *et al.* [150] directly transfer the 2D segmentation label to 3D map through pixel back-projection to achive 3D semantic labelling. Graphical models can be used to improve the smoothness in a post optimisation step [151, 152, 153]. [154, 155] show how multi-view SLAM can be used to improve the object detection performance. Similarly, planes or super-pixels are used in [156, 157, 158] to improve the accuracy of dense mapping in low-texture areas. The performance of these approaches is dependent on the quality of the SLAM returned map.



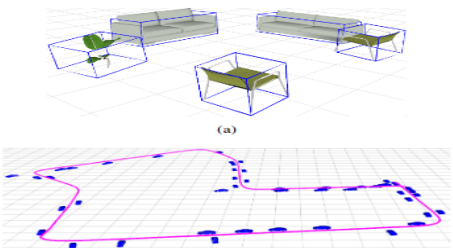
SLAM++ [8]



Galvez *et al.* [159]




Hosseinzadeh *et al.* [7]




CubeSLAM [13]

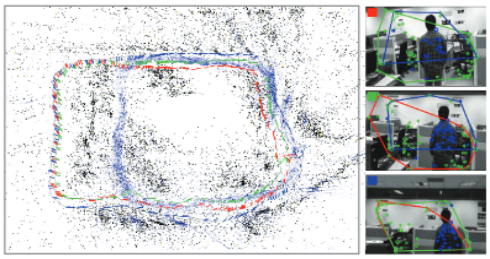
Figure 3.5: **Different semantic SLAM systems.**

**Coupled approaches** The coupled approaches define objects and geometric entities as SLAM landmarks and jointly optimise their parameters along with the camera poses. Compared to points, objects and planes provide richer, longer-range geometry and scale constraints. Bao *et al.* [160] proposed to jointly infer high-level semantic scene components (regions and objects) along with points into a single inference problem. [8] proposed "SLAM++"; a real-time object SLAM system using RGB-D cameras and an a-priori created database of 3D-models of every

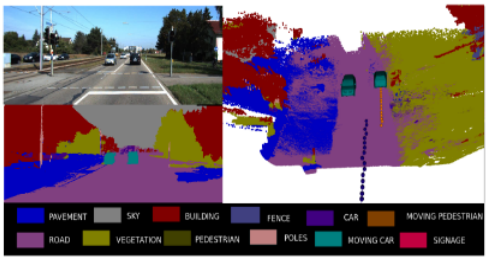
object in the scene. Similarly, [159] proposed a real time monocular object SLAM that relies on prior object models. Objects have also been used to correct the scale drift of monocular SLAM [161, 162]. [163] solved multi-view 3D ellipsoid estimation exploiting object constraints on the size of the object shape and QuadricSLAM [164] extended it to an online SLAM system with no prior models. Lee [165] jointly estimates the layout and performs a global multi-view point cloud registration iteratively to reduce the RGB-D mapping drift. Hsiao *et al.* [68] also use planes to reduce pose drift and create dense mapping for large scale indoor buildings in a key-frame based planar SLAM. [166] proposed "Fusion++"; an online volumetric object-level SLAM using RGB-D camera without objects prior shape models. More recently, [7] proposed a structure-aware SLAM system that jointly estimates points, planes and objects represented as quadrics and introduced point-plane, plane-plane, and plane-object (supporting) relations to constraint the problem. A very recent work in [13] presents a method for single image $3D$ cuboid detection, and multi-view object SLAM from a monocular camera.

### 3.1.6 Dynamic motion information in SLAM

Establishing the spatial and temporal relationships between a robot, stationary and moving objects in a scene serves as a basis for scene understanding and the problems of simultaneous localisation, mapping and moving object tracking are mutually beneficial [167]. In the SLAM community, however, information associated with stationary objects is considered positive, while information drawn from moving objects is seen as degrading the algorithm performance. State-of-the-art SLAM systems either treat data from moving objects as outliers [168, 169, 170, 171, 172] or track them separately using multi-target tracking [173, 174, 175, 176], and very few aim to utilise information from static and dynamic objects into a single framework to improve the accuracy of the estimation [177, 178, 13].



CoSLAM [179]




Reddy *et al.* [180]

MVO [178]


CubeSLAM [13]

Figure 3.6: **Different dynamic SLAM systems.**

One of the earliest works in the area of SLAM in dynamic environments is presented by Hahnel *et al.* [169] who use an Expectation-Maximisation (EM) algorithm to update the probabilistic estimate about which measurements correspond to a static/dynamic object, and remove them from the estimation when they correspond to a dynamic object. Alcantarilla *et al.* [181] introduce dense scene flow for dynamic objects detection, and show improved localisation and mapping results by removing "erroneous" measurements on dynamic objects from the estimation. Tan *et al.* [182] propose an online key-frame update that reliably detects changed features in terms of appearance and structure and discards them if necessary.

Wang *et al.* [167] developed a theory for performing SLAM with Moving Objects Tracking (SLAMMOT). They first presented a SLAM algorithm with generalised objects, which computes the joint posterior over all objects and the robot, an approach that is computationally demanding and generally infeasible as stated by the authors. In the latest version of their SLAM with detection and tracking of moving objects, the estimation problem is decomposed into two separate estimators (moving and stationary objects) to make it feasible to update both filters in real time. Kundu *et al.* [176] tackle the SLAM problem with dynamic objects by solving the problems of Structure from Motion (SfM) and tracking of moving objects in parallel, and unifying the output of the system into a 3D dynamic map of the scene containing the structure and the trajectory of both static and moving objects. Reddy *et al.* [180] use optical flow and depth to compute semantic motion segmentation. They isolate static objects from moving objects and reconstruct them independently, before using semantic constraints to improve the 3D reconstruction.

Bibby and Reid's SLAMIDE [177] estimates the state of 3D features (stationary or dynamic) with a generalised EM algorithm where they use reversible data association to include dynamic objects in a single framework SLAM. A multi-camera SLAM system is proposed by Zhou and Tan [179], that is able to track multiple cameras, as well as reconstruct the 3D position of both static background and moving foreground points. Their system leverages the idea that points

on moving objects share information about relative camera poses at the same time step to estimate all camera poses simultaneously. Judd *et al.* [178] estimate the full $\mathrm{SE}(3)$ motion of both the camera and rigid objects in the scene by applying a multi-motion visual odometry (MVO) multi-model fitting technique. Although this approach does not require prior knowledge of the environment or object 3D models, they only show results on experiments performed on their own lab-environment indoor collected dataset, with no evaluation on any other existing datasets. A very recent work by Yang and Scherer [13] presents a method for single image 3D cuboid detection, and multi-view object SLAM for both static and dynamic environments. Their main interest, however, is the camera pose and object detection accuracy and they provide no evaluation of the accuracy of their object pose estimation.

### 3.1.7 Motion and semantics

Finally, recent approaches have tried to solve the problems of multi-object tracking and semantic scene understanding jointly such as the work presented in Track-RCNN [183] which combines multi-object tracking and segmentation. Only mentioned here for the sake of completeness, this body of literature, however, is not of interest or relevance to the work presented in this thesis, and thus is not covered in here.

## 3.2 Our SLAM system

This section describes our vision for a SLAM system that integrates meta information to achieve accuracy, consistency and robustness. It is worth mentioning that all components were not evaluated together as a complete system, but rather the appropriate components were picked as sub-system components for different applications as will be discussed later in chapters 4, 5 and 6.

Popularity of SLAM has been increasing over the last few decades with the advances in a number of related research fields. At the lower level, SLAM naturally intersects other research fields such as computer vision and signal processing; at the higher-level, SLAM is an appealing mix of geometry, graph theory, optimisation, and probabilistic estimation [184].
A typical SLAM system pipeline, as shown in Fig.3.7, consists of two main components.
A **front-end** component that abstracts sensor data into models that are usable for estimation, and a **back-end** component that performs inference on the abstracted data produced by the front-end. [184].

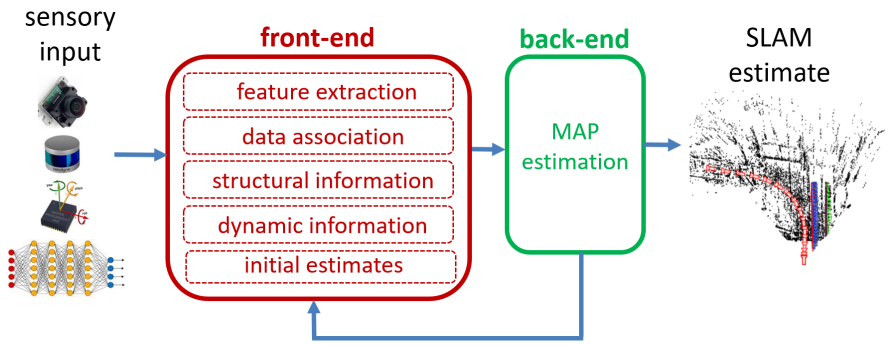



Figure 3.7: **Our vision for a SLAM system architecture.** Vision of front and back-end components in our SLAM system pipeline. Modified version of an original figure that initially appeared in [184].

### 3.2.1 Front-end

In practical robotics applications, it might be (computationally) unfeasible to write directly the sensor measurements as an analytical function of the state, as required in a MAP estimation in the back-end [184]. For instance, if the raw sensor data is an image, as in visual SLAM systems, it might be hard to express the intensity of each pixel in the image as a function of the SLAM state. The same happens with other sensors such as laser scanners, Inertial Measurement Units (IMU), etc.

The reason is fundamentally associated to the representation of the environment. It is almost impossible to design a sufficiently general, yet tractable representation of the environment; and even if such representation exists, it would be difficult to write an analytic function that connects the measurements to the parameters of such a representation.

SLAM front-ends are then sensor and representation-dependent. Before the SLAM back-end, it is then common to have a module, the front-end, that extracts relevant *information* from the sensor data and feeds them into the back-end. In an indirect (feature-based) vision-based SLAM for example, the front-end extracts the pixel location of key points in the environment which are now easy to model within the back-end.

The front-end is also in charge of associating each measurement to a specific landmark in the environment, or more generally to a subset of unknown variables. The data association module in the front-end includes a short-term data association block and a long-term one [184]. Short-term data association is responsible for associating corresponding features in consecutive sensor measurements; for instance, the fact that two pixel measurements in consecutive frames correspond to the same 3D point. On the other hand, long-term data association or loop closure is in charge of associating new measurements to older landmarks with the help of feedback information from the back-end to support loop closure detection and validation.

Finally, the front-end might also provide an initial guess of the random variables in the non-linear

optimisation. For instance, in feature-based monocular SLAM, the front-end usually takes care of the landmark initialisation, by triangulating the position of the landmark from multiple views. In summary, the pre-processing that happens in the front-end is sensor dependent, since the notion of feature changes depending on the input data [184]. And a SLAM front-end is responsible of: 1. extracting and abstracting sensor data into models for inference, 2. short-term data association, 3. long-term data association, and 4. providing an initial guess of random variables to be optimised.

It is worth noting that the recent advances in the fields of perception and scene understanding open wide doors to integrating more information into the estimation problem.
The recent advancement in front-ends made them capable of extending the classical feature detection, to include higher-level entities, such as lines, segments, circles, planes, or even objects. It also made them capable of providing information about objects in the environment, their shape, colour, texture, depth, semantics, and dynamics. This information can be extended to include meta-structural information about the environment, e.g planarity and orthogonality of buildings in a city-like environment. Recent development in the area of machine learning, and neural networks made it possible to detect and classify objects in a scene, making it appealing to integrate semantics and SLAM into a single framework. Motion segmentation and moving object tracking, results of deep learning techniques, have recently steered researchers interests into integrating the dynamics of the environment into SLAM systems.

In the following, we describe the off-the-shelf components of the literature that constitute parts of the front end of our SLAM system.

#### 3.2.1.1 Feature extraction and tracking

In feature-based visual SLAM systems, a core component is the feature extraction and tracking. And despite the recent advances in deep learning techniques to segment higher-level entities, and attempts to include those as environment landmarks in SLAM, point measurements remain to be the most commonly used in visual SLAM systems due to the fact that they are easily obtained from visual data and their integration into a SLAM estimation is fairly easy. Feature extraction and tracking has thus been an attractive research topic since the 90s and continues to be the interest of a large community of computer vision and robotics researchers.

**Descriptor matching** A classical approach to track image features is to first describe the extracted features [63, 65, 60, 61] using a *descriptor* and match features across images based on a Euclidean distance in the feature descriptor vector space.
Our planar SLAM system described in chapter 4 of this thesis uses SURF [65] features to detect, describe and match key points across frames. Plane extraction and association will be described

later.

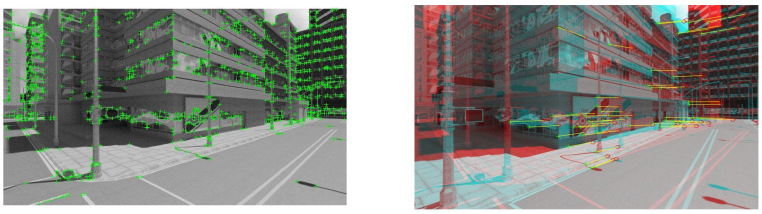


Figure 3.8: **Feature extraction and tracking on a synthetic city dataset [185]**

**Optical flow** Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene [186]. Optical flow can also be defined according to Horn as the distribution of apparent velocities of movement of brightness pattern in an image [187]. The concept of optical flow was first introduced by the American psychologist James J. Gibson in the 1940s to describe the visual stimulus provided to animals moving through the world [188]. Motion estimation has developed as a major application of optical flow research. Optical flow is used in areas such as object detection and tracking [189, 190], image dominant plane extraction [191], robot navigation and visual odometry [192, 193].
Estimation of the optical flow relies on the *brightness consistency assumption* given by:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{3.1}$$

which states that the intensity $I$ of a pixel $(x, y)$ at time $t$ is equal to the intensity of the moved pixel at time $t + \Delta t$. Assuming the movement to be small, the image constraint at $I(x, y, t)$ can be approximated using the first-order Taylor expansion as:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \tag{3.2}$$

It follows that

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{3.3}$$

or

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} = 0 \tag{3.4}$$

which results in

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y = -\frac{\partial I}{\partial t} \tag{3.5}$$

where $v_x$ , $v_y$ are the $x$ and $y$ components of the velocity or optical flow at $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$,

and $\frac{\partial I}{\partial t}$ are the derivatives of the image at $(x, y, t)$ in the corresponding directions.
Various methods are present in the literature to estimate optical flow from images, of which the most widely used are the Lucas-Kanade algorithm [115] and the Horn–Schunck method [187].

Figure 3.9: **Optical flow estimation on KITTI dataset [194] using PWC-Net [118].**

Recently, a number of deep learning techniques have established methods to achieve per-pixel dense optical flow estimation [117, 118]. In our dynamic SLAM system described in chapter 5, PWC-Net [118] was used to ensure long tracks of detected features, especially on moving objects, which are crucial for our feature-based dynamic SLAM system.

#### 3.2.1.2 Structural information

The first piece of the work presented in this thesis deals with SLAM in structured environments; environments with abundant structure to be exploited. We focus on planar surfaces and integrate them into the estimation framework as latent variables that are unobservable but constrained by the observation of environment points and relations to other planar surfaces (parallelism and orthogonality). Our system's front-end is able to extract planes from a single image, associate them across frames, and infer relations between the extracted planes across a sequence of images.

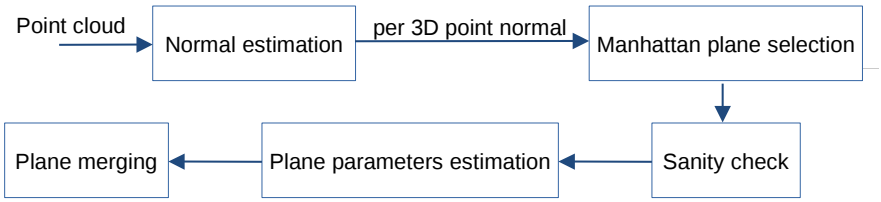



Figure 3.10: **Proposed planar surfaces extraction and association pipeline.**

**Planar surfaces extraction and association** Planar surfaces extraction and association is summarised in Fig. 3.10. First, surface normals are estimated using the "Surface Reconstruction from Unorganised Points" algorithm [195]. This is performed on point clouds to find the normal of every 3D landmark point in the environment to achieve a per point normal estimation, using a

neighbourhood of $n$-points. This parameter was set to six in our planar SLAM system described in chapter 4. This is followed by a Manhattan world selection, to only include surface normals orthogonal to the ground plane and normals to planes orthogonal to the ground plane. A plane sanity check to only keep planes with more than six points defining the plane is performed. Finally plane parameters are estimated that best fit the set of points on the plane by minimising the least-squares of the normal distance to the plane, and a plane merging algorithm based on thresholding of angles between plane normals vectors is performed.

#### 3.2.1.3 Semantic information

Although our SLAM systems presented in chapters 5 and 6 are feature-based, they highly exploit image-based semantic information to achieve robustness, accuracy and consistency.
Our system exploit semantic information in three main ways; 1. instance-level object segmentation, and 2. single image depth extraction, besides 3. optical flow estimation for feature tracking described above. The front-end of our system is then able to provide semantic information to be integrated into the back-end estimation.

**Instance level object segmentation** We exploit image-based semantic information to achieve

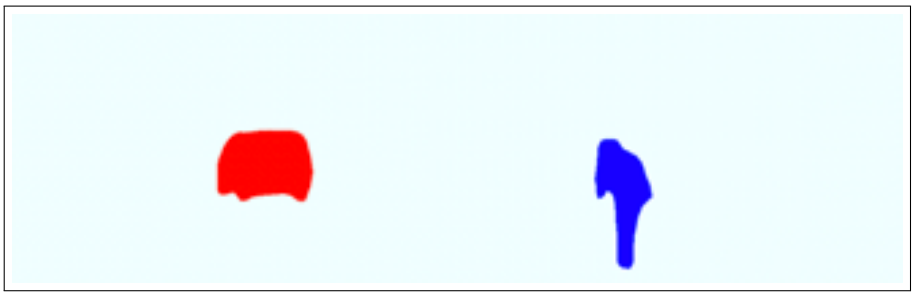


Figure 3.11: **Object segmentation on KITTI dataset [194] using MASK-RCNN [5].**

an object-aware SLAM system, that is able to include and estimate dynamics on the environment into a single SLAM framework. Instance-level object segmentation is performed to achieve per pixel segmentation of objects present in the scene. We employ a learning-based instance-level object segmentation, MASK-RCNN [5] to generate object segmentation masks. Object semantic segmentation can provide significantly important cues for dynamic SLAM. Object classes provide a great classification of potentially dynamic (moveable) objects such as vehicles, cyclists, people and forever static objects such as street signs and traffic lights in an urban environment. In fact, we rely on object segmentation to include every potentially moving object in the scene as a dynamic object and estimate for its motion, without the need to perform any motion segmen-

tation as in chapter 5. Moreover, instance segmentation offers object masks that ensure robust tracking of points on objects by sampling points within the object mask and re-sampling new points if the number of tracked points on an object falls below a certain predefined threshold, as compared to feature detection on objects (such as cars, trucks, buses, etc.) that normally consist of featureless surfaces.

**Single image depth estimation** In an attempt to fully exploit image-based semantic informa-

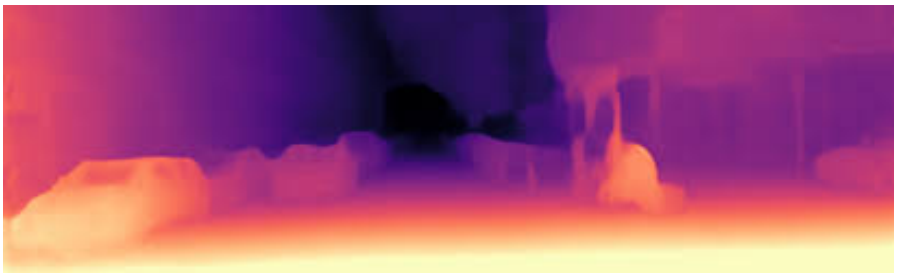


Figure 3.12: **Single image depth estimation on KITTI dataset [194] using Monodepth2 [196].**

tion, we provide a "monocular" version of our dynamic SLAM system. More specifically, our system uses RGB input images only and first performs a single image depth estimation using Monodepth2 [196]; a state-of-the-art off-the-shelf learned approach.

#### 3.2.1.4 Dynamic information

Finally, our front-end is able to provide dynamic information to be integrated in the back-end estimation. Namely, identifying and tracking moving object allows reduce the problem size and better constraint the object motion estimation.

**Motion segmentation** Despite the ability of our system described in chapter 6 to model and estimate the motion of every moveable object in the environment, dynamic object identification helps reduce computational cost of the proposed system. We perform a motion segmentation step to reduce the problem size, memory and computational complexity. The front-end is capable of identifying moving objects based on scene flow estimation. Specifically, after obtaining an initial camera pose ${}^0X_k$, the scene flow vector $f_k^i$ describing the motion of a 3D point ${}^0\boldsymbol{m}^i$ between frames $k-1$ and $k$, can be calculated as in [197]:

$$\boldsymbol{f}_k^i = {}^0\boldsymbol{m}_{k-1}^i - {}^0\boldsymbol{m}_k^i = {}^0\boldsymbol{m}_{k-1}^i - {}^0X_k \, {}^{X_k}\boldsymbol{m}_k^i \,. \tag{3.6}$$

Unlike optical flow, scene flow—ideally only caused by scene motion—can directly decide whether some structure is moving or not.

**Moving object tracking** Objects masks provided by instance-level object segmentation need to be tracked and associated across frames. We propose to use optical flow to associate point labels across frames. A point label is the same as the unique object identifier on which the point was sampled. We introduce and maintain a finite tracking label set $\mathcal{L} \subset \mathbb{N}$, where $l \in \mathcal{L}$ starts from $l = 1$ for the first detected moving object in the scene. The number of elements in $\mathcal{L}$ increases as more moving objects are being detected. Static objects and background are labelled with $l = 0$.

In summary, our system front-end is able to extract, and track features on static and dynamic objects, provide structural, semantic and dynamic information about the lay-out and objects in the environment and offer initial estimates of unknown random variables to be estimated.
Random variable initialisation provided by the front-end will be discussed in our planar and dynamic SLAM systems in chapters 4 and 5 respectively.

### 3.2.2 Back-end

Now that the front-end has provided the above-mentioned information, the role of the back-end module is to perform inference on the abstracted data provided by the front end [184]. We formulate the SLAM inference problem using a graph representation as described in section 3.1.1

#### 3.2.2.1 Graph-based SLAM problem formulation

In SLAM problems, the goal is to estimate the 3D structure of the environment and the camera poses that maximally satisfy a set of measurement constraints [198]. It has been shown in the SLAM literature [34, 22] that Gaussian noise models lead to computationally efficient solutions. Optimisation-based methods propose to formulate the SLAM problem as a non-linear least-squares (NLS) optimisation as explained in section 2.4.3. This is done over a set of variables; the camera/robot poses at different time steps, and the 3D points in the environment. Together these variables constitute the total system state of a classical landmark SLAM problem.
We consider two types of measurements, the odometry obtained by the robot's proprioceptive sensors or offered by the front-end performing visual odometry and the observations of the landmarks in the environment obtained by processing the images from an on-board camera.
The odometry model error $\mathbf{e}_k({}^0X_{k-1}, {}^0X_k)$ is defined as:

$$\mathbf{e}_k({}^0X_{k-1}, {}^0X_k) = ({}^0X_{k-1}^{-1}\ {}^0X_k)^{-1}\ {}^{X_{k-1}}_{k-1}T_k\,, \tag{3.7}$$

where $T = \{{}^{X_{k-1}}_{k-1}T_k \mid k \in \mathcal{T}\}$ is the odometry measurement set with ${}^{X_{k-1}}_{k-1}T_k \in \mathrm{SE}(3)$ and cardinality $n_o$, and with $\mathcal{T}$ the set of all time steps. The odometric factors are shown as orange

circles in Fig. 3.2.
Similarly, the 3D point measurement model error $\mathbf{e}_{i,k}({}^0X_k, {}^0\boldsymbol{m}_k^i)$ is defined as:

$$\mathbf{e}_{i,k}({}^0X_k, {}^0\boldsymbol{m}_k^i) = {}^0X_k^{-1}\,{}^0\mathbf{m}_k^i - \mathbf{z}_k^i\,, \tag{3.8}$$

where $\mathbf{z} = \{\mathbf{z}_k^i \mid i \in \mathcal{M}, k \in \mathcal{T}\}$ is the set of all 3D point measurements at all time steps, with $\mathbf{z}_k^i \in \mathbb{R}^3$ and cardinality $n_z$, and $\mathcal{M}$ the set of all landmarks. The 3D point measurement factors are shown as white circles in Fig. 3.2.
We parameterise the $\mathrm{SE}(3)$ camera pose by elements of the Lie-algebra $\mathbf{x}_k^\wedge \in \mathrm{se}(3)$ as:
${}^0X_k = \exp({}^0\mathbf{x}_k^\wedge)$ , and define $({}^0\mathbf{x}_k^\wedge)^\vee \triangleq {}^0\mathbf{x}_k \in \mathbb{R}^6$ with the *vee* operator a mapping from $\mathrm{se}(3)$ to $\mathbb{R}^6$, and the *wedge* operator as the inverse of the *vee* operator, and defines a mapping from $\mathbb{R}^6$ to $\mathrm{se}(3)$.
Let $\boldsymbol{\theta}_X = \{{}^0\mathbf{x}_k \mid k \in \mathcal{T}\}$ be the set of all camera poses, and $\boldsymbol{\theta}_M = \{{}^0\boldsymbol{m}_k^i | i \in \mathcal{M}, k \in \mathcal{T}\}$ be the set of all 3D points, with ${}^0\boldsymbol{m}_k^i \in \mathbb{R}^3$. Given $\boldsymbol{\theta} = \boldsymbol{\theta}_X \cup \boldsymbol{\theta}_M$ as all the nodes in the graph, and using the Lie-algebra parameterisation of $\mathrm{SE}(3)$ for $X$, the solution of the problem is obtained by minimising the sum of squared non-linear residuals as:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \Big\{ \sum_{k}^{n_o} \mathbf{e}_k^\top({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k)\, \Sigma_o^{-1}\, \mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k) + \sum_{i,k}^{n_z} \mathbf{e}_{i,k}^\top({}^0\mathbf{x}_k, {}^0\,\mathbf{m}_k^i)\, \Sigma_z^{-1}\, \mathbf{e}_{i,k}({}^0\mathbf{x}_k, {}^0\,\mathbf{m}_k^i)) \Big\}\,, \tag{3.9}$$

where $\Sigma_o$ is the odometry noise covariance matrix, and $\Sigma_z$ is the 3D point measurement noise covariance matrix. The error terms in (3.9) are calculated as follows: first the wedge operator is applied to each $\mathbb{R}^6$ element to map it to its $\mathrm{se}(3)$ element, then the exponential is applied to map it to its $\mathrm{SE}(3)$, errors are then computed as described in (3.7) and (3.8). The same procedure is followed when describing a minimisation of residuals for the rest of this document. Iterative non-linear optimisation methods such as Gauss-Newton (described in section 2.3.3) or Levenberg-Marquardt (described in section 2.3.4) can be used to find a solution that minimise (3.9). This formulation is often used in the SLAM literature [32, 53, 54, 198]. The graph formulation of the SLAM is highly intuitive and has the advantage of being able to incorporate several types of observations (odometry, GPS, IMU, sonar, laser scan registration, feature points, feature lines, etc.). Another advantage of this formulation is that it allows for efficient implementations of batch [199, 54] and incremental [53, 57, 200] NLS solvers.
In chapters 4, 5 and 6, we show how this graph formulation is used to incorporate meta information into the estimation problem, and how new factors are introduced to model the structural, semantic and dynamic information provided by the front-end. All back-end graph optimisation

is this thesis is solved in a batch fashion, unless otherwise indicated.

### 3.2.3 Estimation accuracy evaluation

The accuracy of the proposed systems in this thesis is evaluated by comparing the absolute trajectory translational error (ATE), the absolute trajectory rotational error (ARE), the absolute structure error (ASE), the relative trajectory translational error (RTE), the relative trajectory rotational error (RRE), and the relative structure error (RSE) calculated as follows:

$$\text{ATE} = \frac{1}{n_x}\sum_{k=1}^{n_x} \|tr(({}^0X_k{}^{-1})_{est}({}^0X_k)_{gt})\| \tag{3.10}$$

where $tr(.)$ returns the $\mathbb{R}^3$ translational component vector of an $\mathrm{SE}(3)$ element, and $n_x$ is the total number of robot poses.

$$\text{ARE} = \frac{1}{n_x}\sum_{k=1}^{n_x} \|rot(({}^0X_k{}^{-1})_{est}({}^0X_k)_{gt})\| * 180/\pi \tag{3.11}$$

where $rot(.)$ returns the axis-angle representation of the rotational $\mathrm{SO}(3)$ component of an $\mathrm{SE}(3)$ element.

$$\text{ASE} = \frac{1}{n_m}\sum_{i=1}^{n_m} \|({}^0\mathbf{m}^i)_{gt} - ({}^0\mathbf{m}^i)_{est}\| \tag{3.12}$$

where $n_m$ is the total number of structure 3D points.
The relative error metrics are defined as follows:

$$\text{RTE} = \frac{1}{n_x}\sum_{k=1}^{n_x} \|tr(({}^0X_{k-1}^{-1}\,{}^0X_k)_{est}^{-1}({}^0X_{k-1}^{-1}\,{}^0X_k)_{gt})\| \tag{3.13}$$

$$\text{RRE} = \frac{1}{n_x}\sum_{k=1}^{n_x} \|rot(({}^0X_{k-1}^{-1}\,{}^0X_k)_{est}^{-1}({}^0X_{k-1}^{-1}\,{}^0X_k)_{gt})\| * 180/\pi \tag{3.14}$$

$$\text{RSE} = \frac{1}{n_m}\sum_{i=1}^{n_m} \|({}^0\mathbf{m}^i - {}^0\mathbf{m}^{i-1})_{gt} - ({}^0\mathbf{m}^i - {}^0\mathbf{m}^{i-1})_{est}\| \tag{3.15}$$

Chapter 4

# Structural SLAM

Although chaotic, cluttered and highly dynamic, the world we live in has a significant amount of structure. All built environments share some underlying layout that has not yet been fully exploited in SLAM algorithms.

Figure 4.1: **Abundant structural information in urban environments.** Bird-eye view of Zamalek, Cairo. Photo by Vitaliy Raskalov to Cairo Scene.

## 4.1 Motivation

Building globally and structurally consistent maps is an essential capability for robots autonomously navigating the environment and performing tasks. In the SLAM community, there is no universally accepted notion of global structural consistency [201]. Algorithms that are highly accurate (minimising residual errors) can still admit large structural inconsistencies that contravene relatively simple global geometric structure such as vertical walls, flat floors, etc. In particular, the returned map may drift very far from the true global structure of the environment. We refer to such a situation as the algorithm displaying poor global structural consistency.


Photo by GreyOrange Photo by Australia Post

Figure 4.2: **Indoor and outdoor mobile robotics in built environments.**

Classical SLAM algorithms usually make no assumptions about the structure of the scene being analysed [14]. In practice, however, these algorithms are applied to scenes that contain significant structure that could be exploited.

Service robotics carrying out household chores, entertainment robots leading museums visits, warehouse robotics sorting out mail delivery or search and rescue robots within built environments are all examples of robots operating in indoor environments. Outdoor robotics for autonomous driving in urban environments, Unmanned Aerial Vehicles (UAV) for delivery and surveillance robots are examples of autonomous mobile robotic platforms operating outdoors. All these environments share an underlying structure to as how they are constructed.

Approaches in the recent literature to include structural information into the SLAM problem only represent the environment as planar surfaces [146, 140, 151, 11, 68], and marginalise features on those surfaces, motivated by the fact that many planar scenes are featureless. And only a few consider mapping points and planar surfaces [147]. Our work falls in the second category, where we estimate for both planar (such as walls, floors, ceilings, building facades, and roads) surfaces and points pertaining to planar and non-planar (such traffic lights, and street signs.) surfaces to achieve a more realistic mapping of the scene.

The contribution of this chapter is to explore the effect of adding meta-structural information into the estimation problem, and evaluate the structural consistency of the results. In particular, we exploit planar information applied to 3D points that are known *a-priori* to lie on planar surfaces. We also exploit inter-plane constraints in terms of parallelism and orthogonality between the detected planes. The plane parameters are incorporated into the estimation problem as additional random variables that are dependent on the environment points, and/or other planes but are not directly observed by the robot. We show that by utilising prior knowledge of the environment, more accurate and globally consistent solutions can be obtained.

## 4.2 Methodology

### 4.2.1 Problem formulation

The goal is to estimate the 3D structure of the environment and the camera poses that maximally satisfy a set of measurement constraints [198]. As in the literature [22, 34], we assume Gaussian noise models and formulate the SLAM problem as a non-linear least-squares optimisation as described in section 3.2.2.1.
The solution of the problem is obtained by minimising the sum of squared non-linear residuals:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \Big\{ \sum_{k}^{n_o} \mathbf{e}_k^\top({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k)\, \Sigma_o^{-1}\, \mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k) + \sum_{i,k}^{n_z} \mathbf{e}_{i,k}^\top({}^0\mathbf{x}_k, {}^0\,\mathbf{m}^i)\, \Sigma_z^{-1}\, \mathbf{e}_{i,k}({}^0\mathbf{x}_k, {}^0\,\mathbf{m}^i)) \Big\}, \quad (4.1)$$

Iterative non-linear optimisation methods such as Gauss-Newton or Levenberg-Marquardt can be used to find a solution that minimises (4.1). At each iteration, the cost function in (4.1) is linearised and the solution is found by solving a linear least-squares problem in $\delta$ [32]:

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}}{\operatorname{argmin}} \; \frac{1}{2}\|A\boldsymbol{\delta} - \mathbf{b}\|_2^2 \,, \quad (4.2)$$

where $A$ is the system Jacobian matrix which gathers the derivatives of the residuals in ((4.1)) with respect to variables in $\boldsymbol{\theta}$ weighted by a block-diagonal matrix that gathers all the square rooted, inverse covariances of all the observations; and $\mathbf{b}$ is the residual vector evaluated at the current linearisation point weighted by the same block diagonal matrix.

### 4.2.2 Adding planar constraints to a graph-based SLAM system

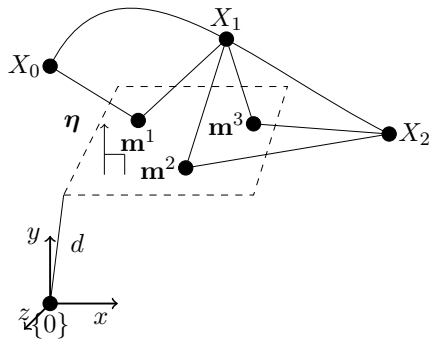



Figure 4.3: **SLAM with planar information representation.** Camera poses are represented by $X_k$, landmarks by $\mathbf{m}^i$ and plane parameters by $\boldsymbol{\eta}$ and $d$.

In this chapter, we explore the effect of integrating additional information about the landmarks in the environment into the SLAM estimation problem. In particular, we exploit planarity information applied to points that pertain to the same plane, and angular information between planes in terms of parallelism or orthogonality constraints. This a common situation in "man-built environments" where planar surfaces are ubiquitous.

While only concerned with constraining some points to planes and applying orthogonality or parallelism constraints between planes, the formulation presented here is general and can be extended to include any type of structural information present in the environment as long as there is a *front-end* that can provide that information and a function that can model it. The implemented front-end uses the raw RGB-D input images to detect, associate and merge planes as shown in Fig. 4.4, and as explained in section 3.2.1.2.

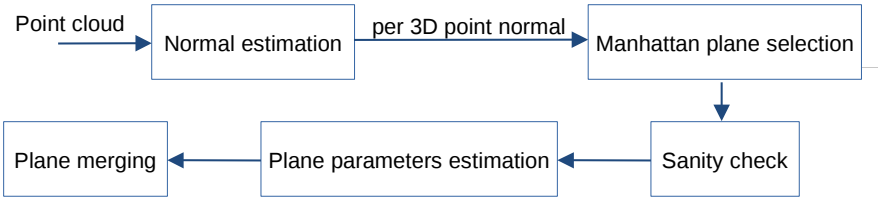



Figure 4.4: **Proposed planar surfaces extraction and association pipeline.**

A plane is parameterised by the unit normal to the planar surface $\boldsymbol{\eta}^s \in \mathrm{S}^2$ the unit sphere in space, with $\boldsymbol{\eta}^s = [\eta_x^s, \eta_y^s, \eta_z^s]^\top$, $||\boldsymbol{\eta}^s|| = 1$, and the distance to the planar surface ${}^0d^s \in \mathbb{R}$ , with $n_s$ the total number of detected planes.

For each point ${}^0\mathbf{m}^i$ that lies on a plane defined by ${}^0\mathbf{p}^s = [\boldsymbol{\eta}^s, {}^0 d^s]^\top$ one has:

$$0 = g({}^0\mathbf{m}^i,\ {}^0\mathbf{p}^s) + q \quad \text{with} \quad g({}^0\mathbf{m}^i,\ {}^0\mathbf{p}^s) = {}^0 d^s - {}^0\mathbf{m}^{i\top}\ \boldsymbol{\eta}^s\ , \tag{4.3}$$

where $q \sim \mathcal{N}(0, \Sigma_q)$ normally distributed zero-mean Gaussian noise. Here $q$ models the true distance of the point from the plane, that is the formulation does not require the underlying plane model to hold exactly, such as for example in the case where points in a corridor may lie on pin-boards on the wall and vary slightly in depth, or points on the buildings lie on the window frames, etc. We choose to use an over-parameterisation to parameterise a plane using $\eta^s$ and $d^s$, as in this way, it is easiest to represent the distance of a point to the plane as in the way stated above, and also implies a mathematically simple Jacobian computation.
Furthermore, given two planes with $\boldsymbol{\eta}^{s_1}$ and $\boldsymbol{\eta}^{s_2}$, an angle constraint can be defined as:

$$\cos(a^t) = \boldsymbol{\eta}^{s_1\top}\ \boldsymbol{\eta}^{s_2} + r\ , \tag{4.4}$$

where $r$ is the normally distributed zero-mean angle noise with covariance matrices $\Sigma_r$. Here $r$ models the confidence that is assigned to plane orthogonality or parallelism constraint $a^t$.
We deliberately choose not to introduce the angle between two planes as a random variable in the estimation, but rather introduce edges and factors to constraint the planes estimates to respect the angular information. This is because it is more realistic that a *front-end* would give semantics in terms of walls, ground, ceiling, etc. from which orthogonality and parallelism relationships can be easily deduced rather than actual measured angles between planar surfaces.

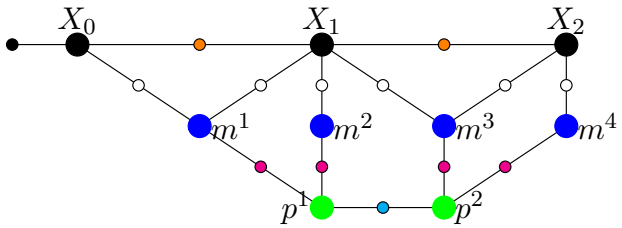



Figure 4.5: **Factor graph representation of a SLAM problem with added planar and angular information.** New variables $p^s$ are added to incorporate the planes as well as the new factors, $g_s$ in magenta to incorporate a point-plane planarity factor and $a_t$ in cyan to incorporate a plane-plane angular factor.

### 4.2.3 Cost function

The factor graph representing a SLAM problem which integrates two planar constraints and one angular constraint is shown in Fig. 4.5. The non-linear least-squares that minimises all the residual errors is then defined by integrating the meta-structural information into the estimation

problem as follows:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \Big\{ \sum_{k}^{n_o} \mathbf{e}_k^\top({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k)\, \Sigma_o^{-1}\, \mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k) + \sum_{i,k}^{n_z} \mathbf{e}_{i,k}^\top({}^0\mathbf{x}_k, {}^0\,\mathbf{m}^i)\, \Sigma_z^{-1}\, \mathbf{e}_{i,k}({}^0\mathbf{x}_k, {}^0\,\mathbf{m}^i)) + \sum_{i,s}^{n_s} (d^s - {}^0\mathbf{m}^{i\top}\, \boldsymbol{\eta}^s)^\top\, {\Sigma_q}^{-1}\, (d^s - {}^0\mathbf{m}^{i\top}\, \boldsymbol{\eta}^s) + \sum_{s}^{n_t} (\cos(a^t) - \boldsymbol{\eta}^{s_1\top}\boldsymbol{\eta}^{s_2})^\top\, {\Sigma_r}^{-1}\, (\cos(a^t) - \boldsymbol{\eta}^{s_1\top}\boldsymbol{\eta}^{s_2}) \Big\}, \quad (4.5)$$

where $n_o$, $n_z$, $n_s$ and $n_t$ are the total number of odometric measurements, point measurements, plane observations and angle observations respectively.
Appendix section A.1 explains particularities on the Jacobian when solving the NLS in (4.5).

## 4.3 Experimental setup

The framework is evaluated on Manhattan-like worlds, where structures typically exhibit a high degree of organisation in the form of orthogonal and parallel planes [202]. In order to evaluate the structural consistency of the proposed methodology, We generated several datasets with ground truth data. This is done by simulating a robot moving in a Manhattan-like world and observing 3D points in the environment using a sensor with a limited field of view. Odometric measurements are also available. We also emulate an advanced front-end that is able to fit planes to the observed 3D points, and provide initial planes parameters estimates to the back-end. Planes are assumed to be either orthogonal or parallel, and a threshold-based decision process is used to determine this.

### 4.3.1 Simulated dataset

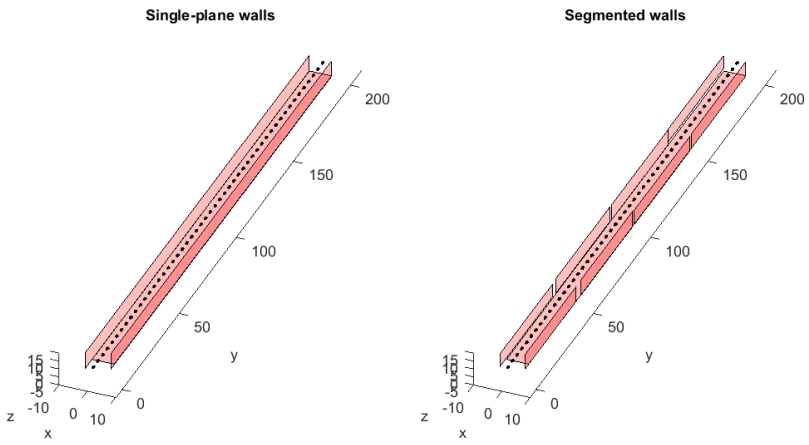



Figure 4.6: **Simulated street dataset.** [https://github.com/MinaHenein/Entity-SLAM]

To clearly observe the effects of adding structural information, we simulate a robot moving forward on a long trajectory without revisiting previous locations. As a baseline, the system is solved with no additional structural information (NoI) integrated into the estimation. This is shown in sub-figures a) in both Fig. 4.7 and 4.8. We then consider two different configurations of planes to model the buildings on the left and right sides of a road.

- In the first case, all buildings on each side are represented as a single plane.
- In the second case, four planes are generated on each side to more accurately and realistically model separate buildings on a street.

These two cases are repeated when planar information (PI) is added to the system, sub-figures b) and d) and when both planar and angular information (PAI) is added, sub-figures c) and e) in both Fig. 4.7 and Fig. 4.8. For each of these cases, the same odometry and point measurements are used to provide a valid comparison. This comparison is repeated with ten sets of randomly generated measurements. Experiments were conducted with combinations of several noise levels for odometry, point measurement, plane constraints and angle between planes. The noise levels are shown in the captions of the corresponding figures.

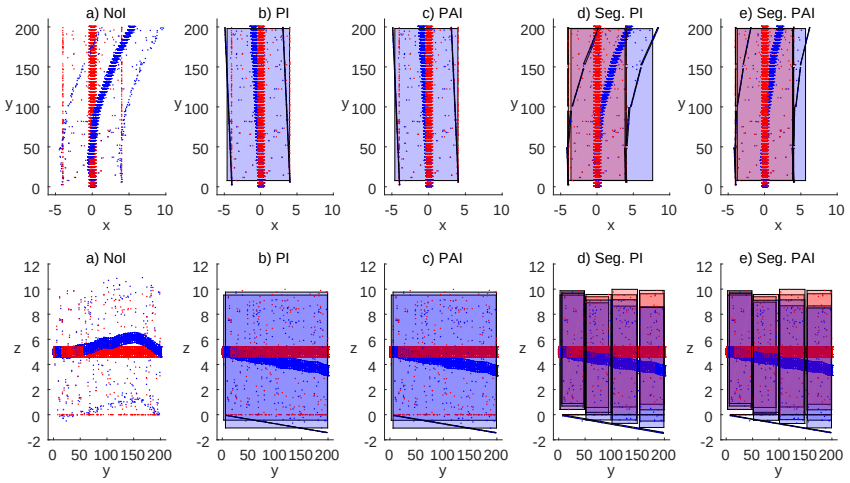



Figure 4.7: **Results on simulated street dataset.** Comparison of SLAM with a) no added structural information, with added planar information when modeling walls as b) one plane and d) four separate planes, and with both planar and angular information when modeling walls as c) one plane and e) four separate planes. The Second row shows a different view of the same scene. Ground-truth is shown in red and the final estimation is shown in blue. The figure is not displayed to scale. Noise levels used are: $\Sigma_o = [0.02m, 0.02m, 0.02m, 2°, 2°, 2°]^2$, $\Sigma_z = [0.2m, 0.2m, 0.2m]^2$, $\Sigma_q = [0.01m]^2$ and $\Sigma_r = [0.05°]^2$.

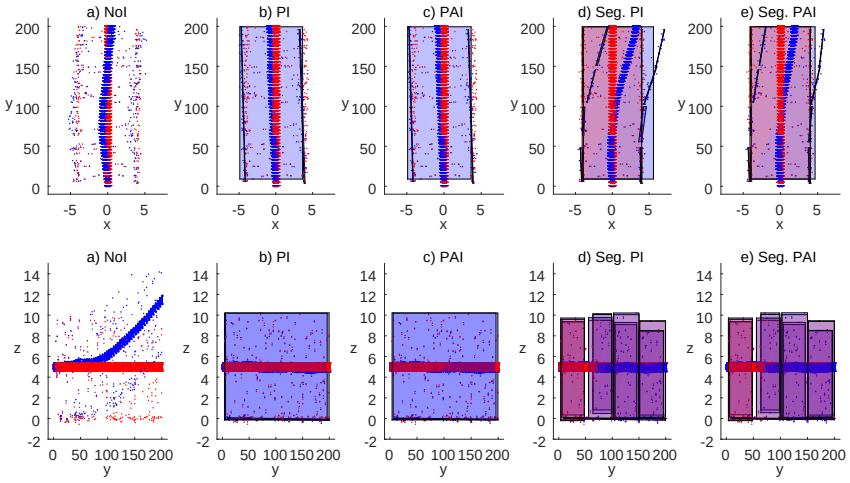



Figure 4.8: **Results on simulated street dataset.** Comparison of SLAM with a) no added structural information, with added planar information when modeling walls as b) one plane and d) four separate planes, and with both planar and angular information when modeling walls as c) one plane and e) four separate planes. The Second row shows a different view of the same scene. Ground-truth is shown in red and the final estimation is shown in blue. The figure is not displayed to scale. Noise levels used are: $\Sigma_o = \text{diag}[0.02m, 0.02m, 0.02m, 2°, 2°, 2°]^2$, $\Sigma_z = \text{diag}[0.2m, 0.2m, 0.2m]^2$, $\Sigma_q = [0.2m]^2$ and $\Sigma_r = [0.05°]^2$.

### 4.3.2 Synthetic city dataset

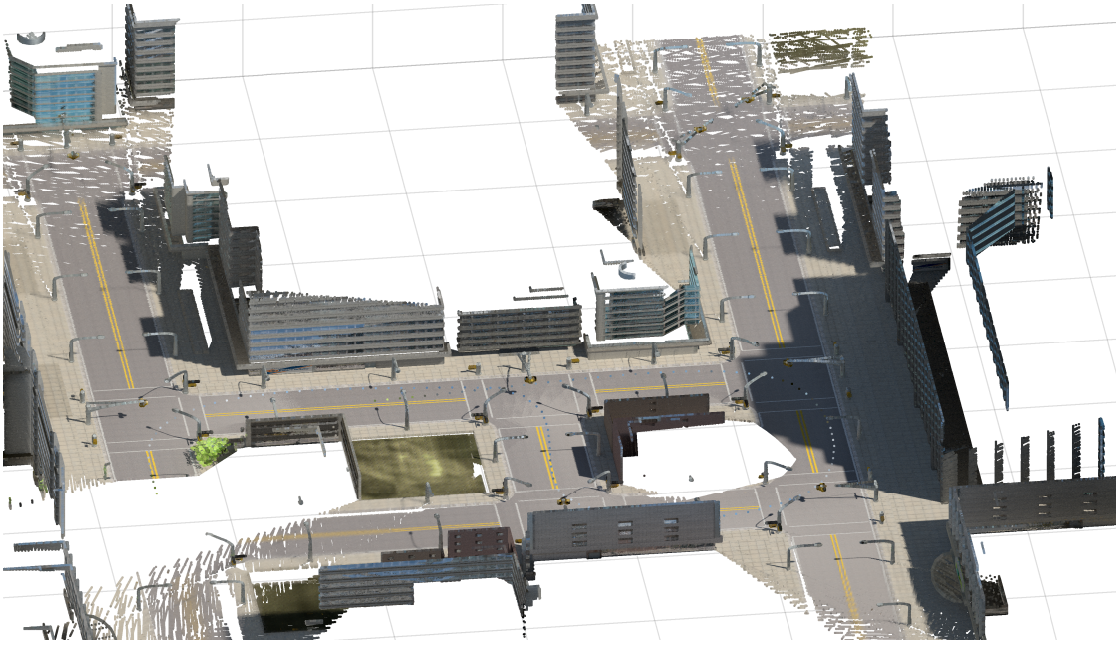


Figure 4.9: **City dataset.** Point cloud generated from synthetic Blender-generated city dataset by Zhang *et al.* [185]

The second dataset is an existing synthetic Blender-generated city by Zhang *et al.* [185], from which we generated a dataset which consists of RGB-D images acquired by a robot travelling in this synthetic city. Ground truth depth can be associated to every pixel in the image. It is worth mentioning that this dataset provides a good model of a realistic environment, with the only difference of having perfect depth estimates and ground-truth camera poses. To realistically model a real case scenario, we deliberately assign high noise levels to odometry and points measurements ($\Sigma_o = \text{diag}[0.4m, 0.4m, 0.4m, 6°, 6°, 6°]^2$, $\Sigma_z = \text{diag}[0.4m, 0.4m, 0.4m]^2$).
A feature matching and tracking algorithm was run to generate a set of 3D point measurements $\in$ $\mathbb{R}^3$. On top of that, we implemented a plane extraction algorithm that uses M-estimator SAmple Consensus (MSAC), a variant of the RANdom SAmple Consensus (RANSAC) as explained in Fig. 4.4, and more detailed in section 3.2.1.2. Only the planes that satisfy the Manhattan world assumption are integrated into the estimation. Two experiments were run with this dataset, one where the robot stops before closing a loop (62 poses and 550 3D points) and a second one with a loop closure (135 camera poses and 1500 3D pints). Experiments were conducted with combinations of several noise levels for odometry, point measurement, plane constraints and angle between planes. The noise levels are shown in the captions of the corresponding figures.

## 4.4 Experimental results

We are focused on analysing the accuracy and structural consistency of the proposed estimation solution and compare it to the classical SLAM formulation which does not integrate any additional information about the 3D points in the environment. In this sequence of experiments, we consider points constrained to planes, and orthogonality and parallelism information between planes, that is using $a^t = 90\,\text{deg}$ or $a^t = 0\,\text{deg}$. The accuracy is evaluated by comparing the absolute trajectory translational error (ATE), the absolute trajectory rotational error (ARE), the absolute structure error (ASE), the relative trajectory translational error (RTE), the relative trajectory rotational error (RRE), and the relative structure error (RSE) as described in section 3.2.3. The absolute error is used to evaluate the structural consistency of the estimation. Numbers in Table. 4.1 and 4.2 show better accuracy results in case of added structural information. More analysis follows in subsections 4.5.1 & 4.5.2.

| | **Simulated Street Dataset** | | | | |
|---|---|---|---|---|---|
| **Error** | **NoI*** | **PI*** | **PAI*** | **SPI*** | **SPAI*** |
| **ATE** | 1.42 m | **0.74 m** | 0.74 m | 0.89 m | **0.83 m** |
| **ARE** | 0.33 ° | 0.12 ° | **0.12** ° | 0.20 ° | **0.18** ° |
| **ASE** | 2.28 m | 0.93 m | **0.92 m** | 1.21 m | **1.11 m** |
| **RTE** | 0.05 m | **0.05 m** | 0.05 m | **0.05 m** | 0.05 m |
| **RRE** | 0.21 ° | 0.16 ° | **0.16** ° | 0.16 ° | **0.16** ° |
| **RSE** | 0.70 m | 0.66 m | **0.66 m** | 0.67 m | **0.67 m** |

Table 4.1: **Results on simulated street dataset.** Average error values for 106 experiments of the simulated dataset.
NoI: without added information, PI: planar information added, PAI: planar & angular information added, SPI and SPAI are the same are PI and PAI respectively in the case where each side of the street is modeled as several planes.

| | Synthetic City Dataset | |
|---|---|---|
| **Error** | **NoI*** | **PAI*** |
| **ATE** | 1.28 m | **0.65 m** |
| **ARE** | 5.79° | **2.44°** |
| **ASE** | 1.29 m | **0.63 m** |
| **RTE** | 0.31 m | **0.30 m** |
| **RRE** | 1.35° | **1.21°** |
| **RSE** | 0.68 m | **0.50 m** |

Table 4.2: **Results on synthetic city dataset.** Error values for the synthetic city dataset without loop closure. * NoI: without added information, PAI: planar & angular information added.

## 4.5 Discussion

### 4.5.1 Analysis of the simulated street dataset

Simulations were run for 16 different combinations of point measurement and plane flatness noise levels, with 10 different randomly generated measurement datasets, giving a total of 160 trials. We only show average results of 106 different trials and figures of two representative trials; full result figures and the associated code can be found on Entity SLAM website on the following url https://github.com/MinaHenein/Entity-SLAM.

We refer to the remainder of trials as 'local minima' trials and these will be discussed later. From the tests one can observe that planar information helps to preserve the structural consistency of the estimated map. This is most obviously seen in the reduction in drift between a) where no structural information is used, to the b) where walls are modelled as single planes, in both Fig. 4.7 and Fig. 4.8 and also reflected in table 4.1. There are some cases where this planar information results in increased drift. The inherent randomness of the measurement noise model can result in the estimation drifting back towards the ground-truth. A close examination of Fig. 4.8 a) shows that the estimated trajectory initially drifts towards the negative in the $x$ axis, but this is corrected by a later drift. Fig. 4.8 b) shows that the planar information tends to preserve the straightness of the structure, and thus this reversal of direction is not allowed. This situation is unlikely to occur in a real environment, where a city is more realistically modelled by separate planes. When considering the case where walls/buildings are modelled as several planes, the system is more susceptible to drift because of the increased freedom allowed by segmentation of the planes. In fact, if the number of planes used to represent one building is sufficiently high, the added planar information act as the no information case; as if every point had the freedom

to lie on a single plane. The addition of angular information helps to reduce this effect, as this additional information reshapes the problem as a single plane. This can be seen by comparing d) and e) in both Fig. 4.7 and Fig. 4.8.

Table 4.1 shows 47% error reduction in ATE, 59% reduction in ASE and 63% reduction in ARE when considering 1 plane on each side (PI & PAI) and 37% error reduction in ATE, 46% reduction in ASE and 38% reduction in ARE for planar information when considering segmented planes (SPI) and 41% error reduction in ATE, 51% reduction in ASE and 44% reduction in ARE for planar and angular information when considering segmented planes (SPAI).

Finally, when including angular information, care should be taken when choosing the strictness of the constraints, as the system can become sensitive to large measurement noise resulting in failure of the estimation. The corrections provided by the angle information must not exceed the freedom in plane fitting allowed by the spatial distribution of the points. This is reinforced by the observation that experiments where points are generated loosely on the planes as in Fig. 4.8 the estimation is less susceptible to failure, as opposed to very close to planes as in Fig 4.7.

**Local-minima cases** It is worth mentioning that angular information should be added with care, as solving for angular constraints has shown to be very sensitive to initial conditions and could cause the problem to get stuck in a local minima. This has happened in 33% of the 160 total experiments run for the simulated street dataset (Table 4.1 does not include these cases).

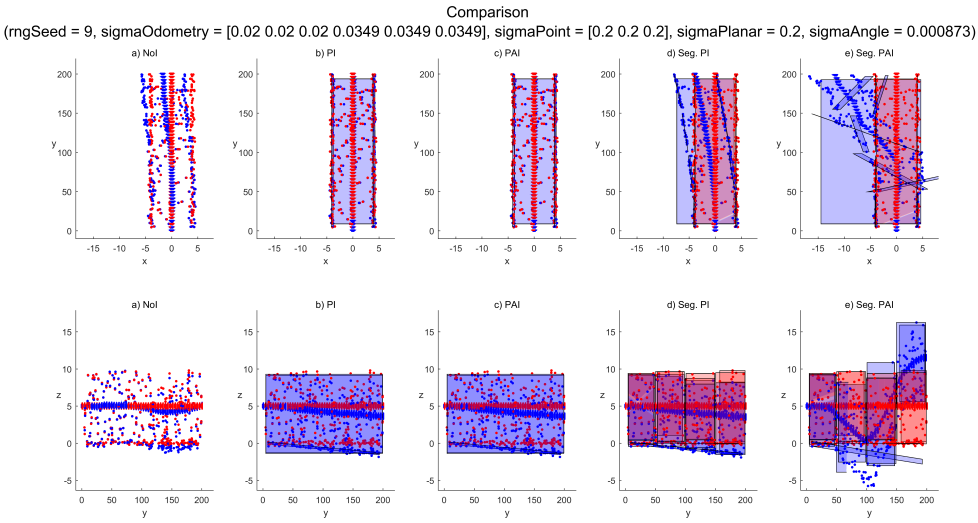



Figure 4.10: **Failure case when adding planar and angular information.** Representative example of a failure case when adding planar and angular information, in the several planes case (SPAI), causes the problem to get stuck in a local-minima.

A possible way to alleviate this effect, is to first solve with added planar information, and only add angular information once a solution of planar information is reached. Fig 4.10 and 4.11 show a representative example of this case, where adding angular information initially caused failure of the estimation. However, when first solving with added planar information and once a solution is reached, re-solving the system with the added angular information and the solution of the planar information as initial estimates yields better results.

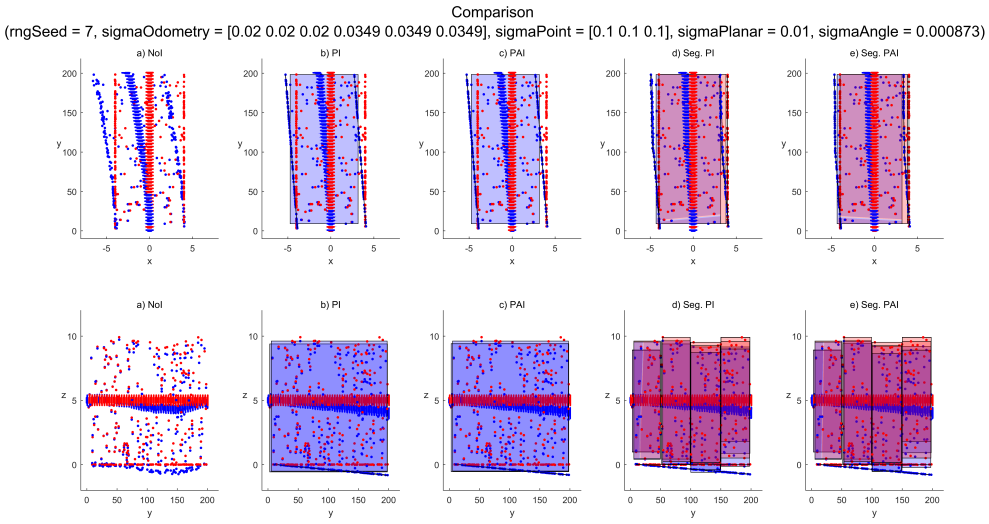



Figure 4.11: **Solution to local-minima problem.** Representative example of a local-minima case solved by first adding planar information, and then adding angular information starting from the solution of the added planar information.

It is important to note that the structural information added, in terms of planarity of a group of points, does not necessarily fully constrain the degrees of freedom of 3D point landmarks. This is the reason of the drift seen in Fig. 4.7. The angular information add further constraints and help reduce this drift. A possible solution is to add geometric distance constraints between a 3D point and all planes orthogonal to the one it resides on.

### 4.5.2 Analysis of the synthetic city dataset

Although run for two different setups (with and without a loop closure), and four different noise levels each, we only show results from one experiment and a single noise level, full results figures can be found at the same web page mentioned above.
It is known that loop closures help to preserve structural consistency when the same locations are revisited and the same environment landmarks are re-observed and used to correct the drift. Therefore in the city dataset, adding planar and angular information appears to have minimal

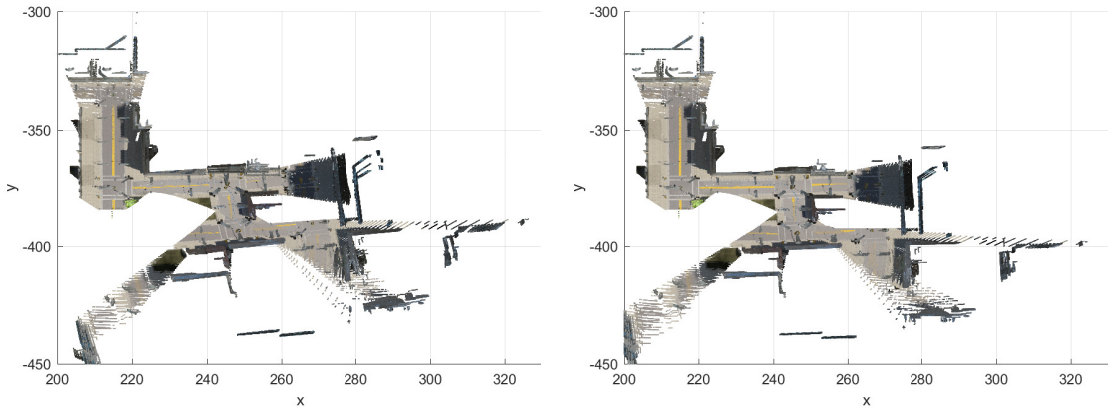



Figure 4.12: **Final city map estimate.** Top view of the final city map estimate (no loop closure) with no added information (left) vs with planar & angular information added (right). Where some buildings appear to lose orthogonality on the left image, the same buildings appear to show planarity and orthogonality on the right image preserving the global consistency of the map. Noise levels used are as follows: $\Sigma_o = \text{diag}[0.4m, 0.4m, 0.4m, 6°, 6°, 6°]^2$, $\Sigma_z = \text{diag}[0.4m, 0.4m, 0.4m]^2$, $\Sigma_q = [0.01m]^2$ and $\Sigma_r = [1°]^2$

effect after the loop is closed. Improvements are however shown in the case where the robot traversed the city but no large loops have been closed yet. It is here where the estimate drifts by a significant amount without added structural information. Using prior knowledge about the environment, in terms of planar and angular information, improves the overall estimation accuracy and preserves consistency. Fig. 4.12 shows the effect of adding planar and angular information on the global consistency of the estimation, and Table 4.2 shows a reduction of 50% in absolute errors. While used as preset values here, the covariance associated with the planar and angular information $\Sigma_q$ and $\Sigma_r$ respectively can be chosen, in real scenarios, based on the semantics of the scene $-$in terms of layout and spatial relationship between segmented surfaces$-$ as a function of the distance of the point to the segmented planes and the confidence of the returned planar segmentation.

Although only tested on simulated and synthetic datasets, we believe the proposed algorithm's applicability to real scenarios is fairly straightforward. A more robust front-end should however be integrated to ensure robust feature tracking and landmark-plane association in the case of real environments. We believe this is made possible with the advances in learning techniques to estimate surface normals and integrate depth and scene semantics to improve the estimation as in [203, 204, 205, 206]

## 4.6 Conclusion and future work

In this chapter, we have shown the effect of additional structural information on the accuracy and global consistency of SLAM. A factor graph formulation of a SLAM problem is used to incorporate prior knowledge of the environment into the estimation. Results show improvements in the estimation accuracy. 3D points pertaining to a certain surface show better convergence results versus the same points with no added information.
The proposed algorithm demonstrates better convergence and global structural consistency results in most cases and still can be improved in many aspects. A possible improvement can be made by using the SLAM estimates to refine the plane detections in a prediction step [11].
While presented in isolation here, the planar and orthogonality information can be combined with other geometric or semantic information as well as dynamics of the objects in the scene to handle a larger group of problems. Different information can be added to the system such as distance information for geometric shapes of known dimensions or velocity information for moving rigid objects, this can greatly improve the incremental segmentation SLAM problem [207] and opens wide doors to SLAM with objects and dynamic SLAM estimation problems.

Chapter 5

# Dynamic SLAM

*"Most of the structures in the visual world are rigid or at least nearly so."*

*- David Marr* [208]

## 5.1 Motivation

While many accurate and efficient solutions to the SLAM problem exist, current SLAM algorithms can be easily induced to fail in highly dynamic environments [184]. The conventional technique to deal with dynamics in SLAM is to either treat any sensor data associated with moving objects as outliers and remove them from the estimation process [168, 169, 170, 171, 172], or to detect moving objects and track them separately using traditional multi-target tracking approaches [173, 174, 175, 176]. The former technique excludes information about dynamic objects in the scene, and generates static only maps. Accuracy of the latter is dependent on the camera pose estimation, which is more susceptible to failure in complex dynamic environments where the presence of reliable static structure is questionable. Increased applications of autonomous systems to dynamic environments is driving a need to challenge the scene rigidity assumption, also known as the static world assumption, that underpins most existing SLAM and Visual Odometry (VO) algorithms.

A typical SLAM system consists of a front-end module, that processes the raw data from the sensors and a back-end module, that integrates the obtained raw and implicit higher-level information into a probabilistic estimation framework. Simple primitives such as 3D locations of salient features are commonly used to represent the environment. This is largely a consequence of the fact that points are easy to detect, track and integrate within the SLAM estimation problem. Other primitives such as lines and planes [67, 11, 12, 68] or even objects [209, 8, 13] have been considered in order to provide richer map representations. Semantic information and

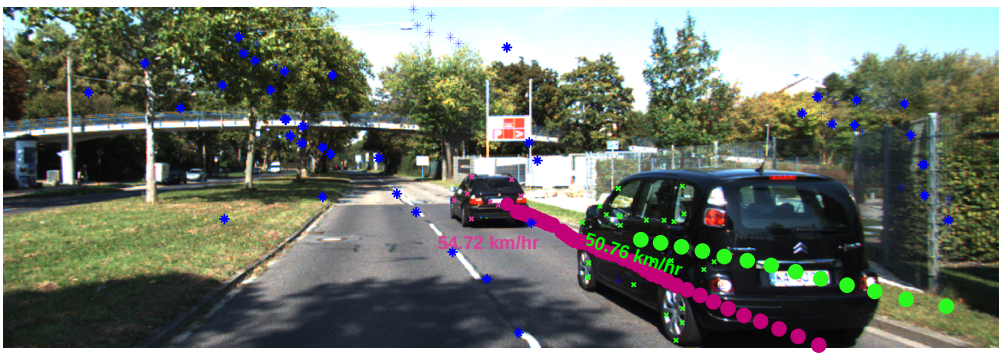



Figure 5.1: **Results of object-aware dynamic SLAM on KITTI seq.03.** Centroids of each object are obtained by applying the motion estimates to the first ground-truth object centroid. Speed estimates are also extracted for each object.

object segmentation can provide important prior information in identifying dynamic objects in the scene [167, 159], and can be highly valuable in a dynamic SLAM framework. Advances in deep learning have provided algorithms that can reliably detect and segment classes of objects at almost real time [19, 5]. Despite recent developments in vision-based object detection and segmentation, the visual SLAM community has not yet fully exploited such information [20]. To incorporate semantic information in existing geometric SLAM algorithms then either a dataset of 3D-models of every object in the scene must be available a-priori [8, 159] or the front end must explicitly provide object pose information in addition to detection and segmentation [210, 211, 212] adding a layer of complexity to the problem. The requirement for accurate 3D-models severely limits the potential domains of application, while to the best of our knowledge, multiple object tracking and 3D pose estimation remain a challenge to learning techniques. There is a clear need for an algorithm that can exploit the powerful detection and segmentation capabilities of modern deep learning techniques without relying on additional pose estimation or object model priors.

In this chapter, we propose a novel model-free, object-aware point-based dynamic SLAM approach that leverages image-based semantic information to model dynamic scenes in a unified estimation framework over robot poses, static and dynamic 3D points, and object motions and simultaneously localise the robot, map the static structure, estimate a full $\mathrm{SE}(3)$ pose change of moving objects and build a dynamic representation of the world.
We redefine the term *mapping* in SLAM to be concerned with a *spatio-temporal representation of the world*, as opposed to the concept of a static map that has long been the emphasis of classical SLAM algorithms, including SLAM systems that can robustly operate in dynamic environments by excluding the dynamics of the world. We also fully exploit the rigid object motion to extract velocity information of objects in the scene, as shown in Fig. 5.1, an emerging task in

autonomous driving which has not yet been thoroughly explored [16]. Although the developed techniques apply to a wide range of applications, velocity information of vehicles is crucial to aid autonomous driving algorithms for tasks such as collision avoidance [17] or adaptive cruise control [18]. The key innovation in the chapter is a novel *pose change representation* used to model the motion of a collection of points pertaining to a given rigid body and the integration of this model into a SLAM optimisation framework. The resulting algorithm is agnostic to the underlying 3D-model of the object.
To the best of our knowledge, this is the first piece of work able to estimate, along with the camera poses, the static and dynamic structure, the full $\mathrm{SE}(3)$ pose change of every rigid object in the scene, extract object velocities and be demonstrable on a real-world outdoor dataset.

## 5.2 Methodology

In the following, we show how to model the motion of a rigid-object in a model free manner based on point tracking. We propose a factor graph optimisation to estimate the camera and object motion along with the environment static structure.

The problem considered is one in which there are relatively large rigid objects moving within the sensing range of the robot that is undertaking the SLAM estimation. The SLAM front-end is able to identify and associate points from the same potentially moving object at different time steps. These points share an underlying motion constraint that can be exploited to improve the quality of the SLAM estimation.

### 5.2.1 Background and notation

#### 5.2.1.1 Coordinate frames

Let ${}^{0}X_k, {}^{0}L_k \in \mathrm{SE}(3)$ be the robot/camera and the object 3D pose respectively, at time $k$ in a global reference frame 0, with $k \in \mathcal{T}$ the set of time steps. Note that calligraphic capital letters are used to represent sets of indices. Fig. 5.2 shows these pose transformations as solid curves.

#### 5.2.1.2 Object and 3D point motions

The object motion between times $k-1$ and $k$ is described by the homogeneous transformation ${}^{L_{k-1}}_{k-1}H_k \in \mathrm{SE}(3)$ according to:

$$ {}^{L_{k-1}}_{k-1}H_k = {}^{0}L_{k-1}^{-1}\,{}^{0}L_k \,. \tag{5.1}$$

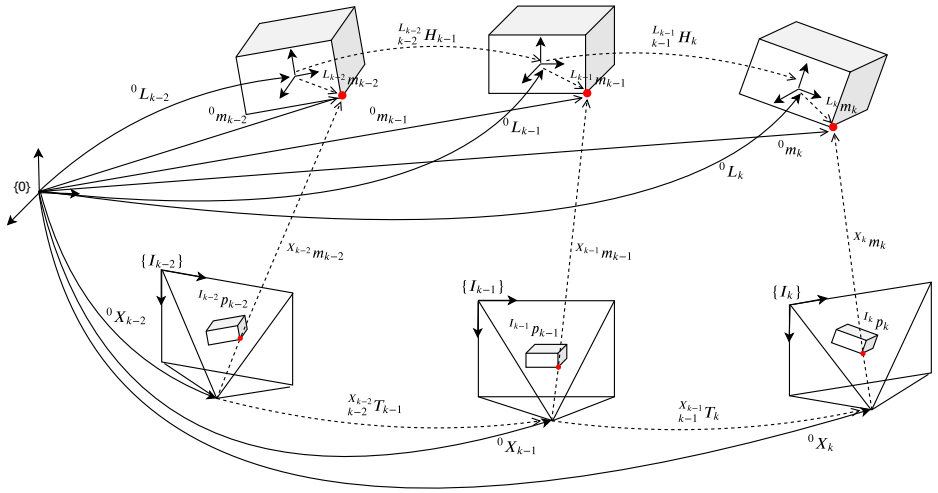



Figure 5.2: **Notation and coordinate frames.** Solid curves represent camera and object poses in inertial frame; ${}^0X$ and ${}^0L$ respectively, and dashed curves their respective motions in body-fixed frame. Solid lines represent 3D points in inertial frame, and dashed lines represent 3D points in camera frames.

Fig. 5.2 shows these motion transformations as dashed curves.
Let ${}^0\mathbf{m}_k^i$ be the homogeneous coordinates of the $i^{th}$ 3D point at time $k$, with ${}^0\mathbf{m}^i = \left[m_x^i, m_y^i, m_z^i, 1\right]^\top \in \mathbb{E}^3$ and $i \in \mathcal{M}$ the set of points.
A point is written in its corresponding object frame as:

$$ {}^{L_k}\mathbf{m}_k^i = {}^0L_k^{-1}\, {}^0\mathbf{m}_k^i \,. \tag{5.2} $$

This is shown as a dashed vector from the object reference frame to the red dot in Fig. 5.2. Substituting the object pose ${}^0L_k$ at time $k$ from (5.1), this becomes:

$$ {}^0\mathbf{m}_k^i = {}^0L_k\, {}^{L_k}\mathbf{m}_k^i = {}^0L_{k-1}\, {}^{L_{k-1}}_{k-1}H_k\, {}^{L_k}\mathbf{m}_k^i \,. \tag{5.3} $$

Note that for rigid body objects, ${}^{L_k}\mathbf{m}_k^i$ stays constant at ${}^L\mathbf{m}^i$, and so ${}^L\mathbf{m}^i = {}^0L_k^{-1}\, {}^0\mathbf{m}_k^i = {}^0L_{k+n}^{-1}\, {}^0\mathbf{m}_{k+n}^i$ for any integer $n \in \mathbb{Z}$. Then, for rigid objects, and with $n = -1$, (5.3) becomes:

$$ {}^0\mathbf{m}_k^i = {}^0L_{k-1}\, {}^{L_{k-1}}_{k-1}H_k\, {}^0L_{k-1}^{-1}\, {}^0\mathbf{m}_{k-1}^i \,. \tag{5.4} $$

(5.4) is crucially important as it relates the same 3D point on a rigid object in motion at consecutive time steps by a homogeneous transformation ${}^0_{k-1}H_k := {}^0L_{k-1}\, {}^{L_{k-1}}_{k-1}H_k\, {}^0L_{k-1}^{-1}$. This equation represents a *frame change of a pose transformation* [213], and shows how the body-

fixed frame pose change ${}^{L_{k-1}}_{k-1}H_k$ relates to the global reference frame pose change ${}_{k-1}{}^{0}_{k}H$. The point motion in global reference frame is then expressed as:

$$ {}^{0}\mathbf{m}^{i}_{k} = {}_{k-1}{}^{0}_{k}H_k \; {}^{0}\mathbf{m}^{i}_{k-1} \; . \tag{5.5} $$

(5.5) is at the core of our approach, as it expresses the rigid object pose change in terms of the points that reside on the object without any knowledge about the object pose or 3D model/shape in an object-aware, model-free manner. (5.5) essentially describes the full $\mathrm{SE}(3)$ motion of each point on a moving object, and exploits the rigidity of the object to utilise all points on a dynamic object and describe a single $\mathrm{SE}(3)$ pose change of the object ${}_{k-1}{}^{0}_{k}H_k \in \mathrm{SE}(3)$ expressed in a global reference frame. It is thanks to the frame change of the pose transformation from the object body-fixed frame to a global reference frame that the pose change of the object is now independent of the object's 3D pose and allows us to express the object's motion without any knowledge about the object 3D model or pose. For the remainder of this document, ${}_{k-1}{}^{0}_{k}H_k$ is referred to as the object pose change or the object motion for ease of reading.
Note that no assumptions about the motion of the object were made, and the resulting quantity describing the object pose change is an element of the Special Euclidean group $\mathrm{SE}(3)$, obtained by a multiplication of three $\mathrm{SE}(3)$ group elements, and describes the full motion of an object in 3D. Appendix section B.1 describes the object $\mathrm{SE}(3)$ pose change in a global reference frame in the two particular cases of pure object translation and pure object rotation in body-fixed frame, and the general case of an object translation and rotation in body-fixed frame.

#### 5.2.1.3 Linear velocity extraction

While estimating the $\mathrm{SE}(3)$ pose change of moving objects in the scene might be very useful for aerial applications, linear velocity of other vehicles on the road is more intuitive in urban driving scenarios, and is crucial piece of information in autonomous driving applications. The linear velocity of a point on an object, expressed in the inertial frame, can be estimated by applying the pose change ${}_{k-1}{}^{0}_{k}H_k$ and taking the difference

$$ \begin{aligned} \mathbf{v} \approx {}^{0}\,\mathbf{m}^{i}_{k} - {}^{0}\,\mathbf{m}^{i}_{k-1} &= \left({}_{k-1}{}^{0}_{k}H_k - I_4\right) {}^{0}\,\mathbf{m}^{i}_{k-1} \\ &= {}_{k-1}{}^{0}_{k}\mathbf{t}_k - \left(I_3 - {}_{k-1}{}^{0}_{k}R_k\right) {}^{0}\mathbf{m}^{i}_{k-1}. \end{aligned} \tag{5.6} $$

To get a more reliable measurement, we average over all points on an object at a certain time. Define $\mathbf{c}_{k-1} := \frac{1}{n}\sum \mathbf{m}_{k-1}^{i}$ for all n points on an object at time $k-1$. Then

$$\begin{aligned}\mathbf{v} &\approx \frac{1}{n}\sum_{i=1}^{n}\left({}_{k-1}^{0}\mathbf{t}_k - (I_3 - {}_{k-1}^{0}R_k)\,{}^{0}\mathbf{m}_{k-1}^{i}\right)\\ &= {}_{k-1}^{0}\mathbf{t}_k - (I_3 - {}_{k-1}^{0}R_k)\,\mathbf{c}_{k-1}\,,\end{aligned} \tag{5.7}$$

where ${}_{k-1}^{0}R_k \in \mathrm{SO}(3)$ and ${}_{k-1}^{0}\mathbf{t}_k \in \mathbb{R}^3$ the rotation and translation components of ${}_{k-1}^{0}H_k$ respectively, $I_3$ is the identity matrix, and $\mathbf{c}_{k-1}$ is the object's centroid position at time $k-1$. As the proposed algorithm is sparse, we do not have access to the object's centroid but rather approximate it by the spatial mean of features detected on the object. (5.7) describes the linear velocity of an object at a certain time, and its magnitude the object's speed. A proof is added in Appendix section B.2 to show that (5.7) is the same quantity in $3D$ as the translation vector from the origin of the object pose at time $\{k-1\}$ to the origin of the object pose at time $\{k\}$ as seen in the global reference frame.

### 5.2.2 Graph optimisation

We model the dynamic SLAM problem as a factor graph. The factor graph formulation is highly intuitive and has the advantage that it allows for efficient implementations of batch [32, 199] and incremental [57, 214, 200] solvers.

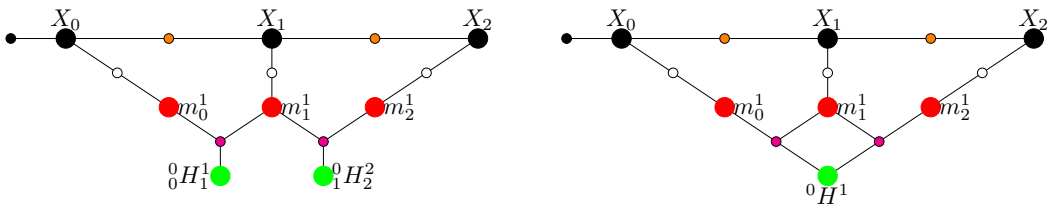



Figure 5.3: **Back-end various factor graph representations.** (a): Factor graph representation of a problem with multiple pose change vertices for the same object. (b): Factor graph representation of a problem with a unique pose change vertex for the same object.

It has been shown that in dynamic SLAM, knowing the type of motion of the objects in the environment is highly valuable [167]. In here, two scenarios are evaluated without and with constant motion model:

• In city scenarios, where the objects motions are subject to changes (acceleration, deceleration, etc.) modelling the motion is challenging. Therefore, we estimate for a new pose change at every

time step. Fig. 5.3(a) shows a factor graph representation of such scenario where the motion of the same object is estimated using two motions vertices for two different time transitions.

• A highway scenario, where every vehicle maintains a constant motion. Fig. 5.3(b) shows the factor graph representation where a single motion is estimated per object. Furthermore, we show that if the body-fixed frame pose change is constant then the reference frame pose change is constant too. For any $k-1, k'-1$ time indices, the constant motion in the body-fixed pose change is:

$$ {}^{L_{k-1}}_{k-1}H_k = C = {}^{L_{k'-1}}_{k'-1}H_{k'} \in \mathrm{SE}(3)\ . \tag{5.8} $$

We rescale (5.1) and use (5.8) to obtain: ${}^0L_k = {}^0L_{k-1}\, C$ which we replace in ${}^0_{k-1}H_k = {}^0L_{k-1}\, C\ {}^0L^{-1}_{k-1}$ to obtain:

$$ {}^0_{k-1}H_k = {}^0L_k\, C\ {}^0L^{-1}_k = {}^0_kH_{k+1}\ . \tag{5.9} $$

It follows that the reference frame pose change for a specific object $j$:
${}^0_{k-1}H^j_k = {}^0H^j = {}^0_{k'}H^j_{k'+1} \in \mathrm{SE}(3)$ holds for any $k, k'$ time indices.
Three types of measurements/observations are integrated into a joint optimisation problem; the 3D point measurements, the visual odometry measurements, and the motion of points on a dynamic object. The 3D point measurement model error $\mathbf{e}_{i,k}({}^0X_k, {}^0\,\mathbf{m}^i_k)$ is defined as:

$$ \mathbf{e}_{i,k}({}^0X_k, {}^0\,\mathbf{m}^i_k) = {}^0\ X^{-1}_k\ {}^0\mathbf{m}^i_k - \mathbf{z}^i_k\ . \tag{5.10} $$

Here $\mathbf{z} = \{\mathbf{z}^i_k \mid i \in \mathcal{M}, k \in \mathcal{T}\}$ is the set of all 3D point measurements at all time steps, with cardinality $n_z$ and $\mathbf{z}^i_k \in \mathbb{R}^3$. The 3D point measurement factors are shown as white circles in Fig. 5.3.
The odometry model error $\mathbf{e}_k({}^0X_{k-1}, {}^0\,X_k)$ is defined as:

$$ \mathbf{e}_k({}^0X_{k-1}, {}^0\,X_k) = ({}^0X^{-1}_{k-1}\ {}^0X_k)^{-1}\ {}^{X_{k-1}}_{k-1}\mathbf{T}_k\ , \tag{5.11} $$

where $\mathbf{T} = \{{}^{X_{k-1}}_{k-1}\mathbf{T}_k \mid k \in \mathcal{T}\}$ is the odometry measurement set with ${}^{X_{k-1}}_{k-1}\mathbf{T}_k \in \mathrm{SE}(3)$ and cardinality $n_o$. The odometric factors are shown as orange circles in Fig. 5.3.
The motion model error of points on dynamic objects $\mathbf{e}_{i,l,k}({}^0\mathbf{m}^i_k, {}^0_{k-1}H^l_k, {}^0\,\mathbf{m}^i_{k-1})$ is defined as:

$$ \mathbf{e}_{i,l,k}({}^0\mathbf{m}^i_k, {}^0_{k-1}H^l_k, {}^0\,\mathbf{m}^i_{k-1}) = {}^0\mathbf{m}^i_k - {}^0_{k-1}H^l_k\ {}^0\mathbf{m}^i_{k-1}\ . \tag{5.12} $$

The motion of all points on a detected rigid object $l$ are characterised by the same pose transformation ${}^0_{k-1}H^l_k \in \mathrm{SE}(3)$ given by (5.5) and the corresponding factor, shown as magenta circles

in Fig. 5.3, is a ternary factor which we call the *motion model of a point on a rigid body*.
Let $\boldsymbol{\theta}_M = \{{}^0\mathbf{m}_k^i \mid i \in \mathcal{M}, k \in \mathcal{T}\}$ be the set of all 3D points. We parameterise the $\mathrm{SE}(3)$ camera pose by elements of the Lie-algebra $\mathbf{x}_k^\wedge \in \mathrm{se}(3)$:

$$ {}^0X_k = \exp({}^0\mathbf{x}_k^\wedge) \,, \tag{5.13}$$

and define $\boldsymbol{\theta}_X = \{{}^0\mathbf{x}_k \mid k \in \mathcal{T}\}$ as the set of all camera poses, with ${}^0\mathbf{x}_k \in \mathbb{R}^6$. We parameterise the $\mathrm{SE}(3)$ object motion ${}_{k-1}^{0}H_k^l$ by elements ${}_{k-1}^{0}\mathbf{h}_k^{l\,\wedge} \in \mathrm{se}(3)$ the Lie-algebra of $\mathrm{SE}(3)$:

$$ {}_{k-1}^{0}H_k^l = \exp({}_{k-1}^{0}\mathbf{h}_k^{l\,\wedge}) \,, \tag{5.14}$$

and define $\boldsymbol{\theta}_H = \{{}_{k-1}^{0}\mathbf{h}_k^l \mid k \in \mathcal{T}, l \in \mathcal{L}\}$ as the set of all object motions, with ${}_{k-1}^{0}\mathbf{h}_k^l \in \mathbb{R}^6$ and $\mathcal{L}$ the set of all object labels. Given $\boldsymbol{\theta} = \boldsymbol{\theta}_X \cup \boldsymbol{\theta}_M \cup \boldsymbol{\theta}_H$ as all the nodes in the graph, and using the Lie-algebra parameterisation of $\mathrm{SE}(3)$ for $X$ and $H$ (substituting (5.13) in (5.10) and (5.11), and substituting (5.14) in (5.12)), the solution of the least squares cost is given by:

$$\begin{aligned}
\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \Big\{ & \sum_{i,k}^{n_z} \rho_h\big(\mathbf{e}_{i,k}^\top({}^0\mathbf{x}_k, {}^0\,\mathbf{m}_k^i)\, \Sigma_z^{-1}\, \mathbf{e}_{i,k}({}^0\mathbf{x}_k, {}^0\,\mathbf{m}_k^i)\big) \\
& + \sum_{k}^{n_o} \rho_h\big(\mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k)^\top\, \Sigma_o^{-1}\, \mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k)\big) \\
& + \sum_{i,l,k}^{n_g} \rho_h\big(\mathbf{e}_{i,l,k}^\top({}^0\mathbf{m}_k^i, {}_{k-1}^{0}\mathbf{h}_k^l, {}^0\,\mathbf{m}_{k-1}^i)\, \Sigma_g^{-1}\, \mathbf{e}_{i,l,k}({}^0\mathbf{m}_k^i, {}_{k-1}^{0}\mathbf{h}_k^l, {}^0\,\mathbf{m}_{k-1}^i)\big) \Big\} \,,
\end{aligned} \tag{5.15}$$

where $\Sigma_z$ is the 3D point measurement noise covariance matrix, $\Sigma_o$ is the odometry noise covariance matrix, and $\Sigma_g$ is the motion noise covariance matrix with $n_g$ the total number of ternary object motion factors. The non-linear least squares problem in (5.15) is solved using Levenberg-Marquardt method.

## 5.3 System

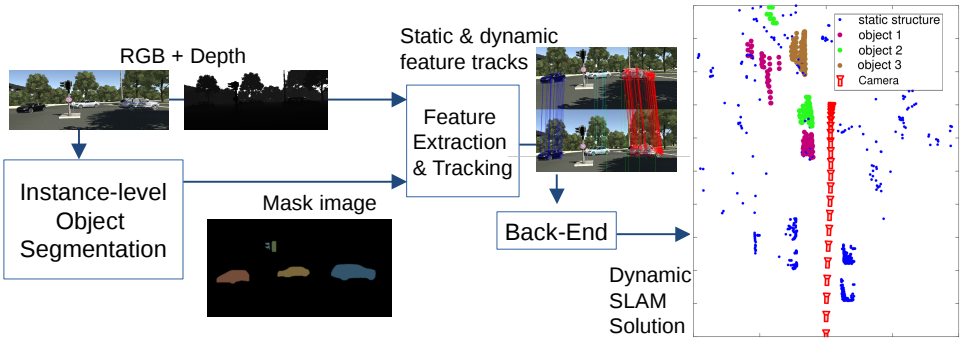



Figure 5.4: **System overview.** Input images are used into an instance level object segmentation algorithm to provide segmentation masks. The algorithm then detects and tracks features on potentially moving objects. Potentially dynamic features along with tracked static features are used to build the graph, that is then fed into the back-end for optimisation.

The system pipeline shown in Fig. 5.4 assumes a robot equipped with an RGB-D camera and proprioceptive sensors (e.g. odometers, IMU). Our feature-based object-aware dynamic SLAM back-end estimates the robot poses, the static and dynamic structure and pose transformations for every detected object in the scene. To ensure features are being detected on moving objects, we employ an instance-level object segmentation algorithm to produce instance level object masks. Object segmentation constitutes an important prior in static/dynamic object classification and tracking of dynamic objects. The front-end then makes use of object masks to detect features on *potentially-moving* objects and on static background. Feature tracking is a crucial module for the success of the proposed approach. Through object segmentation and feature tracking, the SLAM front-end is able to identify and associate points on the same rigid-body object at different time steps. These points share an underlying motion model that we exploit to achieve simultaneous localisation, mapping and moving object tracking. The algorithm does not require the front-end to estimate the objects' pose or use any geometric model of the objects. The static and dynamic 3D measurements along with the measurements from proprioceptive sensors are integrated into the SLAM back-end to simultaneously estimate the camera motion, the static and dynamic structure and the $\mathrm{SE}(3)$ pose transformations of the detected objects in the scene.

## 5.4 Experiments

The following experiments were performed to test: 1) the effect of various front-end components on the camera and object motion estimation accuracy, 2) the performance of the proposed algo-

rithm compared to a classical SLAM followed by moving object tracking from known camera poses, and 3) the consistency of the proposed algorithm in various SLAM scenarios.

It is important to note that we do not claim better camera pose estimation accuracy compared to classical SLAM systems (such as ORB-SLAM 2/3 [215, 216]) or SLAM systems that exclude dynamic objects (such as [172]) in environments with sufficient static structure. We rather show consistent results of the proposed approach in various scenarios, and emphasise that the main advantage of this work is in the spatio-temporal representation of the world and the accurate estimation of dynamic objects' pose change. Comparison of the full system version of this work versus state-of-the-art dynamic SLAM systems that integrate moving objects and estimate their motion is included in the following chapter. Better camera and object motion estimation is achieved.

### 5.4.1 Error metrics

The accuracy of the solution is evaluated vs ground-truth (GT) by comparing the Relative Translational Error (RTE) in %, that is the translational component of the error between the estimated and GT robot pose changes. Similarly, the Relative Rotational Error (RRE) in $^\circ/m$ is the rotational component of the same error. We also evaluate the Relative Structure Error (RSE) in % for all static and dynamic landmarks, as the error between the corresponding relative positions of the estimated and GT structure points. We also provide an evaluation of the object pose change estimates; the Object Motion Translation Error (OMTE) in %, the Object Motion Rotational Error (OMRE) in $^\circ/m$, calculated similar to the camera pose errors described in section 3.2.3 and for driving scenarios, the Object Motion Speed Error (OMSE) in % calculated as the difference between the ground truth speed and the extracted speed calculated from the object's pose change divided by the ground truth speed.

### 5.4.2 Virtual KITTI dataset

#### 5.4.2.1 Description

Virtual KITTI [217] is a photo-realistic synthetic dataset that provides RGB-D videos from a vehicle driving in an urban environment. Frames are fully annotated at the pixel level with unique object tracking identifiers (needed for errors calculations). Ground truth information about camera and object poses is also provided which makes it a perfect dataset to test and evaluate the proposed technique.

#### 5.4.2.2 Goal

We make use of the ground truth data to test the effect of each component in the front-end on the performance of the algorithm and the accuracy of the pose change estimation for the camera and moving objects in the scene. The three aspects studied are errors in: a) depth/3D point measurement, b) object segmentation, and c) feature tracking.

#### 5.4.2.3 Implementation

Due to the fact that the initially proposed algorithm is sparse-based, object pose change estimation is affected by the distribution of the extracted features on moving objects. Another important aspect is the percentage of the object mask in the image. In the experiments reported in this section, we only estimate for objects whose segmentation masks amount to a certain percentage of the total image. This threshold ensures to exclude far-away and partially observed objects that are entering/exiting the camera field of view and which makes their motion estimates inaccurate. This threshold is set to 6% for vKITTI, and 2% for KITTI.
For all the tests reported in this subsection, odometry noise level is kept constant and GT is used for every module of the front-end except the one being tested. Point measurement noise is only varied when testing the effect of depth information, otherwise kept constant as explained below.

**Effect of depth information** We evaluate the performance of the proposed algorithm using ground truth object segmentation, and feature tracking with odometry and varied point measurement noise. The noise levels added are 5% for translational odometry in each axis, and 10% for rotational odometry around each axis. Three different noise levels, drawn from a normal distribution with zero mean and standard deviation $\sigma_1$=0.02, $\sigma_2$=0.04, and $\sigma_3$=0.06 m in each axis per observation, were tested. These noise levels correspond to commercially available LiDAR system, and a RGB-D/stereo system respectively and a third higher value. We acknowledge that noise models for LiDAR and RGB-D/stereo systems are different, in the sense that RGB-D/stereo noise is heavily distance dependent, while the noise range for a LiDAR system is relatively independent of the distance to the measured object. However, for the sake of simplicity and to evaluate the effect of depth information, we model the noise as described above. This is conceptually the same as replacing the depth input in the front-end with a stereo depth estimation algorithm e.g. SPSS [218] or a single image depth estimation for a monocular system e.g. [196]. Point measurement noise is kept at $\sigma_1$ for further tests in this subsection. Effect of different depth information noise levels on the camera and object motion estimation accuracy is shown in Fig. 5.6.

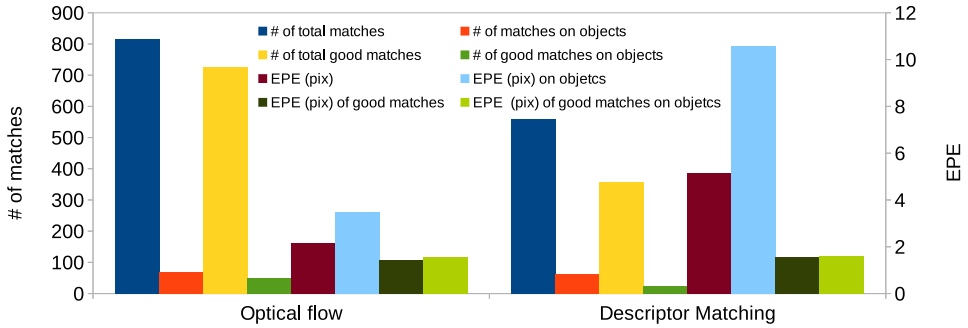



Figure 5.5: **Comparison of optical flow and descriptor matching for feature tracking.**

**Effect of object segmentation** This test aims at evaluating the effect of the object segmentation while using GT feature tracking. We employ MASK-RCNN [5], learning based model, for instance-level object segmentation. We perform evaluation tests of MASK-RCNN on all sequences of vKITTI and KITTI. Results for mean average precision (mAP) and mean intersection over union for predictions only (mIOU_Pred) of the 'car' class are 0.513 and 0.557 for vKITTI and 0.413 and 0.632 for KITTI. Number show good performance, however, testing the effect on camera and object pose change estimation is crucial. MASK-RCNN represented the state-of-the-art instance-level object segmentation algorithm at the time of carrying this research. Effect of MASK-RCNN object segmentation on the camera and object motion estimation accuracy is shown in Fig. 5.6.

**Effect of feature tracking** As the proposed algorithm is sparse feature-based, feature tracking is an essential component of the front-end. In order to test the effect of feature tracking, we first conduct tests on the quantity and quality of feature matches using 1) PWC-Net [118] and 2) feature descriptor matching starting from the same number of detected features using FAST [60] corner detection. Fig. 5.5 shows the number of total and object matches and their corresponding end-point error (EPE), and then extends this test to show these values for "good matches"; matches with less than 3 pixels EPE. PWC-Net represented the state-of-the-art optical flow algorithm at the time of carrying this research. The model is also fine-tuned on KITTI dataset, which makes it a great choice for the application domains of this research. In this experiment, features were described using ORB [61] descriptors. We then explore the effect of feature tracking using optical flow and descriptor matching on the camera and object motion estimation while using GT object segmentation. Results are shown in Fig. 5.6.

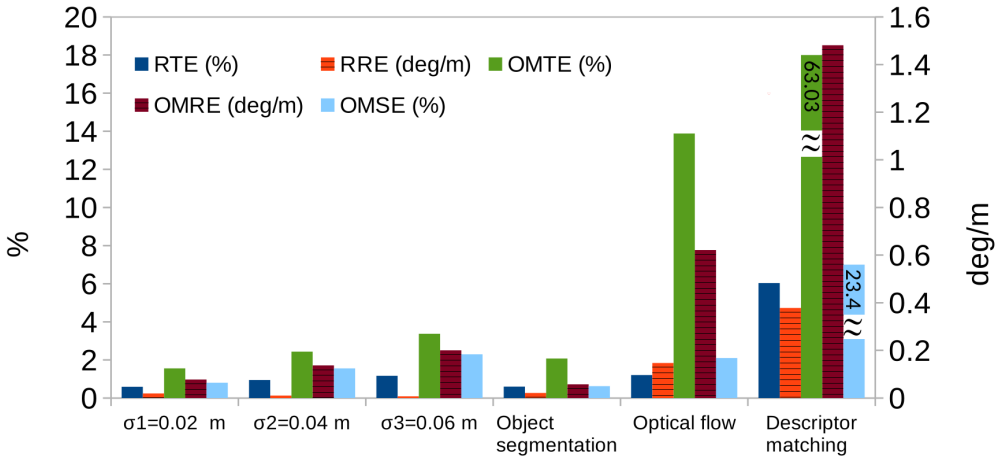



Figure 5.6: **Study of the effect of different front-end components on vKITTI.**

#### 5.4.2.4 Discussion

As shown in Fig. 5.6, the feature tracking is the most crucial component of the front-end and dictates the performance of our feature-based algorithm. Fig. 5.5 and 5.6 show better performance of optical flow over descriptor matching in terms of quantity and quality of features. Object segmentation appears to have the least effect on the estimation quality.

### 5.4.3 Simulated data

#### 5.4.3.1 Description

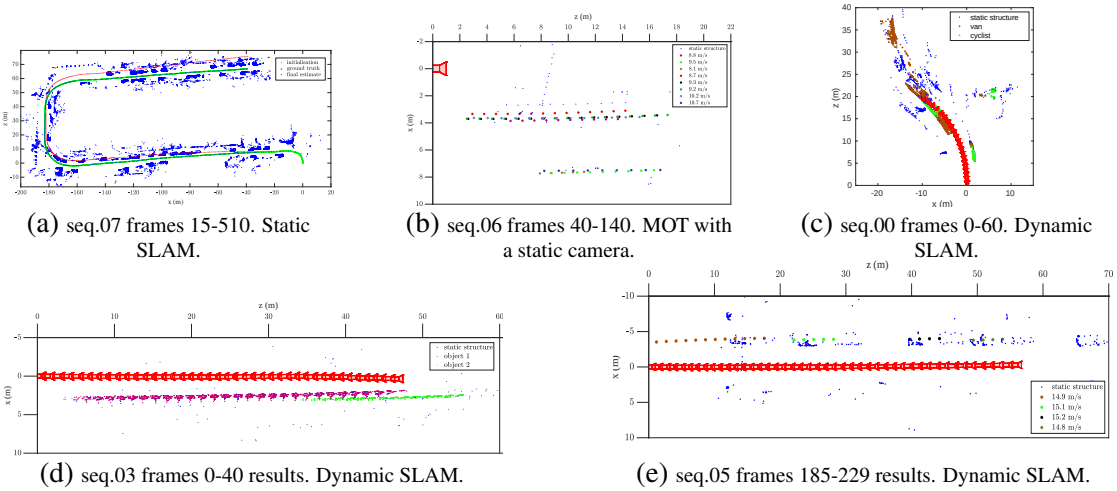
This experiment features a single simulated ellipsoid-shaped object tracked by a robot as it follows a circular motion in an environment with no static structure. The object is simulated to have a constant $\mathrm{SE}(3)$ pose change, and the estimation makes use of this piece of information to estimate for a single object motion for all time steps. The simulation corresponds to a scenario where only moving structure is visible, e.g. a vehicle on a bridge or inside a tunnel occluded by other vehicles driving alongside and failing to track static structure.

#### 5.4.3.2 Goal

This experiment is designed to show that the proposed approach provides good solutions in cases where existing approaches to dynamic SLAM might fail. We compare the proposed algorithm vs. parallel tracking and mapping, e.g. SLAM + Multiple Object Tracking (MOT) [173, 174, 175, 176].

Table 5.1: **Results of applying object-aware dynamic motion integration on simulated data.**

| Error | SLAM+MOT | Ours |
|---|---|---|
| RTE (%) | 4.426 | **3.804** |
| RRE ($^\circ/m$) | 1.34 | **0.486** |
| RSE (%) | 8.019 | **4.177** |
| OMTE (%) | 20.946 | **4.018** |
| OMRE ($^\circ/m$) | 0.349 | **0.055** |



(a) seq.07 frames 15-510. Static SLAM.

(b) seq.06 frames 40-140. MOT with a static camera.

(c) seq.00 frames 0-60. Dynamic SLAM.

(d) seq.03 frames 0-40 results. Dynamic SLAM.

(e) seq.05 frames 185-229 results. Dynamic SLAM.

Figure 5.7: **Sample results on KITTI various sequences.**

#### 5.4.3.3 Discussion

Camera motion in the case of parallel SLAM+MOT is basically a direct integration of odometric measurements. Results in Table 5.1 show the clear advantage of the proposed algorithm that simultaneously estimates the camera and rigid object pose transformations. Improvements are in the range of 80-85% in object pose change estimation. In an extreme case, where no static structure is observed, the proposed algorithm not only improves the object motion estimates but also the camera pose estimation. However, in an environment with enough static structure, both algorithms yield very similar results as shown in the following.

### 5.4.4 KITTI dataset

#### 5.4.4.1 Description

KITTI [194] has been a standard benchmark suite for a number of challenging real-world computer vision tasks. We make use of the KITTI tracking dataset as it provides ground truth object

Table 5.2: **Results of applying object-aware dynamic motion integration on KITTI.**

| | seq.07 | seq.06 | | seq.01 | | seq.03 | | seq.05 | | seq.00 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Error** | **Static Only** | **Static Only** | **Ours** | **Static Only** | **Ours** | **Static Only** | **Ours** | **SLAM + MOT** | **Ours** | **Ours** |
| RTE (%) | 0.039 | 0.000 | 0.000 | 0.248 | 0.248 | 0.016 | 0.016 | 0.022 | 0.025 | 0.020 |
| RRE ($^\circ/m$) | 0.003 | 0.001 | 0.001 | 0.017 | 0.017 | 0.002 | 0.002 | 0.003 | 0.003 | 0.001 |
| RSE (%) | 0.013 | 0.002 | 0.005 | 0.001 | 0.002 | 0.002 | 0.006 | 0.006 | 0.007 | 0.001 |
| OMTE (%) | – | – | 11.646 | – | 10.525 | – | 6.0/59.5 | 23.608 | 23.653 | 42.7/63 |
| OMRE ($^\circ/m$) | – | – | 0.254 | – | 0.555 | – | 0.2/1.0 | 0.472 | 0.473 | 1.2/5.0 |
| OMSE (%) | – | – | 10.587 | – | 5.561 | – | 2.5/2.7 | 11.809 | 11.809 | 20.7/22 |

poses in camera coordinate frame.

#### 5.4.4.2 Goal

This experiment is designed to evaluate and showcase the performance of the proposed algorithm on real-world challenging outdoor scenarios. In relevant dataset sequences, we compare the results to classical static only SLAM (where dynamic objects are considered outliers) and to SLAM+MOT solutions.
To demonstrate the generality of the approach and the fact that the proposed framework performs well in any type of scenarios, we consider three different cases:
a) *Classical SLAM*: A moving robot equipped with an RGB-D camera in a static environment. Seq.07 represents this case and shows that the proposed algorithm performs equally well in a classical scenario with no dynamics and requires no prior knowledge or makes any prior assumptions of the scene.
b) *Multi-object tracking*: A static camera in a dynamic environment as shown in seq.06. For this specific data sequence, we consider a constant pose change assumption. Note that camera pose change errors for this sequence are reported in meters and degrees.
c) *Dynamic SLAM*: A moving robot equipped with an RGB-D camera in a dynamic environment. In here we do the distinction between two sub-scenarios:
• A highway scenario, represented by seq.05, where every vehicle is assumed to have constant motion. This allows us to constraint the problem by assuming a constant pose change model for each detected object in the scene.
• Seq.01, seq.03 and seq.00 represent an intersection and other city driving scenarios, where motion models are difficult to impose. In here, the factor graph formulation allows for the estimation of a new pose change vertex every time step. Seq.03 and seq.00 contain two objects each, therefore the object motion error results are shown separated by a '/'. Seq.00 consists of a "van" and a "cyclist" which slightly violates the rigidity assumption, yet our approach still provides fairly good results.

#### 5.4.4.3 Implementation

The three variants of SLAM; classical, SLAM + MOT, and dynamic are implemented and run in GTSAM [219]. The tracking dataset is thought for camera-only based application, therefore GPS and IMU measurements are fused and further corrupted with noise to simulate odometric measurements available in a robotic (self-driving cars) scenario. The noise values are the same as the ones explained in subsection 5.4.2.3, except seq.07, where twice the noise is added. In autonomous driving, the literature normally distinguishes between depth different ranges for

velocity estimation [16]: near (d $<$ 20 m), medium (20 m $\leq$ d $<$ 45 m) and far range (d $>$ 45 m). In all KITTI experiments presented here, we only consider objects $<$ 22 m of distance to the camera (near and early medium range).

#### 5.4.4.4 Discussion

All results show high accuracy in the estimation of pose change transformations and speeds of moving objects. Results in Table 5.2 show a speed estimation accuracy in the range of 78-97.5%. The second object in seq.03 and seq.00 are particularly hard to process. In seq.03, the second object only occupies a small part of the image, dominated by its wheels having a different motion than the vehicle, yet its speed estimate is reasonable. Seq.00 consists of a van and a cyclist turning at very low speeds ($<$ 5.5 m/s). Their motion estimation is particularly hard because of association errors and the fact that a cyclist is a non-rigid object mostly formed by wheels not obeying the motion model of the object. Although speed errors seem high in percentage, they only account for an average speed error of 0.16 m/s for the van and 0.063 m/s for the cyclist.

## 5.5 Conclusion

In this chapter, we proposed a novel framework that exploits semantic information in the scene with no additional knowledge of the object pose or geometry, to achieve simultaneous localisation, mapping and tracking of dynamic objects. We showed that the use of a novel pose change representation to model the motion of points pertaining to rigid bodies in motion and the integration of such model into a SLAM framework produces accurate results.
An important issue to be analysed in the future, is the computational complexity of SLAM with dynamic objects. In long-term applications, different techniques can be applied to limit the growth of the graph [220, 221]. The estimation could be enhanced even further by assuming a constant motion within a temporal window and making use of this assumption to address occlusions and reduce the problem size. Another possible extension is to use the SLAM back-end estimates to improve the tracking accuracy of the front-end.

Chapter 6

# Visual Dynamic Object-aware SLAM

As this chapter mainly draws on the author's work in collaboration with Jun Zhang, a letter outlining the contribution of the author is included in Appendix C.1 and is agreed upon and singed by both contributors of the work in order to avoid any conflict with the declaration of originality signed at the beginning of the thesis.

## 6.1 Motivation

In the previous chapter, we have shown how to include the dynamics of the scene into the SLAM estimation problem and generate static maps along with tracks and motion estimates of dynamic objects. When dealing with dynamics of the environment, and as opposed to the classical technique of detecting dynamics and rejecting them as outliers, estimating motion of dynamic objects in the scene represents a challenge for SLAM systems operating in real life scenarios, and adds a layer of complexity to the problem in terms of the computational power and memory dedicated to handle the dynamic parts of the environment.

In this chapter, we propose VDO-SLAM, a novel feature-based stereo/RGB-D dynamic *full SLAM system*, that builds on top of the dynamic SLAM approach introduced in chapter 5 and leverages image-based semantic information to simultaneously localise the robot, map the static and dynamic structure, and track motions of rigid objects in real world scenarios. VDO-SLAM is considered an extension of the approach presented in chapter 5 into a full more robust and computationally efficient dynamic SLAM system. The proposed approach draws from the work "Robust Ego and Object 6-DoF Motion Estimation and Tracking" by Jun Zhang *et al.* to: 1) ensure better tracking of features on dynamic objects in the scene, 2) provide better camera and object motion initialisation in the front-end component of the system, and 3) reduce the computational cost of the proposed system by utilising a scene flow-based motion segmentation module,

and couples it with the back-end formulation introduced in "Dynamic SLAM: The Need for Speed" by Mina Henein *et al.*, and *on top* adds 1) a local batch optimisation module that ensures consistency of the returned map, based on the fact that dynamic objects normally appear in one local map, and 2) a way to utilise the back-end estimates to enhance the tracking accuracy of the front-end.

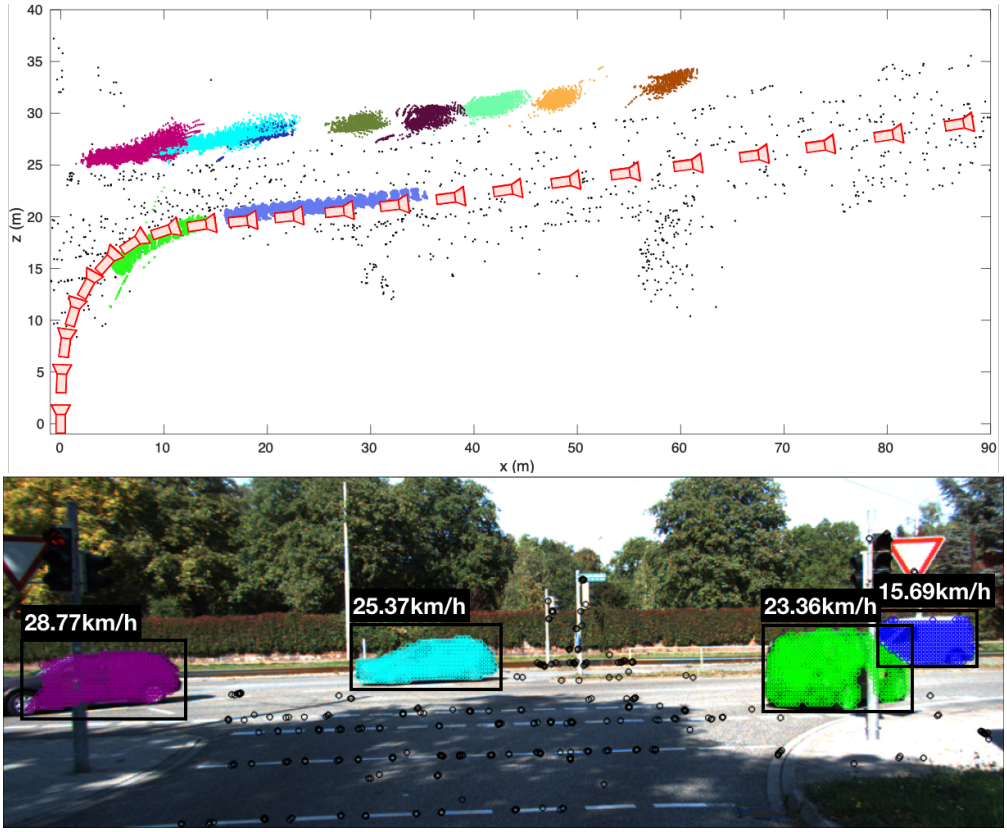



Figure 6.1: **Results of VDO-SLAM system.** (Top) A full map including camera trajectory, static background and moving object structure. (Bottom) Detected points on static background and object body, and estimated object speed. Black circles represents static points, and each object is shown with a different colour.

In summary, the contributions of this chapter are:

- accurate estimation for $\mathrm{SE}(3)$ pose change of dynamic objects that *outperforms* state-of-the-art algorithms,
- a robust method for tracking moving objects exploiting semantic information with the ability to handle indirect occlusions resulting from the failure of semantic object segmentation,
- a demonstrable full system in complex and compelling real-world scenarios.

To the best of our knowledge, this is the first full dynamic SLAM system that is able to achieve motion segmentation, dynamic object tracking, and estimate the camera poses along with the static and dynamic structure, the full $\mathrm{SE}(3)$ pose change of every rigid object in the scene, extract velocity information, and be demonstrable in real-world outdoor scenarios as shown in Fig. 6.1. The performance of the proposed algorithm is demonstrated on real datasets and show capability of the proposed system to resolve rigid object motion estimation and yield motion results that are comparable to the camera pose estimation in accuracy and that outperform state-of-the-art algorithms by an order of magnitude in urban driving scenarios.

## 6.2 Methodology

In the tracking component of the proposed system, shown in Fig. 6.3, the cost function chosen to estimate the camera pose and object motion (described in section 6.2.2) is associated with the 3D-2D re-projection error and is defined on the image plane. Since the noise is better characterised in image plane, this yields more accurate results for camera localisation [222]. Moreover, based on this error term, we propose a novel formulation to jointly optimise the optical flow along with the camera pose and the object motion, to ensure a robust tracking of points (described in section 6.2.2.3). In the mapping module, a 3D error cost function is used to ensure best results of 3D structure and object motions estimation as described in section 6.2.3.

### 6.2.1 Background and notation

#### 6.2.1.1 Points

A point is written in robot/camera frame as ${}^{X_k}\mathbf{m}_k^i = {}^0\mathbf{X}_k^{-1}\,{}^0\mathbf{m}_k^i$. Define $\mathbf{I}_k$ the reference frame associated with the image captured by the camera at time $k$ chosen at the top left corner of the image, and let ${}^{I_k}\mathbf{p}_k^i = \left[u^i, v^i, 1\right] \in \mathbb{E}^2$ be the pixel location on frame $\mathbf{I}_k$ corresponding to the homogeneous 3D point ${}^{X_k}\mathbf{m}_k^i$, which is obtained via the projection function $\pi(\cdot)$ as follows:

$$ {}^{I_k}\mathbf{p}_k^i = \pi({}^{X_k}\mathbf{m}_k^i) = K\,{}^{X_k}\mathbf{m}_k^i\,, \tag{6.1} $$

where $K$ is the camera projection matrix.

The camera and/or object motions both produce an optical flow ${}^{I_k}\boldsymbol{\phi}^i \in \mathbb{R}^2$ that is the displacement vector indicating the motion of pixel ${}^{I_{k-1}}\mathbf{p}_{k-1}^i$ from image frame $I_{k-1}$ to $I_k$, and is given by:

$$ {}^{I_k}\boldsymbol{\phi}^i = {}^{I_k}\tilde{\mathbf{p}}_k^i - {}^{I_{k-1}}\mathbf{p}_{k-1}^i\,. \tag{6.2} $$

Here ${}^{I_k}\tilde{\mathbf{p}}_k^i$ is the correspondence of ${}^{I_{k-1}}\mathbf{p}_{k-1}^i$ in $I_k$. Note that the same notation is used to represent the 2D pixel coordinates $\in \mathbb{R}^2$. In here, we leverage optical flow to find correspondences between consecutive frames.

### 6.2.2 Camera pose and object motion estimation

#### 6.2.2.1 Camera pose estimation

Given a set of static 3D points $\{{}^0\mathbf{m}_{k-1}^i \mid i \in \mathcal{M}, k \in \mathcal{T}\}$ observed at time $k-1$ in global reference frame, and the set of $2D$ correspondences $\{{}^{I_k}\tilde{\mathbf{p}}_k^i \mid i \in \mathcal{M}, k \in \mathcal{T}\}$ in image $\mathbf{I}_k$, the camera pose ${}^0X_k$ is estimated via minimizing the re-projection error:

$$\mathbf{e}_i({}^0X_k) = {}^{I_k}\tilde{\mathbf{p}}_k^i - \pi({}^0X_k^{-1}\,{}^0\mathbf{m}_{k-1}^i)\,. \tag{6.3}$$

Using the Lie-algebra parameterisation of $\mathrm{SE}(3)$ with the substitution of (5.13) into (6.3), the solution of the least squares cost is given by:

$${}^0\mathbf{x}_k^* = \underset{{}^0\mathbf{x}_k}{\operatorname{argmin}} \sum_i^{n_b} \rho_h \left(\mathbf{e}_i^\top({}^0\mathbf{x}_k)\,\Sigma_p^{-1}\,\mathbf{e}_i({}^0\mathbf{x}_k)\right) \tag{6.4}$$

for all $n_b$ visible 3D-2D static background point correspondences between consecutive frames. Here $\rho_h$ is the Huber function [223], and $\Sigma_p$ is the covariance matrix associated with the re-projection error. The estimated camera pose is given by ${}^0X_k^* = \exp({}^0\mathbf{x}_k^{*\wedge})$ and is found using the Levenberg-Marquardt algorithm to solve for (6.4).

#### 6.2.2.2 Object motion estimation

Analogous to the camera pose estimation, a cost function based on re-projection error is constructed to solve for the object motion ${}^0_{k-1}H_k$. Using (5.5), the error term between the re-projection of an object 3D point and the corresponding $2D$ point in image $\mathbf{I}_k$ is:

$$\begin{aligned}\mathbf{e}_i({}^0_{k-1}H_k) :=\;& {}^{I_k}\tilde{\mathbf{p}}_k^i - \pi({}^0X_k^{-1}\;{}^0_{k-1}H_k\,{}^0\mathbf{m}_{k-1}^i)\\ =\;& {}^{I_k}\tilde{\mathbf{p}}_k^i - \pi({}^0_{k-1}G_k\,{}^0\mathbf{m}_{k-1}^i)\,,\end{aligned} \tag{6.5}$$

where ${}^0_{k-1}G_k \in \mathrm{SE}(3)$. Parameterising ${}^0_{k-1}G_k := \exp\left({}^0_{k-1}\mathbf{g}_k^\wedge\right)$ with ${}^0_{k-1}\mathbf{g}_k^\wedge \in \mathrm{se}(3)$, the optimal solution is found via minimising:

$${}^0_{k-1}\mathbf{g}_k^* = \underset{{}^0_{k-1}\mathbf{g}_k}{\operatorname{argmin}} \sum_i^{n_d} \rho_h \left(\mathbf{e}_i^\top({}^0_{k-1}\mathbf{g}_k)\,\Sigma_p^{-1}\,\mathbf{e}_i({}^0_{k-1}\mathbf{g}_k)\right) \tag{6.6}$$

given all $n_d$ visible 3D-2D dynamic point correspondences on an object between frames $k-1$ and $k$. The object motion, ${}^{0}_{k-1}H_k = {}^{0}X_k \; {}^{0}_{k-1}G_k$ can be recovered afterwards.

#### 6.2.2.3 Joint estimation with optical flow

The camera pose and object motion estimation both rely on good image correspondences. Tracking of points on moving objects can be very challenging due to occlusions, large relative motions and large camera-object distances. In order to ensure a robust tracking of points, the technique proposed aims at refining the estimation of the optical flow jointly with the motion estimation.

For camera pose estimation, the error term in (6.3) is reformulated considering (6.2) as:

$$\mathbf{e}_i({}^{0}X_k, {}^{I_k}\boldsymbol{\phi}) = {}^{I_{k-1}}\mathbf{p}^i_{k-1} + {}^{I_k}\boldsymbol{\phi}^i - \pi({}^{0}X_k^{-1}\, {}^{0}\mathbf{m}^i_{k-1}) \,. \tag{6.7}$$

Applying the Lie-algebra parameterisation of $\mathrm{SE}(3)$ element, the optimal solution is obtained via minimising the cost function:

$$\begin{aligned}\{{}^{0}\mathbf{x}^*_k, {}^{k}\boldsymbol{\Phi}^*_k\} = \underset{\{{}^{0}\mathbf{x}_k, {}^{k}\boldsymbol{\Phi}_k\}}{\operatorname{argmin}} \sum_{i}^{n_b} \Big\{ &\rho_h\big(\mathbf{e}_i^\top({}^{I_k}\boldsymbol{\phi}^i)\, \Sigma_\phi^{-1}\, \mathbf{e}_i({}^{I_k}\boldsymbol{\phi}^i)\big) + \\ &\rho_h\big(\mathbf{e}_i^\top({}^{0}\mathbf{x}_k, {}^{I_k}\boldsymbol{\phi}^i)\, \Sigma_p^{-1}\, \mathbf{e}_i({}^{0}\mathbf{x}_k, {}^{I_k}\boldsymbol{\phi}^i)\big)\Big\} \,,\end{aligned} \tag{6.8}$$

where $\rho_h(\mathbf{e}_i^\top({}^{I_k}\boldsymbol{\phi}^i)\, \Sigma_\phi^{-1}\, \mathbf{e}_i({}^{I_k}\boldsymbol{\phi}^i))$ is the regularization term with

$$\mathbf{e}_i({}^{I_k}\boldsymbol{\phi}^i) = {}^{I_k}\hat{\boldsymbol{\phi}}^i - {}^{I_k}\boldsymbol{\phi}^i \,. \tag{6.9}$$

Here ${}^{I_k}\hat{\boldsymbol{\Phi}}^i = \{{}^{I_k}\hat{\boldsymbol{\phi}}^i \mid i \in \mathcal{M}, k \in \mathcal{T}\}$ is the initial optic-flow obtained through classical or learning-based methods, and $\Sigma_\phi$ is the associated covariance matrix. Analogously, the cost function for object motion in (6.6) combining optical flow refinement is given by

$$\begin{aligned}\{{}^{0}_{k-1}\mathbf{g}^*_k, {}^{k}\boldsymbol{\Phi}^*_k\} = \underset{\{{}^{0}_{k-1}\mathbf{g}_k, {}^{k}\boldsymbol{\Phi}_k\}}{\operatorname{argmin}} \sum_{i}^{n_d} \Big\{ &\rho_h\big(\mathbf{e}_i^\top({}^{I_k}\boldsymbol{\phi}^i)\, \Sigma_\phi^{-1}\, \mathbf{e}_i({}^{I_k}\boldsymbol{\phi}^i)\big) + \\ &\rho_h\big(\mathbf{e}_i^\top({}^{0}_{k-1}\mathbf{g}_k, {}^{I_k}\boldsymbol{\phi}^i)\, \Sigma_p^{-1}\, \mathbf{e}_i({}^{0}_{k-1}\mathbf{g}_k, {}^{I_k}\boldsymbol{\phi}^i)\big)\Big\} \,.\end{aligned} \tag{6.10}$$

### 6.2.3 Graph optimisation

The proposed approach formulates the dynamic SLAM as a graph optimisation problem, to refine the camera poses and object motions, and build a global consistent map including static and dynamic structure. We model the dynamic SLAM problem as a factor graph as the one demonstrated in Fig. 6.2.

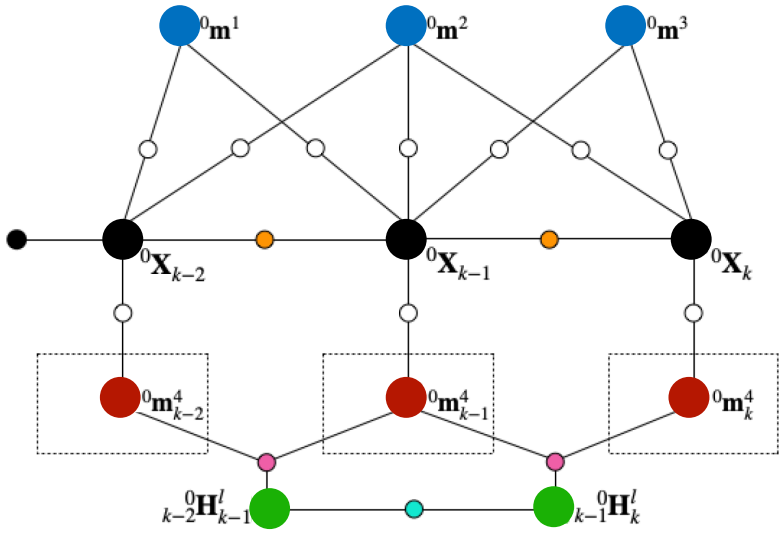



Figure 6.2: **Factor graph representation of an object-aware SLAM system with a moving object.** Large black circles represent the camera poses at different time steps, blue circles represent three static points, ref circles represent the same dynamic point on an object (dashed box) at different time steps and green circles the object pose change between time steps. For ease of visualisation, only one dynamic point is drawn here, however, at the time of estimation, all points on a detected dynamic object are used. A prior unary factor is shown as a smaller black circle, odometry binary factors are shown as orange circles, point measurement binary factors as white circles and point motion ternary factors as magenta circles. A smooth motion binary factor is shown as cyan circle.

Four types of measurements/observations are integrated into a joint optimisation problem; the 3D point measurements, the visual odometry measurements, the motion of points on a dynamic object as explained earlier in section 5.2.2 and the object smooth motion observations. It has been shown that incorporating prior knowledge about the motion of objects in the scene is highly valuable in dynamic SLAM [167, 224]. Motivated by the camera frame rate and the physics laws governing the motion of relatively large objects (vehicles) and preventing their motions to change abruptly, we introduce smooth motion factors to minimise the change in consecutive object motions, with the error term defined as:

$$\mathbf{e}_{l,k}({}^0_{k-2}H^l_{k-1}, {}^0_{k-1}H^l_k) = {}^0_{k-2}H^l_{k-1}{}^{-1}\ {}^0_{k-1}H^l_k. \tag{6.11}$$

The object smooth motion factor $\mathbf{e}_{l,k}({}^0_{k-2}H^l_{k-1}, {}^0_{k-1}H^l_k)$ is used to minimise the change between the object motion at consecutive time steps and is shown as cyan circles in Fig. 6.2. The smooth motion factor is considered a softer version of a constant velocity constraint. While vehicle motion profiles are normally characterised by an acceleration phase, a constant velocity phase and a deceleration phase, it is challenging in a visual SLAM framework to determine when to

start and end each of these phases. We believe that the advances of learning based techniques can provide important cues about the motion/intention of vehicles, and this can integrated in a dynamic visual SLAM framework in future work.

Let $\boldsymbol{\theta}_M = \{{}^0\mathbf{m}_k^i \mid i \in \mathcal{M}, k \in \mathcal{T}\}$ be the set of all 3D points, $\boldsymbol{\theta}_X = \{{}^0\mathbf{x}_k \mid k \in \mathcal{T}\}$ as the set of all camera poses, and $\boldsymbol{\theta}_H = \{{}_{k-1}^0\mathbf{h}_k^l \mid k \in \mathcal{T}, l \in \mathcal{L}\}$ as the set of all object motions. Given $\boldsymbol{\theta} = \boldsymbol{\theta}_X \cup \boldsymbol{\theta}_M \cup \boldsymbol{\theta}_H$ as all the nodes in the graph, and using the Lie-algebra parameterisation of $\mathrm{SE}(3)$ for $X$ and $H$ (substituting (5.13) in (5.10) and (5.11), and substituting (5.14) in (5.12) and (6.11)), the solution of the least squares cost is given by:

$$\begin{aligned}\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \Big\{ & \sum_{i,k}^{n_z} \rho_h\big(\mathbf{e}_{i,k}^\top({}^0\mathbf{x}_k, {}^0\mathbf{m}_k^i)\, \Sigma_z^{-1}\, \mathbf{e}_{i,k}({}^0\mathbf{x}_k, {}^0\mathbf{m}_k^i)\big) \\ & + \sum_{k}^{n_o} \rho_h\big(\mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k)^\top\, \Sigma_o^{-1}\, \mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k)\big) \\ & + \sum_{i,l,k}^{n_g} \rho_h\big(\mathbf{e}_{i,l,k}^\top({}^0\mathbf{m}_k^i, {}_{k-1}^0\mathbf{h}_k^l, {}^0\mathbf{m}_{k-1}^i)\, \Sigma_g^{-1}\, \mathbf{e}_{i,l,k}({}^0\mathbf{m}_k^i, {}_{k-1}^0\mathbf{h}_k^l, {}^0\mathbf{m}_{k-1}^i)\big) \\ & + \sum_{l,k}^{n_s} \rho_h\big(\mathbf{e}_{l,k}({}_{k-2}^0\mathbf{h}_{k-1}^l, {}_{k-1}^0\mathbf{h}_k^l)^\top\, \Sigma_s^{-1}\, \mathbf{e}_{l,k}({}_{k-2}^0\mathbf{h}_{k-1}^l, {}_{k-1}^0\mathbf{h}_k^l)\big)\Big\}\,, \end{aligned} \tag{6.12}$$

where $\Sigma_z$ is the 3D point measurement noise covariance matrix, $\Sigma_o$ is the odometry noise covariance matrix, $\Sigma_g$ is the motion noise covariance matrix with $n_g$ the total number of ternary object motion factors, and $\Sigma_s$ the smooth motion covariance matrix, with $n_s$ the total number of smooth motion factors. The non-linear least squares problem in (6.12) is solved using Levenberg-Marquardt method.

## 6.3 System

In this section, we propose a novel object-aware dynamic SLAM system that robustly estimates both camera and object motions, along with the static and dynamic structure of the environment. The full system overview is shown in Fig. 6.3. The system consists of three main components: image pre-processing, tracking and mapping. The tracking component plays the role of the system's front-end, and the mapping component represents the system's back-end module. The input to the system is stereo or RGB-D images. For stereo images, as a first step, we extract depth information by applying the stereo depth estimation method described in [218] to generate depth maps and the resulting data is treated as RGB-D.

Although this system was initially designed to be an RGB-D system, as an attempt to fully

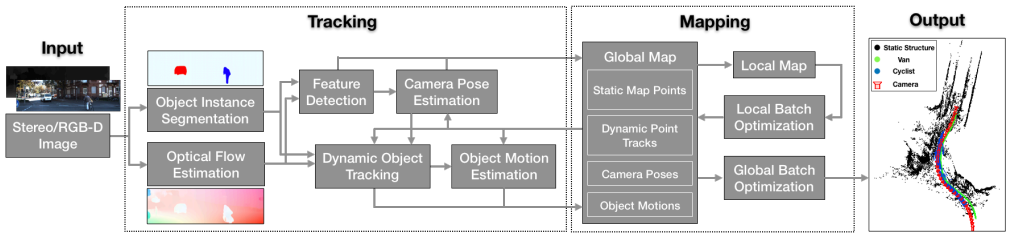



Figure 6.3: **Overview of VDO-SLAM system.** Input images are first pre-processed to generate instance-level object segmentation and dense optical flow. These are then used to track features on static background structure and dynamic objects. Camera poses and object motions estimated from feature tracks are then refined in a global batch optimisation, and a local map is maintained and updated with every new frame. The system outputs camera poses, static structure, tracks of dynamic objects, and estimates of their pose changes over time.

exploit image-based semantic information, we apply single image depth estimation to achieve depth information from monocular camera. Our "learning-based monocular" system is monocular in the sense that only RGB images are used as input to the system, however the estimation problem is formulated using RGB-D data, where the depth is obtained using single image depth estimation.

### 6.3.1 Pre-processing

There are two challenging aspects that this module needs to fulfil. First, to robustly separate static background and objects, and secondly to ensure long-term tracking of dynamic objects. To achieve this, we leverage recent advances in computer vision techniques for instance level semantic segmentation and dense optical flow estimation in order to ensure efficient object motion segmentation and robust object tracking.

#### 6.3.1.1 Object instance segmentation

Instance-level semantic segmentation is used to segment and identify potentially movable objects in the scene. Semantic information constitutes an important prior in the process of separating static and moving object points, e.g., buildings and roads are always static, but cars can be static or dynamic. Instance segmentation helps to further divide semantic foreground into different instance masks, which makes it easier to track each individual object. Moreover, segmentation masks provide a "precise" boundary of the object body that ensures robust tracking of points on the object.

#### 6.3.1.2 Optical flow estimation

The dense optical flow is used to maximise the number of tracked points on moving objects. Most of the moving objects only occupy a small portion of the image. Therefore, using sparse feature matching does not guarantee robust nor long-term feature tracking. Our approach makes use of dense optical flow to considerably increase the number of object points by sampling from all the points within the semantic mask. Dense optical flow is also used to consistently track multiple objects by propagating a unique object identifier assigned to every point on an object mask. Moreover, it allows to recover objects masks if semantic segmentation fails; a task that is extremely difficult to achieve using sparse feature matching.

### 6.3.2 Tracking

The tracking component includes two modules; the camera ego-motion tracking with sub-modules of feature detection and camera pose estimation, and the object motion tracking including sub-modules of dynamic object tracking and object motion estimation.

#### 6.3.2.1 Feature detection

To achieve fast camera pose estimation, we detect a *sparse* set of corner features and track them with optical flow. At each frame, only inlier feature points that fit the estimated camera motion are saved into the map, and used to track correspondences in the next frame. New features are detected and added, if the number of inlier tracks falls below a certain level. These sparse features are detected on static background; image regions excluding the segmented objects.

#### 6.3.2.2 Camera pose estimation

The camera pose is computed using (6.8) for all detected 3D-2D static point correspondences. To ensure robust estimation, a motion model generation method is applied for initialisation. Specifically, the method generates two models and compares their inlier numbers based on re-projection error. One model is generated by propagating the camera previous motion, while the other by computing a new motion transform using P3P [225] algorithm with RANSAC. The motion model that generates most inliers is then selected for initialisation.

#### 6.3.2.3 Dynamic object tracking

The process of object motion tracking consists of two steps. In the first step, segmented objects are classified into static and dynamic. Then we associate the dynamic objects across pairs of consecutive frames.

• Instance-level object segmentation allows us to separate objects from background. Although the algorithm is capable of estimating the motions of all the segmented objects, dynamic object identification helps reduce computational cost of the proposed system. This is done based on scene flow estimation as described in section 3.2.1.4. Unlike optical flow, scene flow−ideally only caused by scene motion−can directly decide whether some structure is moving or not. Ideally, the magnitude of the scene flow vector should be zero for all static 3D points. However, noise or error in depth and matching complicates the situation in real scenarios. To robustly handle this, we compute the scene flow magnitude of all the sampled points on each object. If the magnitude of the scene flow of a certain point is greater than a predefined threshold, the point is considered dynamic. This threshold was set to $0.12$ in all the experiments carried in this chapter. An object is then recognised dynamic if the proportion of "dynamic" points is above a certain level ($30\%$ of total number of points), otherwise static. Thresholds to identify if an object is dynamic were deliberately chosen as mentioned above, to be more conservative as the system is flexible to model a static object as dynamic and estimate a zero motion at every time step, however, the opposite would degrade the system's performance.

• Instance-level object segmentation only provides single-image object labels. Objects then need to be tracked across frames and their motion models propagated over time. We propose to use optical flow to associate point labels across frames. A point label is the same as the unique object identifier on which the point was sampled. We maintain a finite tracking label set $\mathcal{L} \subset \mathbb{N}$, where $l \in \mathcal{L}$ starts from $l = 1$ for the first detected moving object in the scene. The number of elements in $\mathcal{L}$ increases as more moving objects are being detected. Static objects and background are labelled with $l = 0$. Ideally, for each detected object in frame $k$, the labels of all its points should be uniquely aligned with the labels of their correspondences in frame $k - 1$. However, in practice this is affected by the noise, image boundaries and occlusions. To overcome this, we assign all the points with the label that appears most in their correspondences. For a dynamic object, if the most frequent label in the previous frame is $0$, it means that a previously static object starts to move, or a new dynamic object enters the scene, or a previously dynamic and occluded object reappears. In this case, the object is assigned a new tracking label. For partially occluded objects, the same object label is kept across frames, as long as the object is successfully segmented by the instance-level object segmentation algorithm.

#### 6.3.2.4 Object motion estimation

As mentioned above, objects normally appear in small portions in the scene, which makes it hard to get sufficient sparse features to track and estimate their motions robustly. We sample every third point within an object mask, and track them across frames. Similar to the camera pose estimation, only inlier points are saved into the map and used for tracking in the next frame.

When the number of tracked object points decreases below a certain level, new object points are sampled and added. We follow the same method as discussed in section 6.3.2.2 to generate an initial object motion model.

### 6.3.3 Mapping

In the mapping component, a global map is constructed and maintained. Meanwhile, a local map is extracted from the global map, which is based on the current time step and a window of previous time steps. Both maps are updated via a batch optimisation process.

#### 6.3.3.1 Local batch optimisation

We maintain and update a local map. The goal of the local batch optimisation is to ensure accurate camera pose estimates are provided to the global batch optimisation. The camera pose estimation has a big influence on the accuracy of the object motion estimation and the overall performance of the algorithm. The local map is built using a fixed-size sliding window containing the information of the last $n_w$ frames, where $n_w$ is the window size. Local maps share some common information; this defines the overlap between the different windows. We choose to only locally optimise the camera poses and static structure within the window size, as locally optimising the dynamic structure does not bring any benefit to the optimisation unless a hard constraint (e.g. a constant object motion) is assumed within the window. However, the system is able to incorporate static and dynamic structure in the local mapping if needed. When a local map is constructed, similarly, a factor graph optimisation is performed to refine all the variables within the local map, and then update them back into the global map.

#### 6.3.3.2 Global batch optimisation

The output of the tracking component and the local batch optimisation consists of the camera pose, the object motions and the inlier structure. These are saved in a global map that is constructed with all the previous time steps and is continually updated with every new frame. A factor graph is constructed based on the global map after all input frames have been processed. To effectively explore the temporal constraints, only points that have been tracked for more than 3 instances are added into the factor graph. The graph is formulated as an optimisation problem as described in section 6.2.3. The optimisation results serve as the output of the whole system.

#### 6.3.3.3 From mapping to tracking

Maintaining the map provides history information to the estimate of the current state in the tracking module, as shown in Fig. 6.3 with blue arrows going from the global map to multiple

components in the tracking module of the system. Inlier points from the last frame are leveraged to track correspondences in the current frame and estimate camera pose and object motions. The last camera and object motion also serve as possible prior models to initialise the current estimation as described in section 6.3.2.2 and 6.3.2.4. Furthermore, object points help associate semantic masks across frames to ensure robust tracking of objects, by propagating their previously segmented masks in case of "indirect occlusion" resulting from the failure of semantic object segmentation.

## 6.4 Experiments

We evaluate the proposed method in terms of camera motion, object motion and velocity, as well as object tracking performance. The evaluation is done on the Oxford Multimotion Dataset [226] for indoor, and KITTI Tracking dataset [227] for outdoor scenarios. We first evaluate our system versus ORB-SLAM2 [215] as a baseline for the camera pose estimation. We also compare results to two state-of-the-art methods, MVO [178] and CubeSLAM [13], to prove the better performance of VDO-SLAM.

### 6.4.1 System setup

We adopt a learning-based instance-level object segmentation, Mask R-CNN [5], to generate object segmentation masks. The model of this method is trained on COCO dataset [228], and is directly used without any fine-tuning. MASK-RCNN represented the state-of-the-art instance-level object segmentation algorithm at the time of carrying this research. For dense optical flow, we leverage a state-of-the-art method; PWC-Net [118]. PWC-Net represented the state-of-the-art dense optical flow algorithm at the time of carrying this research. The model is trained on FlyingChairs dataset [229], and then fine-tuned on Sintel [230] and KITTI training datasets [194], which makes it a great choice for the application domains of this work. To generate depth maps for a "monocular" version of the proposed system, we apply a learning-based monocular depth estimation method, MonoDepth2 [196]. The model is trained on Depth Eigen split [231] excluding the tested data in the experiments reported here. Feature detection is done using FAST [60] implemented in [61].

### 6.4.2 Error metrics

We use a pose change error metric to evaluate the estimated $\mathrm{SE}(3)$ motion, i.e., given a ground truth motion transform $T$ and a corresponding estimated motion $\hat{T}$, where $T \in \mathrm{SE}(3)$ could be either a camera relative pose or an object motion. The pose change error is computed as:

$E = \hat{T}^{-1}\, T$. The translational error $E_t$ is computed as the $L_2$ norm of the translational component of $E$. The rotational error $E_r$ is calculated as the angle of rotation in an axis-angle representation of the rotational component of $E$. For different camera time steps and different objects in a sequence, we compute the root mean squared error (RMSE) for camera poses and object motions, respectively. The object pose change in body-fixed frame is obtained by transforming the pose change ${}_{k-1}^{0}H_k$ in the inertial frame into the body frame using the object pose ground-truth

$$ {}_{k-1}^{L_{k-1}}H_k = {}^{0}L_{k-1}^{-1}\, {}_{k-1}^{0}H_k\, {}^{0}L_{k-1}. \tag{6.13} $$

We also evaluate the object speed error $E_s$ between the estimated $\hat{\mathbf{v}}$ and the ground truth $\mathbf{v}$ velocities, calculated as: $E_s = |\hat{\mathbf{v}}| - |\mathbf{v}|$. The linear velocity extraction from the object $\mathrm{SE}(3)$ pose change was described in section 5.2.1.3 in chapter 5. For different objects tracked over temporal frames, we also compute the RMSE as an error metric.

### 6.4.3 Oxford multi-motion dataset

The recent Oxford Multi-motion Dataset [226] contains sequences from a moving stereo or RGB-D camera sensor observing multiple swinging boxes or toy cars in an indoor scenario. Ground truth trajectories of the camera and moving objects are obtained via a Vicon motion capture system. We only choose the swinging boxes sequences for evaluation, since results of real driving scenarios are evaluated on KITTI dataset. Table 6.1 shows results compared to the state-of-the-art MVO [178]. As MVO is a visual odometry system without global refinement, we switch off the batch optimisation module of the proposed system and generate results for fair comparison. We use the error metrics described in section 6.4.2.

Table 6.1: **Comparison versus MVO [178] for camera and object motion estimation accuracy** on the sequence of swinging_4_unconstrained sequence in Oxford Multi-motion dataset. Bold numbers indicate the better results.

| | Proposed | | MVO | |
|---|---|---|---|---|
| | $E_r$(deg) | $E_t$(m) | $E_r$(deg) | $E_t$(m) |
| Camera | **0.4525** | **0.0163** | 1.0742 | 0.0338 |
| Top-left Swinging Box | **1.0175** | **0.0302** | 2.9025 | 0.0685 |
| Top-right Swinging and rotating Box | **1.3567** | 0.0229 | 1.4540 | **0.0212** |
| Bottom-left Swinging Box | **1.6356** | **0.0290** | 2.9765 | 0.0502 |
| Bottom-right Rotating Box | 1.7507 | 0.0261 | **1.3489** | **0.0117** |

Overall, the proposed method achieves better accuracy in 7 out of 10 error indexes for camera pose estimation and motion estimation of the 4 moving boxes. In particular, our method

achieves $50\%$ improvements in estimating the camera pose and motion of the swinging boxes, top-left and bottom-left. We obtain slightly higher errors when there is spinning motion of the object observed, in particular the top-right swinging and rotating box, and the bottom-right rotating box. Interestingly, the proposed algorithm performs worse than MVO [178] in these cases, and we believe that this is due to the challenging tracking of features on spinning objects using optical flow. As VDO-SLAM is feature-based, it is best suited for the motion of relatively large objects (e.g. urban driving scenarios) where features exhibit a significant motion between frames. The consequence of this is poor estimation of point motion and consequent degradation of the overall object tracking performance. Even with the associated performance loss for rotating objects, the benefits of dense optical flow motion estimation is clear in the other metrics.

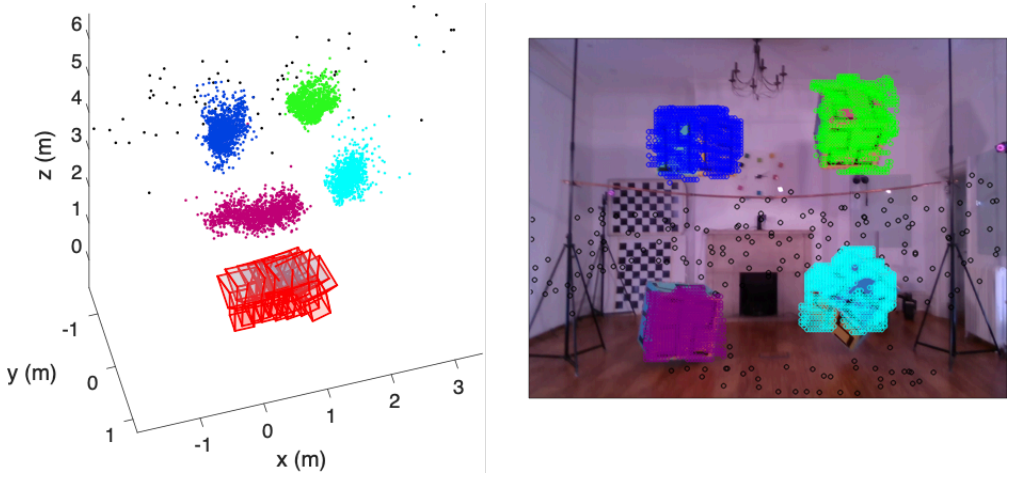



Figure 6.4: **Qualitative results of the proposed method on Oxford Multi-motion Dataset.** (Left) The 3D map including camera trajectory, static structure and tracks of dynamic points. (Right) Detected points on static background and object body. Black color corresponds to static points and features on each object are shown in a different color.

An illustrative result of the map output of the proposed system on Oxford Multi-motion Dataset is shown in Fig. 6.4. Tracks of dynamic features on swinging boxes visually correspond to the actual motion of the boxes. This can be clearly seen in the swinging motion of the bottom-left box shown with purple colour in Fig. 6.4.

### 6.4.4 KITTI tracking dataset

The KITTI Tracking Dataset [227] contains 21 sequences in total with ground truth information about camera and object poses. Among these sequences, some are not included in the evaluation of the proposed system; as they contain no moving objects (static only scenes) or only contain pedestrians that are non-rigid objects, which is outside the scope of this work.

Table 6.2: **Comparison versus ORB-SLAM2 [215] for camera pose estimation accuracy** on nine sequences with moving objects drawn from the KITTI dataset. Bold numbers indicate the better results.

| | Proposed | | ORB-SLAM2 | |
|---|---|---|---|---|
| seq. | $E_r$(deg) | $E_t$(m) | $E_r$(deg) | $E_t$(m) |
| 00 | 0.05 | **0.05** | **0.04** | 0.06 |
| 01 | **0.04** | 0.12 | 0.05 | **0.04** |
| 02 | **0.02** | 0.04 | 0.04 | **0.03** |
| 03 | **0.04** | 0.09 | 0.07 | **0.04** |
| 04 | **0.05** | 0.11 | 0.07 | **0.06** |
| 05 | **0.02** | 0.10 | 0.06 | **0.03** |
| 06 | 0.05 | **0.02** | **0.02** | 0.04 |
| 18 | **0.02** | 0.07 | 0.05 | **0.03** |
| 20 | **0.03** | 0.16 | 0.11 | **0.07** |

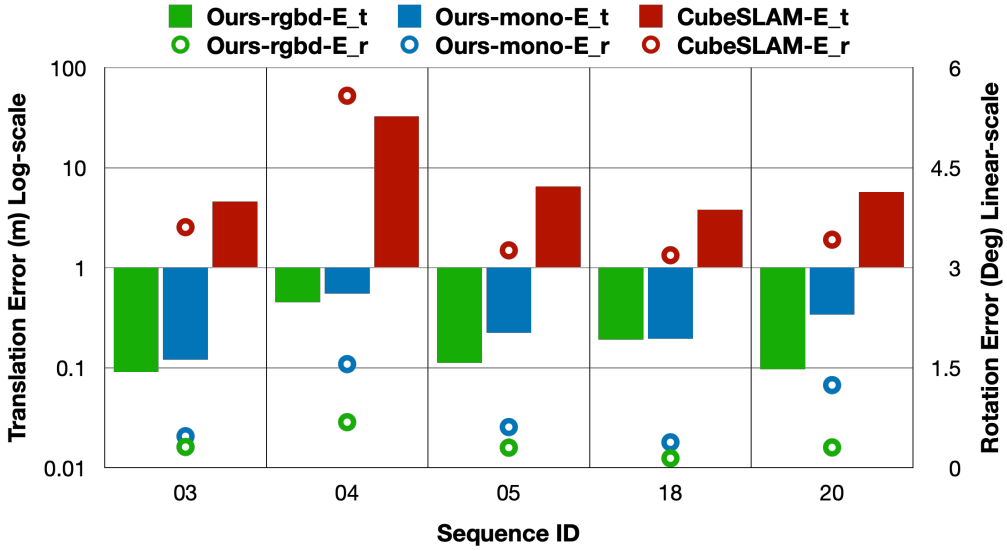



Figure 6.5: **Accuracy of object motion estimation of the proposed method compared to CubeSLAM [13].** The color bars refer to translation error that is corresponding to the left Y-axis in log-scale. The curves refer to rotation error, which corresponds to the right Y-axis in linear-scale.

#### 6.4.4.1 Camera and object motion

Table 6.2 shows results of camera pose estimation in 9 sequences compared to ORB-SLAM2 [215]. While the biggest performance advantage of our algorithm is in the object motion estimation, this experiment is provided to show that our system achieves comparable camera pose estimation results to state-of-the-art SLAM systems [215].

Table 6.3 demonstrates results of both camera and object motion estimation in 9 sequences, compared to CubeSLAM [13], with results of 5 sequences provided by the authors. We ini-

Table 6.3: **Comparison versus CubeSLAM [13] for camera and object motion estimation accuracy** on nine sequences with moving objects drawn from the KITTI dataset. Bold numbers indicate the better result.

| | Proposed RGB-D | | | | Proposed Monocular | | | | CubeSLAM | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Camera | | Object | | Camera | | Object | | Camera | | Object | |
| seq | $E_r$(deg) | $E_t$(m) | $E_r$(deg) | $E_t$(m) | $E_r$(deg) | $E_t$(m) | $E_r$(deg) | $E_t$(m) | $E_r$(deg) | $E_t$(m) | $E_r$(deg) | $E_t$(m) |
| 00 | **0.0512** | **0.0541** | **1.0429** | **0.1061** | 0.0708 | 0.0591 | 1.7242 | 0.3081 | | | | |
| 01 | **0.0368** | **0.1201** | **1.2504** | **0.2129** | 0.1133 | 0.3484 | 1.9514 | 0.5389 | | | | |
| 02 | **0.0152** | **0.0427** | **1.4311** | **0.2358** | 0.0285 | 0.0584 | 2.3104 | 0.5729 | | | | |
| 03 | **0.0377** | **0.0866** | **0.3782** | **0.1055** | 0.1010 | 0.1628 | 0.4722 | 0.1201 | 0.0498 | 0.0929 | 3.6085 | 4.5947 |
| 04 | **0.0476** | **0.1127** | **1.0211** | **0.3376** | 0.1539 | 0.5096 | 1.5543 | 0.5503 | 0.0708 | 0.1159 | 5.5803 | 32.5379 |
| 05 | **0.0246** | 0.0982 | **0.4482** | **0.1098** | 0.0576 | 0.1263 | 0.6103 | 0.2245 | 0.0342 | **0.0696** | 3.2610 | 6.4851 |
| 06 | **0.0477** | **0.0187** | **1.2156** | **0.1304** | 0.0511 | 0.0296 | 3.4771 | 0.3187 | | | | |
| 18 | **0.0192** | 0.0715 | **0.1470** | **0.1131** | 0.0861 | 0.2179 | 0.3818 | 0.1953 | 0.0433 | **0.0510** | 3.1876 | 3.7948 |
| 20 | **0.0293** | **0.1643** | **0.5401** | **0.0944** | 0.0872 | 0.4092 | 1.2395 | 0.3421 | 0.1348 | 0.1888 | 3.4206 | 5.6986 |

tially tried to evaluate CubeSLAM ourselves with the default provided parameters, however errors were much higher, and hence we only report results of the sequences provided by the authors of CubeSLAM after some correspondences. As CubeSLAM is for monocular camera, we also compute results of a learning-based monocular version of our method (as mentioned in section 6.3) for fair comparison.

Overall, both the proposed RGB-D and learning-based monocular methods obtain high accuracy in both camera and object motion estimation. Compared to CubeSLAM, our RGB-D version gets lower errors in camera motion, while the learning-based monocular version slightly higher. Nevertheless, both versions obtain consistently lower errors in object motion estimation. In particular, as demonstrated in Fig. 6.5, the translation and rotation errors in CubeSLAM are all above 3 meters and 3 degrees, with errors reaching 32 meters and 5 degrees in extreme cases respectively. However, the translation errors of the proposed system vary between 0.1-0.3 meters and rotation errors between 0.2-1.5 degrees in case of RGB-D, and 0.1-0.3 meters, and 0.4-3 degrees in case of learning-based monocular, which indicates that our object motion estimation achieves an order of magnitude improvements.

#### 6.4.4.2 Object tracking and velocity

We also demonstrate the performance of tracking dynamic objects, and show results of object speed estimation, which is an important information for autonomous driving applications. Fig. 6.6 illustrates results of object tracking length and object speed for some selected objects (tracked for over 20 frames) in all the tested sequences. Our system is able to track most objects for more than 80% of their occurrence in the sequence. Moreover, the estimated objects speed is always consistently close to the ground truth.

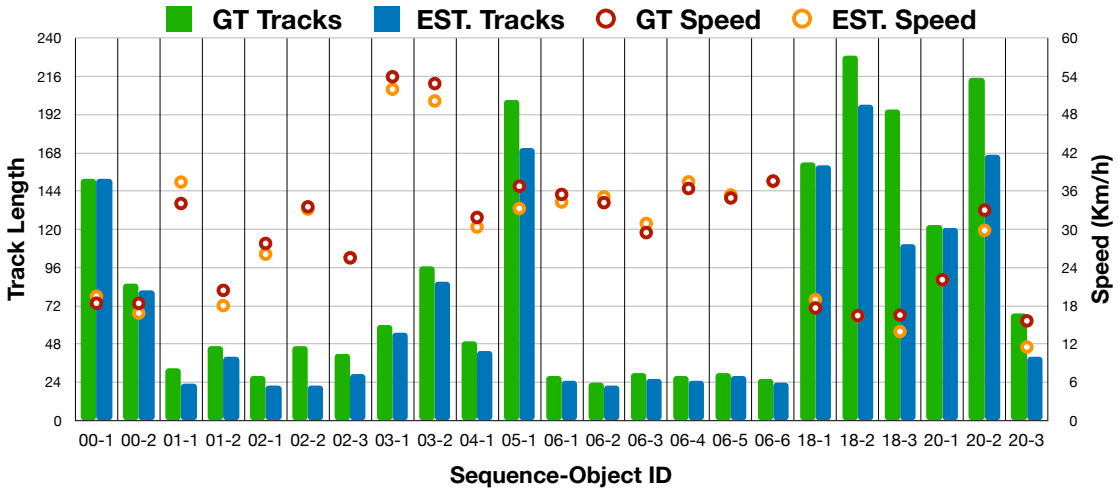



Figure 6.6: **Tracking performance and speed estimation.** Results of object tracking length and object speed for some selected objects (tracked for over 20 frames), due to limited space. The color bars represent the length of object tracks, which is corresponding to the left Y-axis. The circles represent object speeds, which is corresponding to the right Y-axis. GT refers to ground truth, and EST. refers to estimated values.

#### 6.4.4.3 Qualitative results

Fig. 6.7 illustrates the output of the proposed system for three of the KITTI sequences. The proposed system is able to output the camera poses, along with the static structure and dynamic tracks of every detected moving object in the scene in a spatio-temporal map representation.

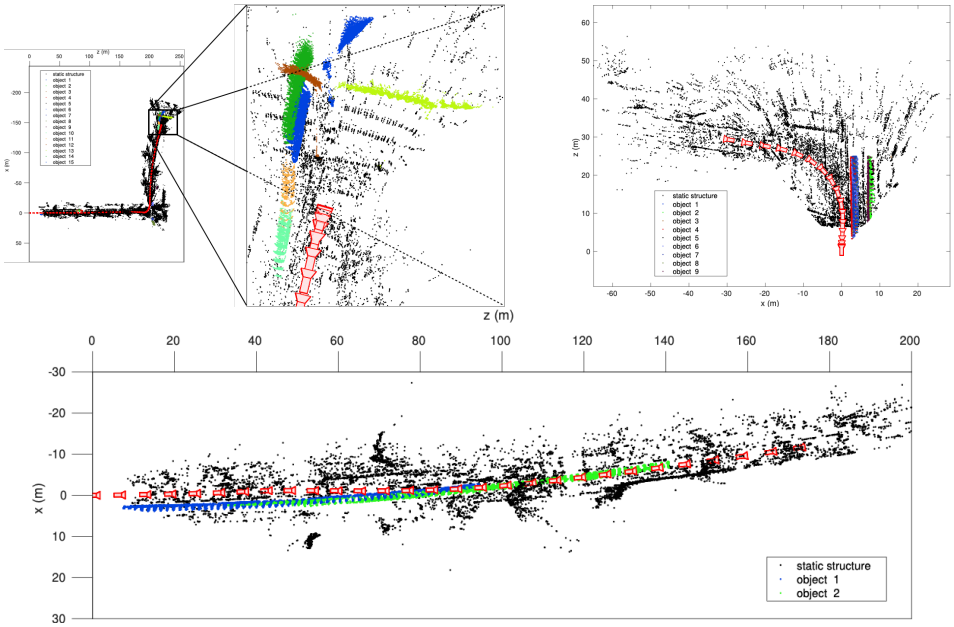



Figure 6.7: **Illustration of system output; a dynamic map with camera poses, static background structure, and tracks of dynamic objects.** Sample results of VDO-SLAM on KITTI sequences. Black represents static background, and each detected object is shown in a different colour. Top left figure represents seq.01 and a zoom-in on the intersection at the end of the sequence, top right figure represents seq.06 and bottom figure represents seq.03.

### 6.4.5 Discussion

Apart from the extensive evaluation in section 6.4.4 and 6.4.3, we also provide detailed experimental results to prove the effectiveness of key modules in the proposed system. Finally, the computational cost of the proposed system is discussed.

#### 6.4.5.1 Robust tracking of points

The graph optimisation explores the spacial and temporal information to refine the camera poses and the object motions, as well as the static and dynamic structure. This process requires robust tracking of good points in terms of both quantity and quality. This was achieved by refining the estimated optical flow jointly with the motion estimation, as discussed in section 6.2.2.3. The effectiveness of joint optimisation is shown by comparing a baseline method that only optimises for the motion (Motion Only) using (6.4) for camera motion or (6.6) for object motion, and the improved method that optimises for both the motion and the optical flow (Joint) using (6.8) or (6.10). Table 6.4 demonstrates that the joint method obtains considerably more points that are tracked for long periods.

Table 6.4: **Quantitative point tracking performance of joint estimation with optical flow versus motion only estimation.** The number of points tracked for more than five frames on the nine sequences of the KITTI dataset. Bold numbers indicate the better results. Underlined bold numbers indicate an order of magnitude increase in number.

| | Background | | Object | |
|---|---|---|---|---|
| seq | Motion Only | Joint | Motion Only | Joint |
| 00 | 1798 | **12812** | 1704 | **7162** |
| 01 | 237 | **5075** | 907 | **4583** |
| 02 | 7642 | **10683** | 52 | **1442** |
| 03 | 778 | **12317** | 343 | **3354** |
| 04 | 9913 | **25861** | 339 | **2802** |
| 05 | 713 | **11627** | 2363 | **2977** |
| 06 | 7898 | **11048** | 482 | **5934** |
| 18 | 4271 | **22503** | 5614 | **14989** |
| 20 | 9838 | **49261** | 9282 | **13434** |

Using the tracked points given by the joint estimation process leads to better estimation of both camera pose and object motion. An improvement of about $15\%$ to $20\%$ in translation and rotation errors was observed over the nine sequences of the KITTI dataset shown above. See Table 6.5.

Table 6.5: **Effect of joint estimation with optical flow versus motion only estimation on the camera and object motion estimation accuracy.** Average camera pose and object motion errors over the nine sequences of the KITTI dataset. Bold numbers indicate the better results.

| | Motion Only | | Joint | |
|---|---|---|---|---|
| | $E_r$(deg) | $E_t$(m) | $E_r$(deg) | $E_t$(m) |
| Camera | 0.0412 | 0.0987 | **0.0344** | **0.0854** |
| Object | 1.0179 | 0.1853 | **0.8305** | **0.1606** |

#### 6.4.5.2 Robustness against non-direct occlusion

The mask segmentation may fail in some cases, due to direct or indirect occlusions (illumination change, etc.). Thanks to the mask propagation method described in section 6.3.3.3, the proposed system is able to handle mask failure cases caused by indirect occlusions. Fig. 6.8 demonstrates an example of tracking a white van for $80$ frames, where the mask segmentation fails in $33$ frames. Despite the object segmentation failure, the system is still continuously able to track the van, and estimate its speed with an average error of $2.64$ km/h across the whole sequence. Speed errors in the second half of the sequence are higher due to partial direct occlusions, and increased distance to the object getting farther away from the camera.

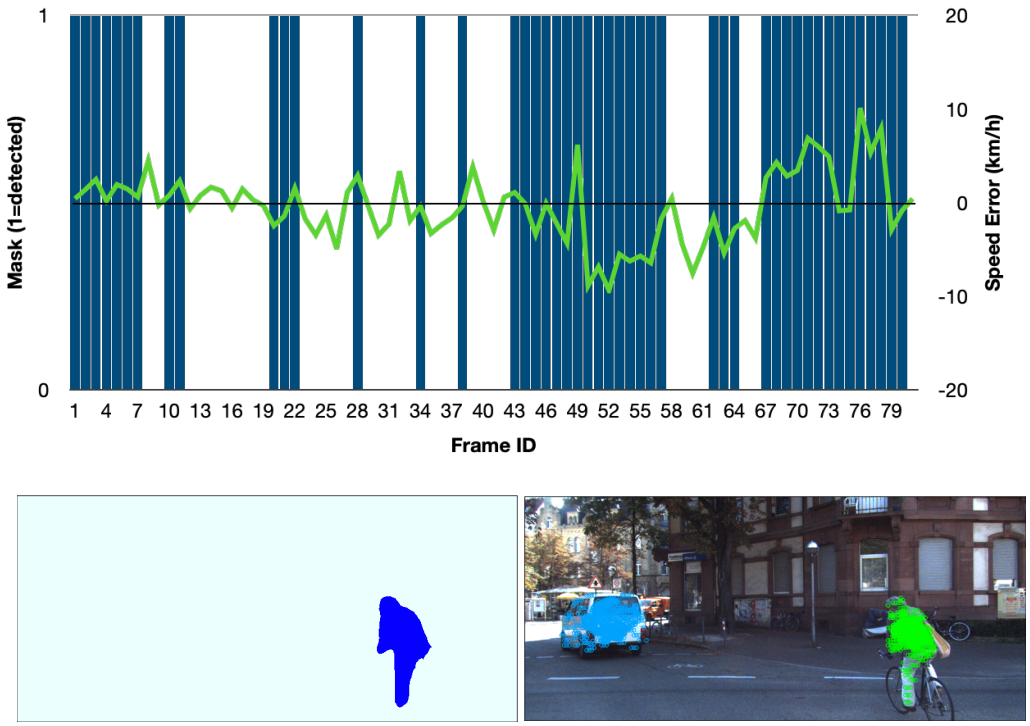



Figure 6.8: **Robustness in tracking performance and speed estimation in case of semantic segmentation failure.**
An example of tracking performance and speed estimation for a white van (ground-truth average speed 20km/h) in seq.00. (Top) Blue bars represent a successful object segmentation, and green curves refer to the object speed error. (Bottom-left) An illustration of semantic segmentation failure on the van. (Bottom-right) Result of propagating the previously tracked features on the van.

#### 6.4.5.3 Global refinement of object motion

Initial object motion estimation (in the tracking component of the system) is independent between frames, since it is purely related to the sensor measurements. As illustrated in Fig. 6.9, the blue curve describes an initial object speed estimate of a wagon observed for 55 frames in sequence 03 of the KITTI tracking dataset. As seen in the figure, the speed estimation is not smooth and large errors occur towards the second half of the sequence. This is mainly caused by the increased distance to the object getting farther away from the camera, and its structure only occupying a small portion of the scene. In this case, the object motion estimation from sensor measurements solely becomes challenging and error-prone. Therefore, we formulate a factor graph and refine the motions together with the static and dynamic structure as discussed in section 6.2.3. The green curve in Fig. 6.9 shows the object speed results after the global refinement, which becomes smoother in the first half of the sequence and is significantly improved in the

second half.

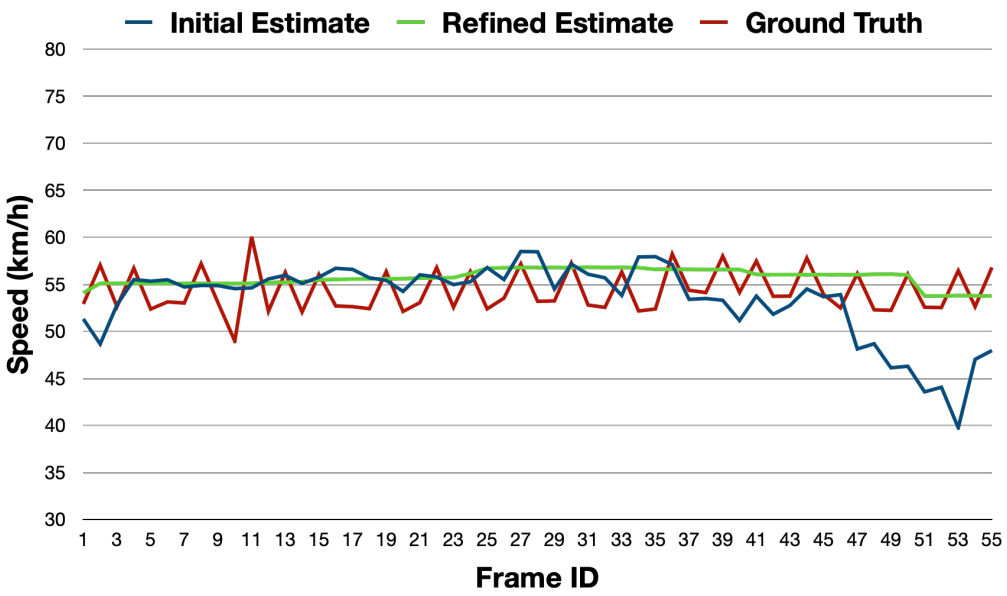



Figure 6.9: **Global refinement effect on object motion estimation.** The initial (blue) and refined (green) estimated speeds of a wagon in seq.03, travelling along a straight road, compared to the ground truth speed (red).

#### 6.4.5.4 Computational analysis

Finally, we provide the computational analysis of the proposed system. The experiments are carried out on an Intel Core i7 2.6 GHz laptop computer with 16 GB RAM. The object semantic segmentation and dense optical flow computation times depend on the GPU power and the CNN model complexity. Many current state-of-the-art algorithms can run in real time [232, 233]. In this work, the semantic segmentation and optical flow results are produced off-line as input to the system. The SLAM system is implemented in C++ on CPU using a modified version of g2o as a back-end [54]. We show the computational time in Table 6.6 for both datasets. In the local batch optimisation, the window size is set to 20 frames with an overlap of 4 frames. The time cost of every system component is averaged over all frames, and sequences. Overall, the tracking part of the proposed system is able to run at the frame rate of 5-8 fps depending on the number of detected moving objects, which can be improved by employing parallel implementation. The runtime of the global batch optimisation strongly depends on the amount of camera poses (number of frames), and objects (density in terms of the number of dynamic objects observed per frame) present in the scene.

Table 6.6: **Runtime of different system components for both datasets.** The time cost of every component is averaged over all frames and sequences, except for the object motion estimation that is averaged over the number of objects, and local batch optimisation that is averaged over the number of frames.

| Dataset | Tasks | Runtime (mSec) |
|---|---|---|
| KITTI | Feature Detection | 16.2550 |
| | Camera Pose Estimation | 52.6542 |
| | Dynamic Object Tracking | 11.4828 |
| | Object Motion Estimation (avg/object) | 22.9081 |
| | Map and Mask Updating | 22.1830 |
| | Local Batch Optimisation (avg /frame) | 18.2828 |
| OMD | Feature Detection | 7.5220 |
| | Camera Pose Estimation | 32.0909 |
| | Dynamic Object Tracking | 28.0535 |
| | Object Motion Estimation (avg/object) | 19.5280 |
| | Map and Mask Updating | 30.3153 |
| | Local Batch Optimisation (avg /frame) | 15.3414 |

## 6.5 Conclusion

In this chapter, we have presented VDO-SLAM, a novel dynamic feature-based SLAM system that exploits image-based semantic information in the scene with no additional knowledge of the object pose or geometry, to achieve simultaneous localisation, mapping and tracking of dynamic objects. The system consistently shows robust and accurate results on indoor and challenging outdoor datasets, and achieves state-of-the-art performance in object motion estimation. We believe the high performance accuracy achieved in object motion estimation is due to the fact that our system is a feature-based system. Feature points remain to be the easiest to detect, track and integrate within a SLAM system, and that require the front-end to have no additional knowledge about the object model, or explicitly provide any information about its pose.

Due to the fact that our VDO-SLAM system is a feature-based system, the performance of the system is sensitive to the quality of feature detection and tracking which is challenging in dynamic environments. While optical flow learning-based methods seem to have achieved great results in indoor and outdoor scenarios, our algorithm might experience a performance degradation in situations where optical flow might suffer, e.g. spinning motion of objects in Oxford multi-motion dataset. The object size in the image also represents a challenge to our system. A failure in object segmentation due to the increased distance of the object to the camera and the small portion of the image the object occupies will result in no motion estimates for such objects. Even if distant objects are detected and a segmentation mask is available, feature detection and tracking on such objects is very challenging. The apparent motion of

distant objects is hard to observe and thus pixel displacements on such objects are challenging to estimate. This might result in an overall degraded performance of our system. We however believe that this is not a great problem, as distant objects do not contribute much to the current dynamic map representation and their motion is not of great importance for the robot state.

An important issue to be reduced is the computational complexity of SLAM with dynamic objects. In long-term applications, different techniques can be applied to limit the growth of the graph [220, 221]. In fact, history summarisation/deletion of map points pertaining to dynamic objects observed far in the past seems to be a natural step towards a long-term SLAM system in highly dynamic environments.

Chapter 7

# Open-source Code

One of the contributions of this thesis is an open source code that is made available for the community to test, develop and extend the work achieved in this thesis.
We present 2 systems implementation; the first system is a MATLAB implementation, and was initially developed to integrate structure information into a SLAM framework (Entity SLAM in section 7.1.1), and then extended to incorporate dynamic information (DO-SLAM in section 7.1.2). The second system is a C++ implementation that represents a single full system to include semantic and dynamic information into a SLAM framework (VDO-SLAM in section 7.2.1).

## 7.1 MATLAB implementation

### 7.1.1 Entity SLAM

We first introduce "Entity SLAM" [https://github.com/MinaHenein/Entity-SLAM/wiki], an extension of classical graph-based SLAM to include higher level entities. We mainly focus on the integration of planar surfaces and introduce planarity (point-plane factor (4.3) in chapter 4), orthogonality and parallelism (plane-plane factors (4.4) in chapter 4) constraints to improve the estimation accuracy in Manhattan-like environments.
The framework is implemented in MATLAB and the object oriented design allows for extension to other primitives and/or information to be integrated.
Modules in the front-end allow the generation of various camera trajectories and map/environment configurations, and the corresponding measurements. The front-end and back-end modules are decoupled, and communicate through *graph files*. This allows for plug-and-play modules to replace the front-end and/or back-end for testing/developing reasons.
The back-end implements a NLS optimisation. Gauss-Newton and Levenberg-Marquardt non-

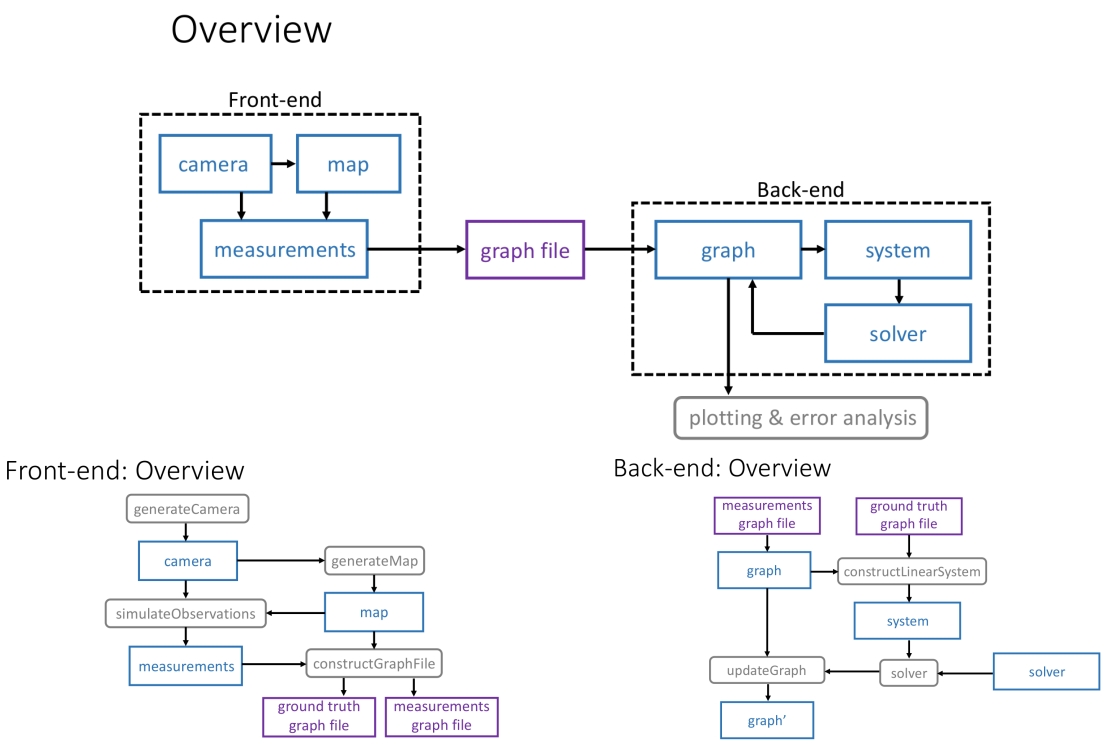



Figure 7.1: **Entity SLAM code overview.**

linear solvers are implemented. The general implementation overview is shown in Fig. 7.1 and a documentation is included for ease of use.

### 7.1.2 Dynamic Object-aware SLAM

The second source code that is made publicly available is "Dynamic Object-aware SLAM" (DO-SLAM) [https://github.com/MinaHenein/do-slam], a feature-based object-aware dynamic SLAM system. This work is an extension of a MATLAB framework (Entity SLAM) that is able to integrate not only simple point measurements but also additional available information about the environment into a single SLAM framework. Added information could be in the form of structural, geometric, kinematic, dynamic or even semantic constraints, although only structure information is implemented in Entity SLAM.
The framework consists of:

1. a simulation component that can reproduce several dynamic environments as shown in Fig. 7.2;

2. a front-end that generates the data for the SLAM problem by tracking features, objects

and providing point associations using simulated or real data inputs;

3. a back-end component that includes different non-linear solvers for batch and incremental processing.

The same code architecture as shown in Fig. 7.1 holds for DO-SLAM.

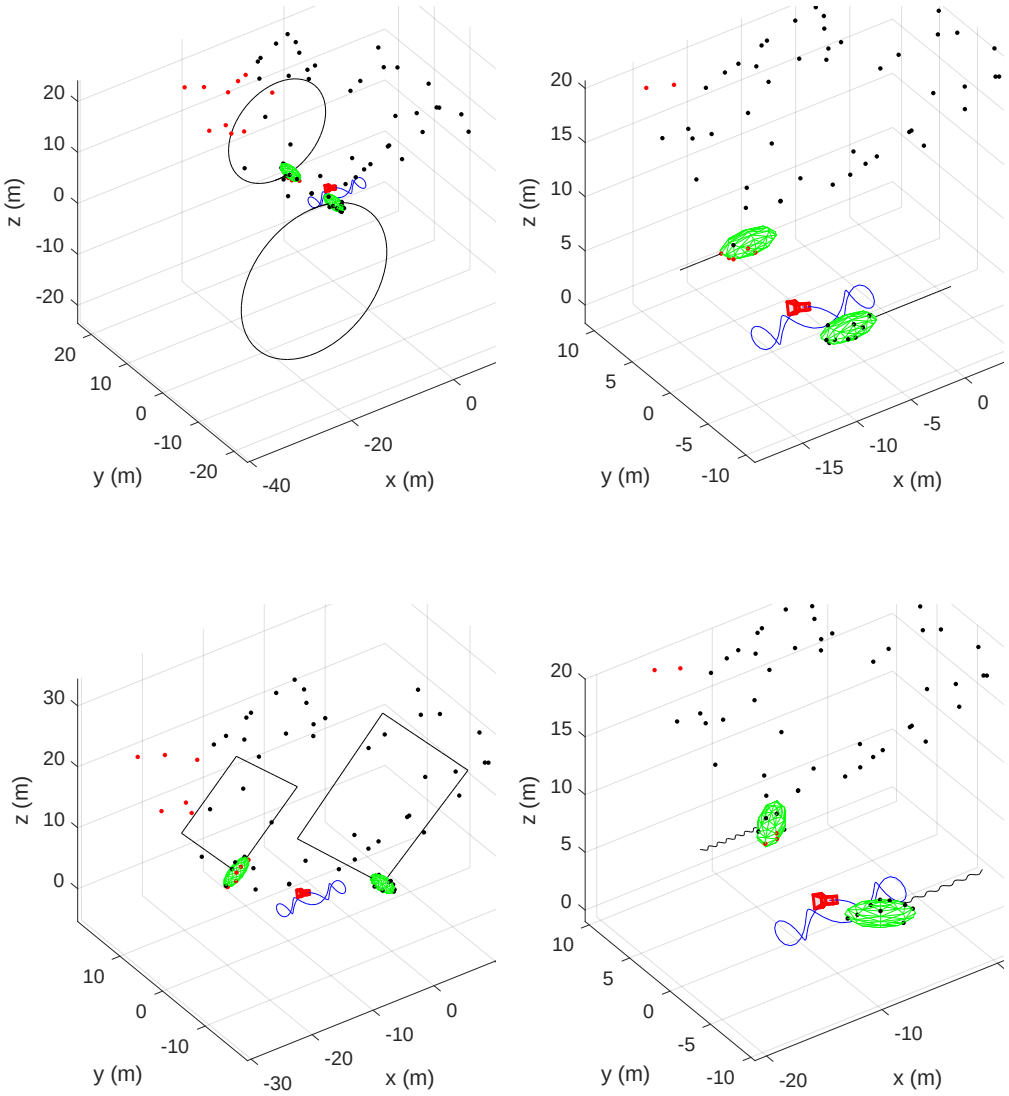



Figure 7.2: **Examples of simulated dynamic environments.** Ellipsoid rigid objects are simulated to have different trajectories, different camera trajectories can also be simulated, and static background points can be added. A limited field of view camera is shown, with red points being the ones observed at that time step.

## 7.2 C++ implementation

### 7.2.1 Visual Dynamic Object-aware SLAM

The last publicly available source code that we provide for the benefit of the community is "Visual Dynamic Object-aware SLAM" (VDO-SLAM) that can be found on [https://github.com/halajun/VDO_SLAM], a feature-based object-aware full SLAM system that exploits semantic image-based information to achieve robustness and consistency in various scenarios including challenging highly-dynamic environments. As opposed to Entity SLAM and DO-SLAM where the main focus was on the back-end, VDO-SLAM is a full RGB-D SLAM system as is explained in chapter 5.

## 7.3 Features

### 7.3.1 Graph-files

The concept of a *graph-file* was introduced with the increased interest in graph-based SLAM. A graph-file is mainly a text file that serves as an easy and memory efficient tool to store graph-based SLAM data. Like graph-based SLAM, a graph-file consists of vertices and edges. Each line of the text file represents an entry in the graph; a graph vertex or edge. A typical SLAM graph-file uses "tags" or "keywords" to identity the type of vertex or edge (e.g. VERTEX_SE2 for a robot/camera pose in $2D$, EDGE_SE2 for an odometry edge between 2 robot poses in $2D$ as defined in g2o graph files).

As an attempt to unify graph-file formats across different SLAM systems:

- we introduce a new graph-file that follows the same structure of a typical graph-file,
- we define our own tags/keywords for different vertices and edges types
- we provide examples of simple 3D classical SLAM problems that include robot poses, and point positions vertices, odometry and point-measurement edges
- we introduce new vertices and edges type, for a SLAM system that utilise structure and dynamic information, and
- we provide examples of graph-files of SLAM systems that utilise structure and dynamic information

- we provide MATLAB conversion functions to convert between our graph-files format and the most commonly used formats; GTSAM [32] and g2o [54] graph-files. Conversion functions are included to allow the conversion of any graph-file type to another.

### 7.3.2 New vertices and edges

To be able to incorporate meta-information in terms of structure, semantic and dynamic information into SLAM, as explained in chapters 4 and 5 , we define new vertices and edges types. We initially implement the new vertices, methods to update them, the new introduced factors, and their Jacobians in MATLAB and then translate them into GTSAM [32] and g2o [54]. Introducing the new factors in GTSAM [32] and g2o [54], along with conversion functions to convert between our graph-files and the above-mentioned SLAM systems graph-files serve as a basis to test, develop and extend the work achieved in this thesis.

### 7.3.3 Visualisation tools

We also include visualisation tools that allow to:

- visualise the simulated environments including structure information,
- plot output maps including structure planar surfaces in addition to non-planar points and camera poses and provide a visualisation of performance compared to ground-truth
- create animation clips that show the evolution of the robot and planar entities estimates
- visualise the simulated environments, including the camera and rigid object trajectories and visible points at each time step,
- create animation clips that show the evolution of the robot and object trajectories
- plot spatio-temporal output maps of dynamic SLAM, including tracks of points on rigid objects in motion, in addition to static background structure and camera/robot poses and provide a visualisation of performance compared to ground-truth
- output video streams of real data sequences (e.g. KITTI [227] and MVO [226])

Examples of structure and dynamic SLAM output visualisations are shown in Fig. 7.3.

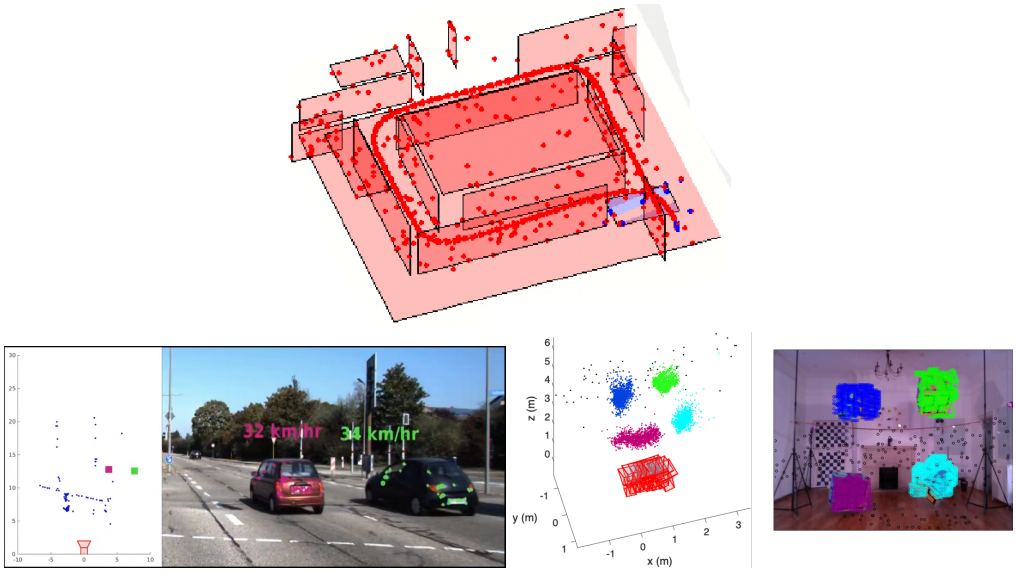



Figure 7.3: **Examples of structure and dynamic SLAM output visualisation.**

### 7.3.4 Error metrics

Finally, as a contribution of this thesis, we propose and implement new error metrics that are suited for dynamic SLAM systems. We provide error functions that read as input the output graph-file solution and the ground-truth graph file and returns an evaluation of the accuracy of the proposed algorithm.

In addition to the classical camera pose errors (Absolute Trajectory Error and Relative Pose Error), we also introduce two new error metrics for SLAM in dynamic environments. The first error metric is concerned with the dynamic structure, and the accuracy of object pose change estimation. We call this error *Motion Induced Pose Error* (MIPE) and is calculated as explained in section 6.4.2 in chapter 5. We also introduce a second error that is more relevant in driving scenarios; object speed error as explained in the same above-mentioned section.

Chapter 8

# Conclusion

## 8.1 Summary and contributions

In this thesis, we have tackled the problem of utilising *meta-information* into a SLAM framework to achieve a robust and consistent representation of the environment and challenge some of the most limiting assumptions in the literature. Previous approaches have been addressed and a number of new techniques have been presented.
The main achievements are:

- A SLAM framework that is able to integrate *different types of information* about the environment, and an optimisation method with novel observation functions to achieve robust and consistent map representation and robot poses.
- A *structural SLAM* framework that integrates *structural information* about the layout of points in the environment in the form of planarity information and introduces higher-level information associated with orthogonality and parallelism of planes to achieve structural consistency of the returned map.
- A novel way to integrate information about dynamic and static structures in the environment into a single estimation framework resulting in accurate robot pose and spatio-temporal(time-varying) maps estimation.
- Speed SLAM, a model-free object-aware feature-based SLAM system that exploits *semantic information* to achieve robust moving object tracking, accurate estimation of dynamic objects full $\mathrm{SE}(3)$ motion, and extract velocity information of moving objects in the scene, along with an accurate robot localisation and mapping of static environment structure.

- A robust method to enhance *scene 3D understanding* for moving object tracking exploiting semantic information, and the ability to deal with non-direct occlusions resulting from the failure of semantic object segmentation.

- An *open-source code* that is made available for the community to develop and extend including novel observation functions to integrate structural, semantic and dynamic information, and is made flexible to include any type of implicit information as long as there is an algorithm "sensor" that can provide this information and a function that can model it.

## 8.2 Future work

Integrating meta information into a visual SLAM framework has shown to improve the estimation accuracy and to open doors to a wider class of problems such as SLAM in texture-less environments or in cluttered and dynamic environments. While SLAM algorithms are concerned with the problem of continually building a map of some *unknown* environments and being able to localise the robot within this map at all time steps, we have a great amount of information about these unknown environments that should be exploited in a SLAM framework.
This thesis represents a building block towards a SLAM system that utilises meta information into visual graph-based SLAM to achieve robust, consistent and dynamic representations of the world.

- The SLAM system presented in this thesis is feature-based. Although this 'feature-based' property is at the core of the accuracy of the proposed system, the computational complexity associated with such systems becomes unbound in large-scale complex environments. While only concerned with feature-based SLAM, the system presented in this thesis is layout-aware (chapter 4) and object-aware (chapter 5). This allows for the natural extension of introducing higher-level entities (e.g. planes) or objects (e.g. cuboids, quadrics, etc.) to replace the points and reduce the computational and memory requirements of the system. This grouping of points into entities and entities into objects is an interesting question, and determining the right point in time to group points into a plane or an object, or split a plane or object back into its constituting points is worth exploring.

- The problems of SLAM and scene understanding are mutually beneficial. While presented as a 'one-direction' communication, where the structural, semantic and dynamic information are integrated into a SLAM framework to improve the estimation, a 'closed-loop' system should utilise the SLAM estimates to guide, refine and optimise the scene understanding. This feedback could be in the form of projected point/object locations in the future to guide the tracking, hallucinating object shapes to handle occlusions, etc.

While explored on a small scale here, handling indirect occlusions and recovering object masks when segmentation fails to ensure tracking, more research in this direction looks promising.

- With a part of this thesis trying to tackle dynamic environments, and with the advances in the field of event cameras, high frequency sensors and the use of direct methods, filtering approaches,that only consider the current robot state and the map, are starting to regain a lot of attention. They however suffer large drift in the long term, and are inconsistent when applied to the inherently non-linear SLAM problem. We believe a combination of filtering and optimisation based methods is the right way to tackle SLAM in dynamic environments. A graph-based SLAM system where each node is a filtering approach estimating the latest robot state and the map seems to be a promising way to tackle such environments. Dynamic SLAM will continue to be a key component in many robotics, augmented and virtual reality applications and thus pushing the accuracy and robustness of SLAM in dynamic environments and extending the results of this thesis to other scene topologies are streams of work worth exploring.

- Finally, while only concerned with structural, semantic and dynamic information, this thesis and the system developed represents a building block towards integrating any type of available information into a single framework. Geometric information in the form of distance information for geometric shapes of known dimensions, affordance information between objects (e.g supporting, inside, over, under, etc.) or dynamic information in the form of motion models for moving objects could be integrated. Although presented briefly in this thesis, the constant motion assumption was tested for vehicles on highways, and smooth trajectories (models that prohibit abrupt changes of motion) are integrated in the estimation using a temporal window fashion. With the advances of learning based techniques, important cues about the motion/intention of a vehicle (accelerating, moving with a constant motion or decelerating) can be provided based on attention models to street signs, traffic lights, etc., integrating such information into a dynamic SLAM framework is believed to achieved significant improvements.

Appendix A

# Structural SLAM

## A.1 Particularities of the non-linear least-squares Jacobian

Taking the derivative of $g(m^i, \eta^s, d^s) = d^s - m^{i\top}\eta^s$ with respect to $\eta^s \in S^2$ involves projecting the differential of $g(m^i, \eta^s, d^s)$ onto the tangent space of $S^2$ using the projector $(I - \eta^s\eta^{s\top})$ where $I$ is the identity matrix:

$$\nabla_{\eta^s}\, g(m^i, \eta^s, d^s) = (I - \eta^s\eta^{s\top})\frac{\partial g(m^i, \eta^s, d^s)}{\partial \eta^s}\,. \tag{A.1}$$

Observe that representing a plane by four parameters is an over-parameterisation, which causes the system matrix to be singular with zero eigen-values associated with the term $(I - \eta^s\eta^{s\top})$ and any attempt to invert it within the NLS solver will fail.

There are two ways to overcome this problem. The parameterisation of the plane can be characterised by $\eta^s$ now in $\mathbb{R}^3$ and $d^s \in \mathbb{R}$, in which case the optimisation should constrain $\eta^{s\top}\eta^s - 1 = 0$, to ensure the solution converges to a unit norm of the normal vector $\eta^s$. This can be modelled as a unary factor in the factor graph in Fig. 4.5. This approach not only introduces extra unnecessary factors in the graph for each detected plane but it is also very sensitive to the confidence associated to this prior.

A second and more elegant approach is to keep the $\eta^s \in S^2$ parameterisation and to solve a constrained optimisation problem. This can be done by calculating the perpendicular kernel [234] to the matrix $A^\top A$ and solve the equation $(K^\perp)^T A^\top A K^\perp \delta' = (K^\perp)^T A^\top b$ and then reconstruct the $\delta = K^\perp \delta'$. This is explained in the following.

We first start with the Gauss-Newton linear system described by the equation:

$$A^\top A\boldsymbol{\delta} = A^\top \boldsymbol{b}\,, \tag{A.2}$$

with optimal solution $\boldsymbol{\delta}^*$. We define $K$, a matrix whose columns span the null space of $A^T A$, and $K^\perp$ an orthogonal matrix to $K$ and whose columns span the domain of $A^\top A$ by construction. The size of $A$ is $M \times N$, where:
$N = \dim(X_k) \cdot n_X + \dim(m^i) \cdot n_m + \dim(\eta^s) \cdot n_s + \dim(d^s) \cdot n_s$,
with $n_X, n_m, n_s$ total number of robot poses, landmark points and detected planes respectively.
$M = \dim(o_k) \cdot m_k + \dim(z_k^i) \cdot m_i + \dim(p^s) \cdot m_s + \dim(a^t) \cdot m_t$,
with $m_k, m_i, m_s$ and $m_t$ total number of odometry measurements, landmark points measurements, plane observations and angle observations respectively.
Therefore $K \in \mathbb{R}^{N \times n_s}$ and $K^\perp \in \mathbb{R}^{N \times (N - n_s)}$.
Since the solution $\boldsymbol{\delta}^*$ lies in the domain space of $A^\top A$, we can write $\boldsymbol{\delta}^* = K^\perp \boldsymbol{\delta}^{*'}$, where $\boldsymbol{\delta}^{*'} \in \mathbb{R}^{N - n_s}$. Thus one has:

$$A^\top A \boldsymbol{\delta}^* = A^\top A K^\perp \boldsymbol{\delta}^{*'} = A^\top \boldsymbol{b} \,. \tag{A.3}$$

Pre-multiplying by $(K^\perp)^\top$ gives:

$$(K^\perp)^\top A^\top A K^\perp \boldsymbol{\delta}^{*'} = (K^\perp)^\top A^\top \boldsymbol{b} \tag{A.4}$$

We solve for $\boldsymbol{\delta}^{*'}$ and then reconstruct $\boldsymbol{\delta}^* = K^\perp \boldsymbol{\delta}^{*'}$ for the solution of the actual system.
It is also worth mentioning that due to the sparsity and block structure of the matrix $A^\top A$, the product $(K^\perp)^\top A^\top A K^\perp$ and $(K^\perp)^\top A^\top b$ can be computed efficiently by exploiting the block structure.

Appendix B

# Dynamic SLAM

## B.1 Dynamic object point $\mathrm{SE}(3)$ motion

This section is devoted to describe the object $\mathrm{SE}(3)$ pose change in a global reference frame in the two particular cases of pure object translation and pure object rotation in body-fixed frame, and the general case of an object translation and rotation in body-fixed frame.

### B.1.1 Object pure translation

We start by writing the object pose in a global reference frame at consecutive time steps as follows: ${}^0L_{k-1} = \begin{pmatrix} {}^0R_{L_{k-1}} & {}^0t_{L_{k-1}} \\ \mathbf{0} & 1 \end{pmatrix}$ and ${}^0L_k = \begin{pmatrix} {}^0R_{L_{k-1}} & {}^0t_{L_k} \\ \mathbf{0} & 1 \end{pmatrix}$. where ${}^0R_{L_{k-1}} \in \mathrm{SO}(3)$ and ${}^0t_{L_{k-1}} \in \mathbb{R}^3$ the rotation and translation components of ${}^0L_{k-1}$, and ${}^0R_{L_{k-1}}$, ${}^0t_{L_k}$ their corresponding at time $k$. Note that if the object is undergoing a pure translation, the rotation component of the object pose remains the same. We first write the expression for ${}^{L_{k-1}}_{k-1}H_k$ based on its definition in (5.1).

$$ {}^{L_{k-1}}_{k-1}H_k = \begin{pmatrix} I_3 & {}^0R_{L_{k-1}}({}^0t_{L_k} - {}^0t_{L_{k-1}}) \\ \mathbf{0} & 1 \end{pmatrix} \tag{B.1} $$

Substituting for ${}^{L_{k-1}}_{k-1}H_k$ by its definition in (5.1), the object pose change in global reference frame can be expressed as:

$$ {}^0_{k-1}H_k = {}^0L_{k-1}\ {}^{L_{k-1}}_{k-1}H_k\ {}^0L^{-1}_{k-1} = {}^0L_k\ {}^0L^{-1}_{k-1} \tag{B.2} $$

which can be written as:

$$ {}^{0}_{k-1}H_k = \begin{pmatrix} I_3 & {}^{0}t_{L_k} - {}^{0}t_{L_{k-1}} \\ \mathbf{0} & 1 \end{pmatrix} \tag{B.3} $$

Note the difference between ${}^{L_{k-1}}_{k-1}H_k$ and ${}^{0}_{k-1}H_k$ in case of a pure translational motion in the object body-fixed frame. ${}^{0}_{k-1}H_k$ is quite intuitive in the case of an object pure translation in body-fixed frame and shows that our approach is able to calculate the right object $\mathrm{SE}(3)$ motion in case of an object pure translation.

#### B.1.1.1 Illustrative example

An illustrative simulated example is shown where a single ellipsoid is undergoing a pure translational motion in 3D, followed by a robot that observes the object and estimates its motion based on measurements of points on the ellipsoid. The object is simulated to have an initial pose of [10; 0; 0; 0; 0; 0.2] and a pose change of [1.5; 0.5; 1; 0; 0; 0] in body-fixed frame, written as [translation; scaled axis-angle representation of rotation].

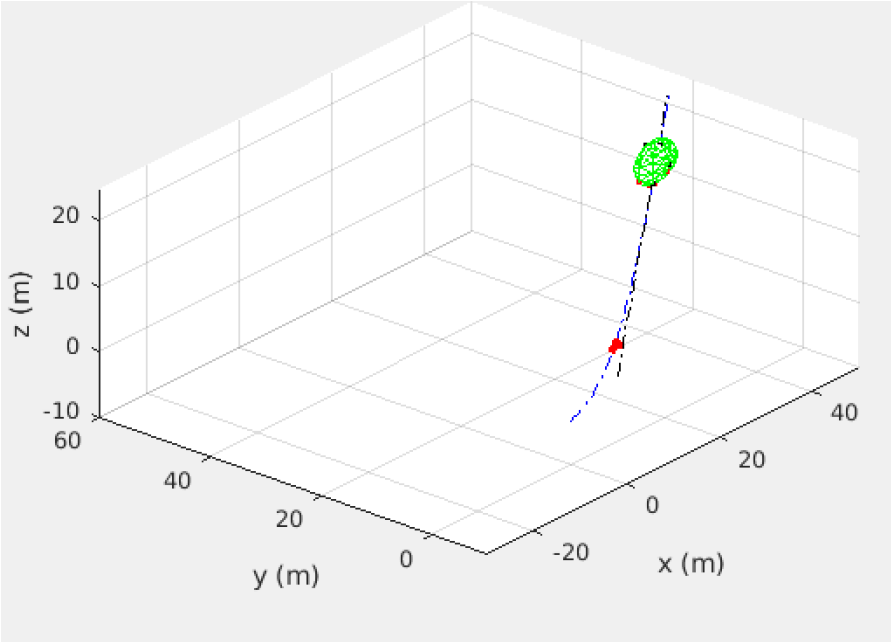



Figure B.1: **Simulated example of a rigid object undergoing a pure translation.**

Starting from an identity initial motion estimate, and assuming known robot poses and a perfect 3D point measurement sensor, our algorithm is able to estimate the exact object pose change in a global frame, here [1.3708; 0.7880; 1; 0; 0; 0], with zero translational and rotational error only from measurements of points that reside on the object. Note the effect of the object's

pose initial rotational component on the difference between the object pose change in body-fixed frame and global frame.

### B.1.2 Object pure rotation

We start by writing the object pose in a global reference frame at consecutive time steps as follows: ${}^{0}L_{k-1} = \begin{pmatrix} {}^{0}R_{L_{k-1}} & {}^{0}t_{L_{k-1}} \\ \mathbf{0} & 1 \end{pmatrix}$ and ${}^{0}L_{k} = \begin{pmatrix} {}^{0}R_{L_{k}} & {}^{0}t_{L_{k-1}} \\ \mathbf{0} & 1 \end{pmatrix}$. where ${}^{0}R_{L_{k-1}} \in \mathrm{SO}(3)$ and ${}^{0}t_{L_{k-1}} \in \mathbb{R}^3$ the rotation and translation components of ${}^{0}L_{k-1}$, and ${}^{0}R_{L_{k}}$, ${}^{0}t_{L_{k-1}}$ their corresponding at time $k$. Note that if the object is undergoing a pure rotation, the translation component of the object pose remains the same. We first write the expression for ${}^{L_{k-1}}_{k-1}H_{k}$ based on its definition in (5.1).

$$ {}^{L_{k-1}}_{k-1}H_{k} = \begin{pmatrix} {}^{0}R^{\top}_{L_{k-1}} \, {}^{0}R_{L_{k}} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \tag{B.4} $$

Substituting for ${}^{L_{k-1}}_{k-1}H_{k}$ by its definition in (5.1), the object pose change in global reference frame can be expressed as:

$$ {}^{0}_{k-1}H_{k} = {}^{0}L_{k-1} \, {}^{L_{k-1}}_{k-1}H_{k} \, {}^{0}L^{-1}_{k-1} = {}^{0}L_{k} \, {}^{0}L^{-1}_{k-1} \tag{B.5} $$

which can be written as:

$$ {}^{0}_{k-1}H_{k} = \begin{pmatrix} {}^{0}R_{L_{k}} \, {}^{0}R^{\top}_{L_{k-1}} & {}^{0}t_{L_{k-1}} - {}^{0}R_{L_{k}} \, {}^{0}R^{\top}_{L_{k-1}} \, {}^{0}t_{L_{k-1}} \\ \mathbf{0} & 1 \end{pmatrix} \tag{B.6} $$

${}^{0}_{k-1}H_{k}$ is not as intuitive in the case of an object pure rotation in body-fixed frame. A rotation in the object body-fixed frame will always result in a rotation and translation components in the object pose change in global frame, due to the frame change of the object pose transformation.

#### B.1.2.1 Illustrative example

An illustrative simulated example is shown where a single ellipsoid is undergoing a pure rotational motion in 3D, observed by a robot that estimates its motion based on measurements of points on the ellipsoid. The object is simulated to have an initial pose of [10; 0; 0; 0; 0; 0.2] and a pose change of [0; 0; 0; 0.0045; 0.0201; 0.0499] in body-fixed frame, written as [translation; scaled axis-angle representation of rotation].

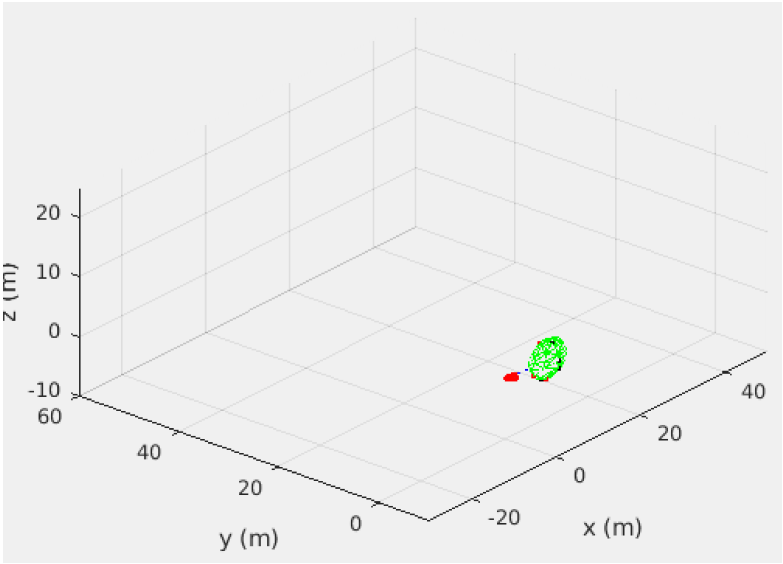



Figure B.2: **Simulated example of a rigid object undergoing a pure rotation.**

Starting from an identity initial motion estimate, and assuming known robot poses and a perfect 3D point measurement sensor, our algorithm is able to estimate the exact object pose change in a global frame, here [0.0146; -0.4993; 0.2059; 0.0004; 0.0206; 0.0499], with zero translational and rotational error only from measurements of points that reside on the object. Again, the simulated example shows that a rotation in the object body-fixed frame will always result in a rotation and translation components in the object pose change in global frame.

### B.1.3 Object translation and rotation

We start by writing the object pose in a global reference frame at consecutive time steps as follows:

$${}^0L_{k-1} = \begin{pmatrix} {}^0R_{L_{k-1}} & {}^0t_{L_{k-1}} \\ \mathbf{0} & 1 \end{pmatrix} \text{ and } {}^0L_k = \begin{pmatrix} {}^0R_{L_k} & {}^0t_{L_k} \\ \mathbf{0} & 1 \end{pmatrix}. \text{ where } {}^0R_{L_{k-1}} \in \mathrm{SO}(3) \text{ and } {}^0t_{L_{k-1}} \in \mathbb{R}^3$$ the rotation and translation components of ${}^0L_{k-1}$, and ${}^0R_{L_k}$, ${}^0t_{L_k}$ their corresponding at time $k$. We first write the expression for ${}^{L_{k-1}}_{k-1}H_k$ based on its definition in (5.1).

$$ {}^{L_{k-1}}_{k-1}H_k = \begin{pmatrix} {}^0R^\top_{L_{k-1}} \, {}^0R_{L_k} & {}^0R^\top_{L_{k-1}}({}^0t_{L_k} - {}^0t_{L_{k-1}}) \\ \mathbf{0} & 1 \end{pmatrix} \tag{B.7}$$

Substituting for ${}^{L_{k-1}}_{k-1}H_k$ by its definition in (5.1), the object pose change in global reference

frame can be expressed as:

$$ {}^{0}_{k-1}H_k = {}^{0}L_{k-1}\ {}^{L_{k-1}}_{k-1}H_k\ {}^{0}L^{-1}_{k-1} = {}^{0}L_k\ {}^{0}L^{-1}_{k-1} \tag{B.8} $$

which can be written as:

$$ {}^{0}_{k-1}H_k = \begin{pmatrix} {}^{0}R_{L_k}\ {}^{0}R^{\top}_{L_{k-1}} & {}^{0}t_{L_k} - {}^{0}R_{L_k}\ {}^{0}R^{\top}_{L_{k-1}}\ {}^{0}t_{L_{k-1}} \\ \mathbf{0} & 1 \end{pmatrix} \tag{B.9} $$

As discussed in the case of a pure rotation, a rotation in the object body-fixed frame will always result in a rotation and translation components in the object pose change in global frame, due to the frame change of the object pose transformation.

#### B.1.3.1 Illustrative example

An illustrative simulated example is shown where a single ellipsoid is undergoing a full $\mathrm{SE}(3)$ motion in 3D, followed by a robot that observes the object and estimates its motion based on measurements of points on the ellipsoid. The object is simulated to have an initial pose of [10; 0; 0; 0; 0; 0.2] and a pose change of [1.5; 0.5; 1; 0.0045; 0.0201; 0.0499] in body-fixed frame, written as [translation; scaled axis-angle representation of rotation].

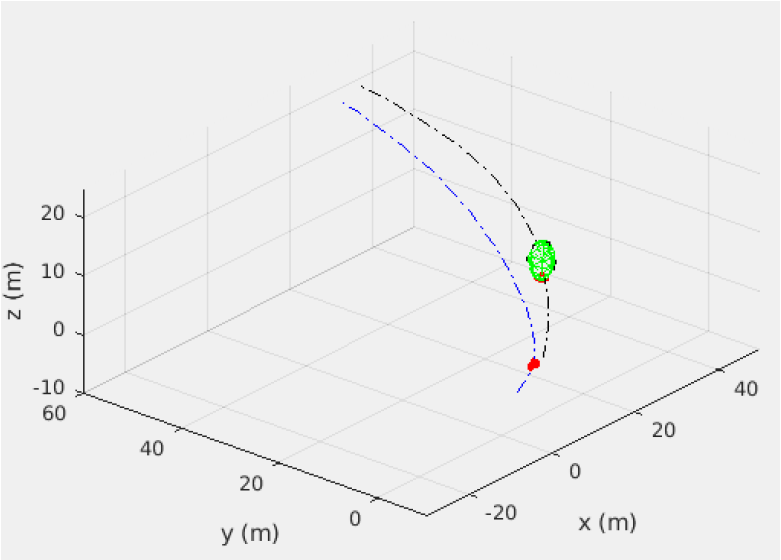



Figure B.3: **Simulated example of a rigid object undergoing a full** $\mathrm{SE}(3)$ **motion.**

Starting from an identity initial motion estimate, and assuming known robot poses and a perfect 3D point measurement sensor, our algorithm is able to estimate the exact object pose

change in a global frame, here [1.3854; 0.2888; 1.2059; 0.0004; 0.0206; 0.0499], with zero translational and rotational error only from measurements of points that reside on the object.

### B.1.4 Discussion

For all of the above cases, the resulting pose of the object at time $k$ is equivalently obtained by applying the object motion in global reference frame or in body-fixed frame accordingly for all possible object motions:

$$ {}^{0}L_k = {}^{0}_{k-1}H_k \, {}^{0}L_{k-1} = {}^{0}L_{k-1} \, {}^{L_{k-1}}_{k-1}H_k \tag{B.10} $$

which proves that our approach makes no assumptions about the motion of the object, and shows the capability of our approach to calculate the full object $\mathrm{SE}(3)$ pose change irrespective of the type of motion the object has undergone.

## B.2 Linear velocity extraction from object motion proof

To see that (5.7) is the same quantity in $3D$ as the translation vector from the origin of the object pose at time $\{k-1\}$ to the origin of the object pose at time $\{k\}$ as seen in the global reference frame $\{0\}$, we start by writing the object pose change in global reference frame and substitute for ${}^{L_{k-1}}_{k-1}H_k$ by its definition in (5.1)

$$ {}^{0}_{k-1}H_k = {}^{0}L_{k-1} \, {}^{L_{k-1}}_{k-1}H_k \, {}^{0}L^{-1}_{k-1} = {}^{0}L_k \, {}^{0}L^{-1}_{k-1} \tag{B.11} $$

Assuming ${}^{0}R_{L_{k-1}} \in \mathrm{SO}(3)$ and ${}^{0}t_{L_{k-1}} \in \mathbb{R}^3$ the rotation and translation components of ${}^{0}L_{k-1}$, and ${}^{0}R_{L_k}$, ${}^{0}t_{L_k}$ their corresponding at time $k$, the translation and rotation parts of ${}^{0}_{k-1}H_k$ can be expressed as ${}^{0}t_{L_k} - {}^{0}R_{L_k} \, {}^{0}R^{\top}_{L_{k-1}} \, {}^{0}t_{L_{k-1}}$ and ${}^{0}R_{L_k} \, {}^{0}R^{\top}_{L_{k-1}}$. Substituting these two quantities into (5.7), we get

$$ v = {}^{0}t_{L_k} - {}^{0}R_{L_k} \, {}^{0}R^{\top}_{L_{k-1}} \, {}^{0}t_{L_{k-1}} - (I_3 - {}^{0}R_{L_k} \, {}^{0}R^{\top}_{L_{k-1}}) \, {}^{0}t_{L_{k-1}} \tag{B.12} $$

which reduces to $v = {}^{0}t_{L_k} - {}^{0}t_{L_{k-1}}$ which is the translation vector from the origin of the object pose at time $\{k-1\}$ to the origin of the object pose at time $\{k\}$ as seen in the global reference frame $\{0\}$.

Appendix C

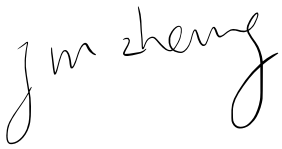
# Visual Dynamic Object-aware SLAM

## C.1 Authors' contribution disclosure

*The piece of research work presented in chapter "Visual Dynamic Object-aware SLAM" is a collaboration between the author and Jun Zhang, where the author's main contributions were the back-end formulation and its implementation, the mathematical modelling of the dynamic object motion in terms of the points that constitute the object, the velocity extraction from the object pose change estimation, the engagement with authors of state-of-the-art dynamic SLAM systems for evaluation and comparison to the work in question, and parts of the technical writing.*

Mina Henein
June 21$^{st}$, 2020

Jun Zhang
June 21$^{st}$,2020



# Bibliography


[1] Anoop Cherian and Stephen Gould. “Second-order Temporal Pooling for Action Recognition”. In: *arXiv preprint arXiv:1704.06925* (2017).

[2] Bruce Xiaohan Nie, Caiming Xiong, and Song-Chun Zhu. “Joint action recognition and pose estimation from video”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1293–1301.

[3] Haoyu Ren, Mostafa El-khamy, and Jungwon Lee. “Deep Robust Single Image Depth Estimation Neural Network Using Scene Understanding”. In: *arXiv preprint arXiv:1906.03279* (2019).

[4] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. “PlaneRCNN: 3d plane detection and reconstruction from a single image”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4450–4459.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask R-CNN”. In: *IEEE International Conference on Computer Vision (ICCV), 2017*. IEEE. 2017, pp. 2980–2988.

[6] Johan Vertens, Abhinav Valada, and Wolfram Burgard. “SMSnet: Semantic motion segmentation using deep convolutional neural networks”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 582–589.

[7] Mehdi Hosseinzadeh, Yasir Latif, Trung Pham, Niko Suenderhauf, and Ian Reid. “Structure aware SLAM using quadrics and planes”. In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 410–426.

[8] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. “Slam++: Simultaneous localisation and mapping at the level of

objects". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013*. IEEE. 2013, pp. 1352–1359.

[9] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. "DS-SLAM: A semantic visual SLAM towards dynamic environments". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1168–1174.

[10] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. "Dense planar slam". In: *IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2014*. IEEE. 2014, pp. 157–164.

[11] Michael Kaess. "Simultaneous localization and mapping with infinite planes". In: *IEEE International Conference on Robotics and Automation (ICRA), 2015*. IEEE. 2015, pp. 4605–4611.

[12] Mina Henein, Montiel Abello, Viorela Ila, and Robert Mahony. "Exploring The Effect of Meta-Structural Information on the Global Consistency of SLAM". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems 2017*. The Australian National University. 2017.

[13] Shichao Yang and Sebastian Scherer. "CubeSLAM: Monocular 3-D Object SLAM". In: *IEEE Transactions on Robotics* (2019).

[14] Richard Szeliski and Philip HS Torr. "Geometrically constrained structure from motion: Points on planes". In: *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*. Springer. 1998, pp. 171–186.

[15] Aisha Walcott-Bryant, Michael Kaess, Hordur Johannsson, and John J Leonard. "Dynamic pose graph SLAM: Long-term mapping in low dynamic environments". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012*. IEEE. 2012, pp. 1871–1878.

[16] Moritz Kampelmühler, Michael G Müller, and Christoph Feichtenhofer. "Camera-based vehicle velocity estimation from monocular video". In: (2018).

[17] Romuald Aufrère, Jay Gowdy, Christoph Mertz, Chuck Thorpe, Chieh-Chih Wang, and Teruko Yata. "Perception for collision avoidance and autonomous driving". In: *Mechatronics* 13.10 (2003), pp. 1149–1161.

[18] Ronald K Jurgen. *Adaptive cruise control*. Tech. rep. SAE Technical Paper, 2006.

[19] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. *Detectron*. https://github.com/facebookresearch/detectron. 2018.

[20] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. "QuadricSLAM: Dual Quadrics as SLAM Landmarks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 313–314.

[21] John J Leonard and Hugh F Durrant-Whyte. "Mobile robot localization by tracking geometric beacons". In: *IEEE transactions on robotics and automation* 7.3 (1991), pp. 376–382.

[22] Randall Smith, Matthew Self, and Peter Cheeseman. "Estimating uncertain spatial relationships in robotics". In: *Autonomous robot vehicles*. Springer, 1990, pp. 167–193.

[23] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[24] Hauke Strasdat. "Local accuracy and global consistency for efficient visual SLAM". PhD thesis. Department of Computing, Imperial College London, 2012.

[25] Jonathan Richard Shewchuk et al. *An introduction to the conjugate gradient method without the agonizing pain*. 1994.

[26] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. "Methods for non-linear least squares problems". In: (2004).

[27] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. "FastSLAM: A factored solution to the simultaneous localization and mapping problem". In: *Aaai/iaai*. 2002, pp. 593–598.

[28] Kevin P Murphy. "Bayesian map learning in dynamic environments". In: *Advances in Neural Information Processing Systems*. 2000, pp. 1015–1021.

[29] Mark A Paskin. "Thin junction tree filters for simultaneous localization and mapping". In: *in International Joint Conference on Artificial Intelligence*. Citeseer. 2003.

[30] Sebastian Thrun. "Probabilistic robotics". In: *Communications of the ACM* 45.3 (2002), pp. 52–57.

[31] John Folkesson and Henrik Christensen. "Graphical SLAM-a self-correcting map". In: *IEEE International Conference on Robotics and Automation, (ICRA) 2004.* Vol. 1. IEEE. 2004, pp. 383–390.

[32] Frank Dellaert and Michael Kaess. "Square Root SAM: Simultaneous localization and mapping via square root information smoothing". In: *The International Journal of Robotics Research* 25.12 (2006), pp. 1181–1203.

[33] Jose A Castellanos, JMM Montiel, José Neira, and Juan D Tardós. "The SPmap: A probabilistic framework for simultaneous localization and map building". In: *IEEE Transactions on Robotics and Automation* 15.5 (1999), pp. 948–952.

[34] M.W.M. Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. "A solution to the simultaneous localization and map building (SLAM) problem". In: *IEEE Transactions on Robotics and Automation* 17.3 (2001), pp. 229–241.

[35] John J Leonard, Hugh F Durrant-Whyte, and Ingemar J Cox. "Dynamic map building for an autonomous mobile robot". In: *The International Journal of Robotics Research* 11.4 (1992), pp. 286–298.

[36] Peter S Maybeck. *Stochastic models, estimation, and control*. Academic press, 1982.

[37] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. "Factor graphs and the sum-product algorithm". In: *IEEE Transactions on information theory* 47.2 (2001), pp. 498–519.

[38] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Vol. 16. Siam, 1996.

[39] Jose-Luis Blanco. "A tutorial on se (3) transformation parameterizations and on-manifold optimization". In: *University of Malaga, Tech. Rep* 3 (2010).

[40] Ethan Eade. "Lie groups for 2d and 3d transformations". In: *URL http://ethaneade.com/lie. pdf, revised Dec* (2013).

[41] P Cheeseman, R Smith, and M Self. "A stochastic map for uncertain spatial relationships". In: *4th International Symposium on Robotic Research*. 1987, pp. 467–474.

[42] Nicholas Ayache and Olivier D Faugeras. "Maintaining representations of the environment of a mobile robot". In: *IEEE transactions on Robotics and Automation* 5.6 (1989), pp. 804–819.

[43] Daniel Hahnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements". In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS) 2003.* Vol. 1. IEEE. 2003, pp. 206–211.

[44] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improved techniques for grid mapping with rao-blackwellized particle filters". In: *IEEE Transactions on Robotics.* 23.1 (2007), pp. 34–46.

[45] Ryan M Eustice, Hanumant Singh, and John J Leonard. "Exactly sparse delayed-state filters". In: *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA) 2005*. IEEE. 2005, pp. 2417–2424.

[46] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. "Simultaneous localization and mapping with sparse extended information filters". In: *The International Journal of Robotics Research* 23.7-8 (2004), pp. 693–716.

[47] Chanki Kim, Rathinasamy Sakthivel, and Wan Kyun Chung. "Unscented FastSLAM: a robust and efficient solution to the SLAM problem". In: *IEEE Transactions on robotics* 24.4 (2008), pp. 808–820.

[48] Steven Holmes, Georg Klein, and David W Murray. "A square root unscented Kalman filter for visual monoSLAM". In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 3710–3716.

[49] Simon J Julier and Jeffrey K Uhlmann. "A counter example to the theory of simultaneous localization and map building". In: *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA) 2001*. Vol. 4. IEEE. 2001, pp. 4238–4243.

[50] Feng Lu and Evangelos Milios. "Globally consistent range scan alignment for environment mapping". In: *Autonomous robots* 4.4 (1997), pp. 333–349.

[51] Edwin Olson, John Leonard, and Seth Teller. "Fast iterative alignment of pose graphs with poor initial estimates". In: *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA) 2006.* IEEE. 2006, pp. 2262–2269.

[52] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. "Bundle adjustment—a modern synthesis". In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372.

[53] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. "iSAM: Incremental smoothing and mapping". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378.

[54] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. "g 2 o: A general framework for graph optimization". In: *IEEE International Conference on Robotics and Automation (ICRA), 2011*. IEEE. 2011, pp. 3607–3613.

[55] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. "A tutorial on graph-based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.

[56] Giorgio Grisetti, Cyrill Stachniss, Slawomir Grzonka, and Wolfram Burgard. “A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent.” In: *Robotics: Science and Systems*. 2007, pp. 27–30.

[57] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. “iSAM2: Incremental smoothing and mapping using the Bayes tree”. In: *The International Journal of Robotics Research* (2011), p. 0278364911430419.

[58] Georg Klein and David Murray. “Parallel tracking and mapping for small AR workspaces”. In: *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE. 2007, pp. 225–234.

[59] Hauke Strasdat, JMM Montiel, and Andrew J Davison. “Real-time monocular SLAM: Why filter?” In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 2657–2664.

[60] Edward Rosten and Tom Drummond. “Machine learning for high-speed corner detection”. In: *European conference on computer vision*. Springer. 2006, pp. 430–443.

[61] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.

[62] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[63] David G Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. IEEE. 1999, pp. 1150–1157.

[64] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. “RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments”. In: *The International Journal of Robotics Research* 31.5 (2012), pp. 647–663.

[65] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features”. In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[66] Andreas Geiger, Julius Ziegler, and Christoph Stiller. “Stereoscan: Dense 3d reconstruction in real-time”. In: *2011 IEEE intelligent vehicles symposium (IV)*. Ieee. 2011, pp. 963–968.

[67] Paloma de la Puente and Diego Rodríguez-Losada. “Feature based graph-SLAM in structured environments”. In: *Autonomous Robots* 37.3 (2014), pp. 243–260.

[68] Ming Hsiao, Eric Westman, Guofeng Zhang, and Michael Kaess. "Keyframe-based dense planar SLAM". In: *IEEE International Conference on Robotics and Automation (ICRA), 2017*. IEEE. 2017, pp. 5110–5117.

[69] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM". In: *European conference on computer vision*. Springer. 2014, pp. 834–849.

[70] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. "DTAM: Dense tracking and mapping in real-time". In: *2011 international conference on computer vision*. IEEE. 2011, pp. 2320–2327.

[71] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry". In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625.

[72] Matthew Brown and Sabine Süsstrunk. "Multi-spectral SIFT for scene category recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2011, pp. 177–184.

[73] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2. IEEE. 2006, pp. 2169–2178.

[74] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[76] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[77] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. "SSD: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[78] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. "Learning deep features for scene recognition using places database". In: *Advances in neural information processing systems*. 2014, pp. 487–495.

[79] Nils Plath, Marc Toussaint, and Shinichi Nakajima. “Multi-class image segmentation using conditional random fields and global classification”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. 2009, pp. 817–824.

[80] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE computer society conference on computer vision and pattern recognition*. Vol. 1. IEEE. 2005, pp. 886–893.

[81] Lubomir Bourdev, Subhransu Maji, Thomas Brox, and Jitendra Malik. “Detecting people using mutually consistent poselet activations”. In: *European conference on computer vision*. Springer. 2010, pp. 168–181.

[82] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. “Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context”. In: *International journal of computer vision* 81.1 (2009), pp. 2–23.

[83] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. “Class segmentation and object localization with superpixel neighborhoods”. In: *2009 IEEE 12th international conference on computer vision*. IEEE. 2009, pp. 670–677.

[84] Xiang-Yang Wang, Ting Wang, and Juan Bu. “Color image segmentation using pixel wise support vector machine classification”. In: *Pattern Recognition* 44.4 (2011), pp. 777–787.

[85] Tin Kam Ho. “Random decision forests”. In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[86] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. “Semantic texton forests for image categorization and segmentation”. In: *2008 IEEE conference on computer vision and pattern recognition*. IEEE. 2008, pp. 1–8.

[87] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. “Semantic segmentation with second-order pooling”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 430–443.

[88] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. “Multiscale conditional random fields for image labeling”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 2. IEEE. 2004, pp. II–II.

[89] Philipp Krähenbühl and Vladlen Koltun. “Efficient inference in fully connected crfs with gaussian edge potentials”. In: *Advances in neural information processing systems*. 2011, pp. 109–117.

[90] L'ubor Ladickỳ, Chris Russell, Pushmeet Kohli, and Philip HS Torr. "Associative hierarchical crfs for object class image segmentation". In: *IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 739–746.

[91] Joao Carreira and Cristian Sminchisescu. "CPMC: Automatic object segmentation using constrained parametric min-cuts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.7 (2011), pp. 1312–1328.

[92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[93] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[94] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[95] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[96] Huikai Wu, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yizhou Yu. "FastFCN: Rethinking dilated convolution in the backbone for semantic segmentation". In: *arXiv preprint arXiv:1903.11816* (2019).

[97] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[98] Jifeng Dai, Kaiming He, and Jian Sun. "Instance-aware semantic segmentation via multi-task network cascades". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3150–3158.

[99] Fisher Yu and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[100] Weiqiang Li, Jiatong Mu, and Guizhong Liu. "Multiple Object Tracking with Motion and Appearance Cues". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[101] René Vidal and Richard Hartley. "Motion segmentation with missing data using power factorization and gpca". In: *null*. IEEE. 2004, pp. 310–316.

[102] Shankar R Rao, Allen Y Yang, S Shankar Sastry, and Yi Ma. "Robust algebraic segmentation of mixed rigid-body and planar motions from two views". In: *International journal of computer vision* 88.3 (2010), pp. 425–446.

[103] Ehsan Elhamifar and René Vidal. "Sparse subspace clustering". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 2790–2797.

[104] Taotao Lai, Hanzi Wang, Yan Yan, Tat-Jun Chin, and Wan-Lei Zhao. "Motion segmentation via a sparsity constraint". In: *IEEE Transactions on Intelligent Transportation Systems* 18.4 (2017), pp. 973–983.

[105] Xun Xu, Loong Fah Cheong, and Zhuwen Li. "Motion segmentation by exploiting complementary geometric models". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2859–2867.

[106] Cosimo Rubino, Alessio Del Bue, and Tat-Jun Chin. "Practical Motion Segmentation for Urban Street View Scenes". In: *Robotics and Automation (ICRA), 2018 IEEE International Conference on*. IEEE. 2018.

[107] Jun Zhang and Viorela Ila. "Multi-frame Motion Segmentation for Dynamic Scene Modeling". In: (2018).

[108] Zia Khan, Tucker Balch, and Frank Dellaert. "MCMC-based particle filtering for tracking a variable number of interacting targets". In: *IEEE transactions on pattern analysis and machine intelligence* 27.11 (2005), pp. 1805–1819.

[109] Xiaolong Zhou, Hui Yu, Honghai Liu, and Youfu Li. "Tracking multiple video targets with an improved GM-PHD tracker". In: *Sensors* 15.12 (2015), pp. 30240–30260.

[110] Anton Milan, S Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. "Online multi-target tracking using recurrent neural networks". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[111] Chanho Kim, Fuxin Li, and James M Rehg. "Multi-object tracking with neural gating using bilinear lstm". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 200–215.

[112] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems". In: (1960), pp. 35–45.

[113] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. "Tracking the untrackable: Learning to track multiple cues with long-term dependencies". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 300–311.

[114] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. "Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4836–4845.

[115] Bruce D Lucas, Takeo Kanade, et al. "An iterative image registration technique with an application to stereo vision". In: (1981).

[116] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "Flownet: Learning optical flow with convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2758–2766.

[117] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. "Flownet 2.0: Evolution of optical flow estimation with deep networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2462–2470.

[118] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8934–8943.

[119] Wongun Choi and Silvio Savarese. "Multiple target tracking in world coordinate with single, minimally calibrated camera". In: *European Conference on Computer Vision*. Springer. 2010, pp. 553–567.

[120] Nam Le, Alexander Heili, and Jean-Marc Odobez. "Long-term time-sensitive costs for crf-based tracking by detection". In: *European Conference on Computer Vision*. Springer. 2016, pp. 43–51.

[121] Zheng Wu, Ashwin Thangali, Stan Sclaroff, and Margrit Betke. "Coupling detection and data association for multiple object tracking". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1948–1955.

[122] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 261–268.

[123] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple online and realtime tracking with a deep association metric". In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[124] Liming Zhao, Xi Li, Yueting Zhuang, and Jingdong Wang. "Deeply-learned part-aligned representations for person re-identification". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3219–3228.

[125] Long Chen, Haizhou Ai, Zijie Zhuang, and Chong Shang. "Real-time multiple people tracking with deeply learned candidate selection and person re-identification". In: *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2018, pp. 1–6.

[126] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. "Learning by tracking: Siamese CNN for robust target association". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016, pp. 33–40.

[127] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. "Object detection with discriminatively trained part-based models". In: *IEEE transactions on pattern analysis and machine intelligence* 32.9 (2009), pp. 1627–1645.

[128] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[129] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[130] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[131] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[132] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. "Single-shot refinement neural network for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4203–4212.

[133] Zhaowei Cai and Nuno Vasconcelos. "Cascade r-cnn: Delving into high quality object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6154–6162.

[134] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. "Globally-optimal greedy algorithms for tracking a variable number of objects". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE. 2011, pp. 1201–1208.

[135] Siyu Tang, Bjoern Andres, Miykhaylo Andriluka, and Bernt Schiele. "Subgraph decomposition for multi-target tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5033–5041.

[136] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. "Multi-person tracking by multicut and deep matching". In: *European Conference on Computer Vision*. Springer. 2016, pp. 100–111.

[137] James Munkres. "Algorithms for the assignment and transportation problems". In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.

[138] Yihong Xu, Yutong Ban, Xavier Alameda-Pineda, and Radu Horaud. "DeepMOT: A Differentiable Framework for Training Multiple Object Trackers". In: *arXiv preprint arXiv:1906.06618* (2019).

[139] Marco Zucchelli, José Santos-Victor, and Henrik I Christensen. "Constrained structure and motion estimation from optical flow". In: *16th International Conference on Pattern Recognition, Proceedings 2002.* Vol. 1. IEEE. 2002, pp. 339–342.

[140] Tae-kyeong Lee, Seungwook Lim, Seongsoo Lee, Shounan An, and Se-young Oh. "Indoor mapping using planes extracted from noisy RGB-D sensors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),2012*. IEEE. 2012, pp. 1727–1733.

[141] Jan Weingarten and Roland Siegwart. "3D SLAM using planar segments". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),2006*. IEEE. 2006, pp. 3062–3067.

[142] Fabien Servant, Eric Marchand, Pascal Houlier, and Isabelle Marchal. "Visual planes-based simultaneous localization and model refinement for augmented reality". In: *19th International Conference on Pattern Recognition (ICPR), 2008*. IEEE. 2008, pp. 1–4.

[143] Andrew P Gee, Denis Chekhlov, Andrew Calway, and Walterio Mayol-Cuevas. "Discovering higher level structure in visual SLAM". In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 980–990.

[144] José Martínez-Carranza and Andrew Calway. "Unifying Planar and Point Mapping in Monocular SLAM." In: *British Machine Vision Conference (BMVC), 2010*. Citeseer. 2010, pp. 1–11.

[145] Rudolph Triebel and Wolfram Burgard. "Improving simultaneous mapping and localization in 3d using global constraints". In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 20. 3. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2005, p. 1330.

[146] Alexander JB Trevor, John G Rogers, and Henrik I Christensen. "Planar surface SLAM with 3D and 2D sensors". In: *IEEE International Conference on Robotics and Automation (ICRA), 2012*. IEEE. 2012, pp. 3041–3048.

[147] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. "Point-plane SLAM for hand-held 3D sensors". In: *IEEE International Conference on Robotics and Automation (ICRA), 2013*. IEEE. 2013, pp. 5182–5189.

[148] Ming Hsiao, Eric Westman, and Michael Kaess. "Dense planar-inertial slam with structural constraints". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6521–6528.

[149] Patrick Geneva, Kevin Eckenhoff, Yulin Yang, and Guoquan Huang. "LIPS: Lidar-inertial 3d plane slam". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 123–130.

[150] Sunando Sengupta, Eric Greveson, Ali Shahrokni, and Philip HS Torr. "Urban 3d semantic modelling using stereo vision". In: *2013 IEEE International Conference on robotics and Automation*. IEEE. 2013, pp. 580–585.

[151] Alexander Hermans, Georgios Floros, and Bastian Leibe. "Dense 3d semantic mapping of indoor scenes from rgb-d images". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 2631–2638.

[152] Vibhav Vineet, Ondrej Miksik, Morten Lidegaard, Matthias Nießner, Stuart Golodetz, Victor A Prisacariu, Olaf Kähler, David W Murray, Shahram Izadi, Patrick Pérez, et al. "Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 75–82.

[153] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and James M Rehg. "Joint semantic segmentation and 3d reconstruction from monocular video". In: *European Conference on Computer Vision*. Springer. 2014, pp. 703–718.

[154] Jingming Dong, Xiaohan Fei, and Stefano Soatto. "Visual-inertial-semantic scene representation for 3D object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 960–970.

[155] Sudeep Pillai and John Leonard. "Monocular slam supported object recognition". In: *Robotics: Science and systems* (2015).

[156] Alejo Concha and Javier Civera. "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 5686–5693.

[157] Yasutaka Furukawa and Jean Ponce. "Accurate, dense, and robust multiview stereopsis". In: *IEEE transactions on pattern analysis and machine intelligence* 32.8 (2009), pp. 1362–1376.

[158] Pedro Pinies, Lina Maria Paz, and Paul Newman. "Dense mono reconstruction: Living with the pain of the plain plane". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 5226–5231.

[159] Dorian Gálvez-López, Marta Salas, Juan D Tardós, and JMM Montiel. "Real-time monocular object slam". In: *Robotics and Autonomous Systems* 75 (2016), pp. 435–449.

[160] Sid Yingze Bao, Mohit Bagra, Yu-Wei Chao, and Silvio Savarese. "Semantic structure from motion with points, regions, and objects". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 2703–2710.

[161] Duncan P Frost, Olaf Kähler, and David W Murray. "Object-aware bundle adjustment for correcting monocular scale drift". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 4770–4776.

[162] Edgar Sucar and Jean-Bernard Hayet. "Bayesian scale estimation for monocular SLAM based on generic object detection for correcting scale drift". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–7.

[163] Cosimo Rubino, Marco Crocco, and Alessio Del Bue. "3d object localisation from multiview image detections". In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2017), pp. 1281–1294.

[164] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. "QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM". In: *IEEE Robotics and Automation Letters* 4.1 (2018), pp. 1–8.

[165] Jeong-Kyun Lee, Jaewon Yea, Min-Gyu Park, and Kuk-Jin Yoon. "Joint layout estimation and global multi-view registration for indoor reconstruction". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 162–171.

[166] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. "Fusion++: Volumetric object-level slam". In: *2018 international conference on 3D vision (3DV)*. IEEE. 2018, pp. 32–41.

[167] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. "Simultaneous localization, mapping and moving object tracking". In: *The International Journal of Robotics Research* 26.9 (2007), pp. 889–916.

[168] Dirk Hahnel, Dirk Schulz, and Wolfram Burgard. “Map building with mobile robots in populated environments”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.* Vol. 1. IEEE. 2002, pp. 496–501.

[169] Dirk Hahnel, Rudolph Triebel, Wolfram Burgard, and Sebastian Thrun. “Map building with mobile robots in dynamic environments”. In: *Proceedings of IEEE International Conference on Robotics and Automation, (ICRA) 2003.* Vol. 2. IEEE. 2003, pp. 1557–1563.

[170] Denis F Wolf and Gaurav S Sukhatme. “Mobile robot simultaneous localization and mapping in dynamic environments”. In: *Autonomous Robots* 19.1 (2005), pp. 53–65.

[171] Huijing Zhao, Masaki Chiba, Ryosuke Shibasaki, Xiaowei Shao, Jinshi Cui, and Hongbin Zha. “SLAM in a dynamic large outdoor environment using a laser scanner”. In: *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.* IEEE. 2008, pp. 1455–1462.

[172] Berta Bescos, José M Fácil, Javier Civera, and José Neira. “DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4076–4083.

[173] Chieh-Chih Wang, Charles Thorpe, and Sebastian Thrun. “Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas”. In: *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA’03*. Vol. 1. IEEE. 2003, pp. 842–849.

[174] Isaac Miller and Mark Campbell. “Rao-blackwellized particle filtering for mapping dynamic environments”. In: *IEEE International Conference on Robotics and Automation, 2007*. IEEE. 2007, pp. 3862–3869.

[175] John G Rogers, Alexander JB Trevor, Carlos Nieto-Granda, and Henrik I Christensen. “Slam with expectation maximization for moveable object tracking”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010*. IEEE. 2010, pp. 2077–2082.

[176] Abhijit Kundu, K Madhava Krishna, and CV Jawahar. “Realtime multibody visual SLAM with a smoothly moving monocular camera”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2080–2087.

[177] Charles Bibby and Ian Reid. “Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association”. In: *Proceedings of Robotics: Science and Systems*. 2007.

[178] Kevin M Judd, Jonathan D Gammell, and Paul Newman. "Multimotion visual odometry (MVO): Simultaneous estimation of camera and third-party motions". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3949–3956.

[179] Danping Zou and Ping Tan. "Coslam: Collaborative visual slam in dynamic environments". In: *IEEE transactions on pattern analysis and machine intelligence* 35.2 (2013), pp. 354–366.

[180] N Dinesh Reddy, Prateek Singhal, Visesh Chari, and K Madhava Krishna. "Dynamic body VSLAM with semantic constraints". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015*. IEEE. 2015, pp. 1897–1904.

[181] Pablo F Alcantarilla, José J Yebes, Javier Almazán, and Luis M Bergasa. "On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1290–1297.

[182] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. "Robust monocular SLAM in dynamic environments". In: *IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2013*. IEEE. 2013, pp. 209–218.

[183] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. "MOTS: Multi-object tracking and segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7942–7951.

[184] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[185] Zichao Zhang, Henri Rebecq, Christian Forster, and Davide Scaramuzza. "Benefit of Large Field-of-View Cameras for Visual Odometry". In: *IEEE International Conference on Robotics and Automation (ICRA), 2016*. IEEE. 2016.

[186] David H Warren and Edward R Strelow. *Electronic spatial sensing for the blind: contributions from perception, rehabilitation, and computer vision*. Vol. 99. Springer Science & Business Media, 2013.

[187] Berthold KP Horn and Brian G Schunck. "Determining optical flow". In: *Techniques and Applications of Image Understanding*. Vol. 281. International Society for Optics and Photonics. 1981, pp. 319–331.

[188] James J Gibson. “The perception of the visual world.” In: (1950).

[189] Ashit Talukder and Larry Matthies. “Real-time detection of moving objects from moving vehicles using dense stereo and optical flow”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 4. IEEE. 2004, pp. 3718–3725.

[190] Abhishek Kumar Chauhan and Prashant Krishan. “Moving object tracking using gaussian mixture model and optical flow”. In: *International Journal of Advanced Research in Computer Science and Software Engineering* 3.4 (2013).

[191] Naoya Ohnishi and Atsushi Imiya. “Dominant plane detection from optical flow for robot navigation”. In: *Pattern Recognition Letters* 27.9 (2006), pp. 1009–1021.

[192] Hernán Badino, Akihiro Yamamoto, and Takeo Kanade. “Visual odometry by multi-frame feature integration”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2013, pp. 222–229.

[193] Jean-Luc Stevens and Robert Mahony. “Vision based forward sensitive reactive control for a quadrotor VTOL”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 5232–5238.

[194] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[195] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. “Surface reconstruction from unorganized points”. In: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 1992, pp. 71–78.

[196] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. “Digging into Self-Supervised Monocular Depth Prediction”. In: (2019).

[197] Zhaoyang Lv, Kihwan Kim, Alejandro Troccoli, James Rehg, and Jan Kautz. “Learning Rigidity in Dynamic Scenes with a Moving Camera for 3D Motion Field Estimation”. In: *ECCV*. 2018.

[198] Lukas Polok, Marek Solony, Viorela Ila, Pavel Smrz, and Pavel Zemcik. “Efficient implementation for block matrix operations for nonlinear least squares problems in robotic applications”. In: *IEEE International Conference on Robotics and Automation (ICRA), 2013*. IEEE. 2013, pp. 2263–2269.

[199] Sameer Agarwal, Keir Mierle, and Others. *Ceres Solver*. http://ceres-solver.org.

[200] Viorela Ila, Lukáš Polok, Marek Šolony, and Pavel Svoboda. "SLAM++-A highly efficient and temporally scalable incremental SLAM framework". In: *International Journal of Robotics Research* Online First.0 (2017), pp. 1–21. DOI: 10.1177/0278364917691110.

[201] Mladen Mazuran, Gian Diego Tipaldi, Luciano Spinello, Wolfram Burgard, and Cyrill Stachniss. "A statistical measure for map consistency in slam". In: *IEEE International Conference on Robotics and Automation (ICRA), 2014*. IEEE. 2014, pp. 3650–3655.

[202] Julian Straub, Guy Rosman, Oren Freifeld, John J Leonard, and John W Fisher. "A mixture of manhattan frames: Beyond the manhattan world". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014*. IEEE. 2014, pp. 3770–3777.

[203] David Eigen and Rob Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2650–2658.

[204] Xiaolong Wang, David Fouhey, and Abhinav Gupta. "Designing deep networks for surface normal estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 539–547.

[205] Jin Zeng, Yanfeng Tong, Yunmu Huang, Qiong Yan, Wenxiu Sun, Jing Chen, and Yongtian Wang. "Deep surface normal estimation with hierarchical RGB-D fusion". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6153–6162.

[206] Huangying Zhan, Chamara Saroj Weerasekera, Ravi Garg, and Ian Reid. "Self-supervised learning for single view depth and surface normal estimation". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 4811–4817.

[207] Keisuke Tateno, Federico Tombari, and Nassir Navab. "Real-time and scalable incremental segmentation on dense SLAM". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015*. IEEE. 2015, pp. 4465–4472.

[208] David Marr. "Vision: A computational investigation into the human representation and processing of visual information". In: *Henry Holt and Company Inc., New York, NY* 2.4.2 (1982).

[209] Beipeng Mu, Shih-Yuan Liu, Liam Paull, John Leonard, and Jonathan P How. "SLAM with objects using a nonparametric pose graph". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016*. IEEE. 2016, pp. 4602–4609.

[210] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. "MOT16: A Benchmark for Multi-Object Tracking". In: *arXiv:1603.00831 [cs]* (Mar. 2016). arXiv: 1603.00831. URL: http://arxiv.org/abs/1603.00831.

[211] Arunkumar Byravan and Dieter Fox. "Se3-nets: Learning rigid body motion using deep neural networks". In: *IEEE International Conference on Robotics and Automation (ICRA), 2017*. IEEE. 2017, pp. 173–180.

[212] Paul Wohlhart and Vincent Lepetit. "Learning descriptors for object recognition and 3d pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3109–3118.

[213] Gregory S. Chirikjian, Robert Mahony, Sipu Ruan, and Jochen Trumpf. "Pose changes from a different point of view". In: *Proceedings of the ASME International Design Engineering Technical Conferences (IDETC) 2017*. ASME. 2017.

[214] Lukas Polok, Viorela Ila, Marek Solony, Pavel Smrz, and Pavel Zemcik. "Incremental Block Cholesky Factorization for Nonlinear Least Squares in Robotics". In: *Proceedings of Robotics: Science and Systems*. Berlin, Germany, 2013. DOI: 10.15607/RSS.2013.IX.042.

[215] Raúl Mur-Artal and Juan D. Tardós. "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[216] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM". In: *arXiv preprint arXiv:2007.11898* (2020).

[217] A Gaidon, Q Wang, Y Cabon, and E Vig. "Virtual Worlds as Proxy for Multi-Object Tracking Analysis". In: *CVPR*. 2016.

[218] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. "Efficient joint segmentation, occlusion labeling, stereo and flow estimation". In: *European Conference on Computer Vision*. Springer. 2014, pp. 756–771.

[219] Frank Dellaert. *Factor graphs and GTSAM: A hands-on introduction*. Tech. rep. Georgia Institute of Technology, 2012.

[220] Hauke Strasdat, Andrew J Davison, JM Martìnez Montiel, and Kurt Konolige. "Double window optimisation for constant time visual SLAM". In: *IEEE International Conference on Computer Vision (ICCV), 2011*. IEEE. 2011, pp. 2352–2359.

[221] Viorela Ila, Josep M Porta, and Juan Andrade-Cetto. "Information-based compact pose SLAM". In: *IEEE Transactions on Robotics* 26.1 (2010), pp. 78–93.

[222] David Nistér, Oleg Naroditsky, and James Bergen. "Visual odometry". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. Ieee. 2004, pp. I–I.

[223] Peter J Huber. "Robust estimation of a location parameter". In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[224] Mina Henein, Jun Zhang, Robert Mahony, and Viorela Ila. "Dynamic SLAM: The Need For Speed". In: *2020 IEEE International Conference on Robotics and Automation (ICRA). To appear.* (2020).

[225] Tong Ke and Stergios I Roumeliotis. "An efficient algebraic solution to the perspective-three-point problem". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7225–7233.

[226] Kevin Michael Judd and Jonathan D Gammell. "The Oxford multimotion dataset: Multiple SE (3) motions with ground truth". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 800–807.

[227] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* (2013).

[228] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[229] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4040–4048.

[230] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. "A naturalistic open source movie for optical flow evaluation". In: *European conference on computer vision*. Springer. 2012, pp. 611–625.

[231] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network". In: *Advances in neural information processing systems*. 2014, pp. 2366–2374.

[232] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. "YOLACT: real-time instance segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9157–9166.

[233] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. "A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization". In: 2020. URL: http://mmlab.ie.cuhk.edu.hk/projects/LiteFlowNet/.

[234] Gilbert Strang. *Linear algebra and its applications*. 04; QA184, S8. 1976.