Robust Visual Odometry and Dynamic Scene Modelling

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

Presented by

Jun Zhang

Accepted on the recommendation of

Dr. Viorela Ila, University of Sydney
A/Prof. Laurent Kneip, ShanghaiTech University
Prof. Hongdong Li, Australian National University
Prof. Robert Mahony, Australian National University

May 2021

Except where otherwise indicated, this thesis is my own original work.

Jun Zhang
14 May 2021

To my beloved families.

Acknowledgments

When I was a young boy in my adolescence, who showed up on the soccer field every afternoon while other peers were worrying about their studies, I could hardly imagine the moment when I would be writing the thesis for a PhD degree, but it does happen now.

In the last year of my undergraduate period, when Prof. Zhenbao Liu was showing his fancy demo of detecting healthy rabbits in the class, it was my first time to understand what research is and feel that doing research is such a cool thing. After that I decided to study a postgraduate degree, under supervision by Zhenbao. During the first year, I mainly focused on taking professional courses, reading related academic papers, and training programming skills (Matlab and C++). It was until the time when Zhenbao recommended me to work with A/Prof. Zouhui Lian in Peking University as a visiting student, I started to get involved in a research project on mesh segmentation for 3D shape retrieval in the area of computer graphics. It was quite a pleasant time working with Zouhui, and he taught me many useful skills for academic research. With this wonderful experience, I made up my mind to go further to pursuit a PhD degree in Australian National University.

The incredible journey of my PhD began since the November of 2016. I would like to thank A/Prof. Laurent Kneip and Prof. Hongdong Li for giving me the opportunity to work in one of the top computer vision and robotics groups in the world - Australian Centre for Robotic Vision (ACRV). Although Laurent left for a new position in ShanghaiTech University soon after I started, he led me patiently into the field of robotic vision during the short-term supervision. Later Dr. Viorela Ila became my primary supervisor, who introduced me into the community of visual odometry (VO) and simultaneous localization and mapping (SLAM). VO/SLAM is commonly known to be a extremely challenging topic from the perspective of both theory and practice. But I feel so lucky to have my supervisor Viorela to walk me through this tough road. She is always in patience and would like to share whatever she knows with me. The memories we were working together towards deadlines will never be forgotten. In particular, I would like to sincerely thank Prof. Robert Mahony for being my major advisor since the third year of my PhD. I have learned so much from him, not only his vision of robotics and control, but also the way of how to be a good researcher.

I also would like to show my appreciation to all my colleagues and friends in the laboratory, for their sharing and giving. It is really my pleasure to meet you, Yi Zhou, Liu Liu, Liyuan Pan, Jing Zhang, Hongguang Zhang, Yiran Zhong, Yonhon Ng, Zheyu Zhuang, Pieter Van Goor, Ziang Cheng, Mina Henein, Xin Yu, Ziwei Wang, Yujiao Shi, Gerard Kennedy, Shah Chowdhury, Tong Zhang, Medhani Menikdiwela, Shihao Jiang, Wei Mao, Ryan Pike, *etc.*

Finally, I am most grateful to my families for their support at all time, especially great thanks to my wife, Qingtan Wang, for her company and care all the time.

Abstract

Image-based estimation of camera trajectory, known as visual odometry (VO), has been a popular solution for robot navigation in the past decade due to its low-cost and widely applicable properties. The problem of tracking self-motion as well as motion of objects in the scene using information from a camera is known as multi-body visual odometry and is a challenging task.

The performance of VO is heavily sensitive to poor imaging conditions (*i.e.*, direct sunlight, shadow and image blur), which limits its feasibility in many challenging scenarios. Current VO solutions can provide accurate camera motion estimation in largely static scene. However, the deployment of robotic systems in our daily lives requires systems to work in significantly more complex, dynamic environment. This thesis aims to develop robust VO solutions against two challenging cases, underwater and highly dynamic environments, by extensively analysing and overcoming the difficulties in both cases to achieve accurate ego-motion estimation. Furthermore, to better understand and exploit dynamic scene information, this thesis also investigates the motion of moving objects in dynamic scene, and presents a novel way to integrate ego and object motion estimation into a single framework.

In particular, the problem of VO in underwater is challenging due to poor imaging condition and inconsistent motion caused by water flow. This thesis intensively tests and evaluates possible solutions to the mentioned issues, and proposes a stereo underwater VO system that is able to robustly and accurately localize the autonomous underwater vehicle (AUV).

Visual odometry in dynamic environment is challenging because dynamic parts of the scene violate the static world assumption fundamental in most classical visual odometry algorithms. If moving parts of a scene dominate the static scene, off-the-shelf VO systems either fail completely or return poor quality trajectory estimation. Most existing techniques try to simplify the problem by removing dynamic information. Arguably, in most scenarios, the dynamics corresponds to a finite number of individual objects that are rigid or piecewise rigid, and their motions can be tracked and estimated in the same way as the ego-motion. With this consideration, the thesis proposes a brand new way to model and estimate object motion, and introduces a novel multi-body VO system that addresses the problem of tracking of both ego and object motion in dynamic outdoor scenes.

Based on the proposed multi-body VO framework, this thesis also exploits the spatial and temporal relationships between the camera and object motions, as well as static and dynamic structures, to obtain more consistent and accurate estimations. To this end, the thesis introduces a novel visual dynamic object-aware SLAM system, that is able to achieve robust multiple moving objects tracking, accurate estimation of full $\mathrm{SE}(3)$ object motions, and extract inherent linear velocity information of moving objects, along with an accurate robot localisation and mapping of environment structure. The performance of the proposed system is demonstrated on real datasets, showing its capability to resolve rigid object motion estimation and yielding results that outperform state-of-the-art algorithms by an order of magnitude in urban driving scenarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Contributions

I, robot
Don't have the capacity
To dream about tomorrow
But I never spell a word wrong

Louis Shalako, "Mr. Robot", Selected Poems.

Sight is a human's most highly developed sense. It is important for learning, using tools and devices, moving about, and enjoying human endeavor. Vision stems from the eyes and the brain functioning as an integrated system of awesome complexity. An image is sensed by over a hundred million visual cells in each eye, processed in the retina and transmitted through fibers in each optic nerve to the centres of vision in the brain. Each second the brain performs about a hundred billion "floating point logical operations" to transform the two dimensional image into symbols that encapsulate the solid reality of the three dimensional space. Modern cameras work in the similar principle as our eyes do, whereas here imaging chips play the role of visual cells. With these artificial eyes, researchers expect to endow robots the ability to perceive the world visually as we humans do, by introducing 'algorithms' into robots' brains to process the imaging data.

For an autonomous mobile robot to move around in an unknown environment and perform complex tasks, it needs in priority to know where it is and what the surrounding environment looks like. These two basic tasks, originally coined as *Simultaneous Localization And Mapping* (SLAM) by Leonard and Durrant-Whyte [1991a], has been an active field of research for more than three decades. Today, SLAM algorithms are a standard module on mobile robots, which have been implemented using different sensor modalities, such as laser range finders, sonars, or cameras. Visual SLAM, where the primary sensors are cameras, becomes a popular solution in recent years, since camera sensors are small, low-cost, power efficient and ubiquitous. SLAM (visual SLAM in particular) aims to obtain a global, consistent estimate of the camera trajectory, as well as a map of the environment, via detecting previously visited area and subsequently refining the whole trajectory (loop closure). If the goal is to incrementally recover the camera trajectory with scene reconstruction, and potentially optimize only over a partial trajectory and local map, we speak of *Visual Odometry* (VO) Nistér et al. [2004]. VO only focuses on local consistency of the trajectory and the local map is used to reduce the drift in the estimate of local trajectory, whereas SLAM is concerned with the global map consis-

tency. This thesis mainly aims at developing visual odometry systems using stereo or RGB-D cameras, with one of them being extended to a full visual SLAM system. In the rest of this thesis I use the acronym VO/SLAM to refer to the combined subject domain comprising all visual spatial awareness algorithms including visual odometry and visual SLAM algorithms.

1.1 Importance of Visual Odometry/SLAM for Robotics

In recent years, VO/SLAM technologies have been used in a wide variety of application domains due to the simple, inexpensive and commonly reachable sensor configuration. Specifically, it is used for accurate camera pose estimation in augmented reality (AR) systems Comport et al. [2006]; Runz et al. [2018], where the configuration of these systems can be generic items such as camera-mounted tablets or smart phones. Another typical example is navigation for micro aerial vehicles (MAVs) Forster et al. [2014]; Qin et al. [2018]: small flying robots required light-weight on-board sensors, which makes camera sensor the most appealing option. With the help of VO/SLAM techniques, MAVs are able to achieve self-navigation with low drifts in attitude and position.

(a) Augmented reality. (b) DJI commercial MAVs. (c) NASA curiosity rover.

(d) Reef monitoring robot. (e) Dyson robotic vacuum cleaner. (f) Waymo autonomous car.

Figure 1.1: **Examples of VO/SLAM applications.**

Visual odometry/SLAM also plays a significant role in the field of robotic control and navigation when no external reference signal is available. For instance, a stereo VO system Moravec [1980a] is built for the NASA Mars exploration program in 1980, as an alternative to wheel odometry which is always affected by slippage in uneven terrains. The demand for ocean exploitation requires autonomous underwater vehicles (AUVs) to be able to perform self-localization in underwater environment Wirth et al. [2013]; Burguera et al. [2014]. Global Positioning System (GPS) is a well-established technology for tracking and navigations. However, GPS-based systems are not useful indoors, or places covered by limited sky views, such as forest or urban canyon, *etc.*

Autonomous driving in urban scenarios represents the biggest real application to visual odometry/SLAM nowadays. Autonomous vehicles could potentially use VO/SLAM systems for mapping and understanding the world around them Ros et al. [2012]; Lategahn et al. [2011]. Field robots in agriculture, as well as drones, can use the same technology to independently travel around crop fields Dong et al. [2017]; Cheein et al. [2011]. Other applications cover the area of service, social and medical robots. This includes vacuum cleaning and lawn-mowing robots for housekeeping, robot companions and assistants for elderly people, nano-robots for minimally invasive surgeries (MIS), to name a few.

1.2 Visual Odometry/SLAM Challenges

The ability to sense the location of a camera, as well as the environment around it, without knowing any data points beforehand is incredibly difficult. In particular, VO/SLAM algorithms are sensitive to environmental changes such as lighting conditions, surrounding texture, dynamic factor and the presence of rain, snow, *etc.* Other conditions that lead to poor tracking data are motion blur, shadows, visual similarity, degenerate configuration, and occlusions. Along with these, some man-made errors are also brought into the data during image acquisition and processing steps such as lens distortion and calibration, feature matching, triangulation. Two real world examples that are highly challenging to visual odometry/SLAM algorithms are underwater and dynamic scenes.

1.2.1 Underwater Scenario

Visual odometry/SLAM becomes a popular solution for localizing AUVs in underwater, since it is a cost effective alternative to other sensors such as sonar. Nevertheless, unlike terrestrial environment where traditional VO/SLAM algorithms work quite well, localization from vision becomes more complex in underwater and state-of-the-art algorithms fail when the operating conditions become too harsh. One of the main reasons causing failure is the poor imaging condition. The light attenuation caused by the water medium shortens the visual perception to limited distance. Besides, the propagation of light is backscattered by floating particles, causing turbidity effects on the captured images. The water medium also cause the motion of AUV to be inconsistent with oscillation, which makes it harder to estimate the motion and brings motion blur to the image.

1.2.2 Dynamic Scenario

Dynamic scenes are quite common in the daily world we live. Moving creatures will come along, no matter artificial or real, as long as there are human activities. Visual navigation in non-static environments becomes challenging because the dynamic parts violate the static scene assumption. If moving parts of a scene dominate the static scene, off-the-shelf VO/SLAM systems either fail completely or return poor quality trajectory estimation.

In most scenarios, the dynamics corresponds to a finite number of individual objects that are rigid or piecewise rigid, and their motions can be tracked and estimated in the same way as the ego-motion. The problem of simultaneously tracking self-motion and motion of objects

in the scene using only cameras is known as multi-body visual odometry. Multi-body VO is a challenging task because it requires the system to solve together the problems of motion segmentation, multi-object tracking, and motion estimation. Each individual problem is still an active research area in the community of computer vision and robotics.

1.3 Contributions

The scope of this thesis is to develop, test and evaluate visual odometry algorithms, that are able to robustly localize the robot in two challenging scenarios: underwater and dynamic environments. For dynamic scenarios, the thesis further proposes to segment the dynamic scene into multiple individual objects and track their motions. In the end, a single framework is formulated to explore the spatial and temporal relationships between the camera and object motions, as well as static and dynamic structures, to get more consistent and accurate estimations. Overall, the contributions of this thesis are summarised as the following three main parts:

- This thesis proposes a robust and effective stereo VO system, which overcomes the difficulties of poor imaging condition and inconsistent motion caused by water flow, and accurately localizes the AUV. The thesis intensively tests and evaluates possible solutions for three problems in underwater: feature association, inconsistent motion and image enhancement, and carefully modified this system to accommodate the aforementioned challenging issues. Experimental results demonstrate that the proposed system outperforms existing VO systems in underwater environment.

- This thesis proposes a novel way to model and estimate object motion, and introduces a robust solution to achieve accurate ego and object motion estimation with consistent track-ability for dynamic multi-body visual odometry. A compact and effective framework is proposed leveraging recent advances in semantic instance-level segmentation and accurate optical flow estimation. A novel formulation, jointly optimizing $\mathrm{SE}(3)$ motion and optical flow is introduced that improves the quality of the tracked points and the motion estimation accuracy. The proposed approach is evaluated on the virtual dataset and tested on the real dataset, demonstrating its applicability to autonomous driving applications.

- This thesis extends the second part to a full visual SLAM system, which integrates information about dynamic and static structures in the environment into a single estimation framework, resulting in accurate robot pose and spatio-temporal maps estimation. The system is able to achieve robust moving object tracking, accurate estimation of full $\mathrm{SE}(3)$ pose change of dynamic objects with extraction of their linear velocity information, as well as accurate robot localisation and mapping of environment structure. The performance of the proposed system is demonstrated on both real indoor and outdoor datasets. Results show consistent and substantial improvements over state-of-the-art algorithms.

1.4 Thesis Outline

The remainder of this thesis is structured as follows:

- Chapter 2 reviews related literature and provides context for the work presented.
- Chapter 3 introduces the necessary mathematical preliminaries, notations and problem formulations required for later chapters.
- Chapter 4 describes a robust and effective stereo underwater VO system that addresses the issues of feature association and motion inconsistency in underwater, and accurately localizes the AUV.
- Chapter 5 details a novel multi-body visual odometry pipeline that solves the problem of tracking of both ego and object motion in dynamic outdoor scenes.
- Chapter 6 presents a novel feature-based dynamic object-aware SLAM system, that is able to localise the robot, estimate a full $\mathrm{SE}(3)$ pose change of moving objects and build a full map of the environment containing both static and dynamic structure.
- Chapter 7 summarises the outputs of this thesis and offers concluding remarks.

1.5 Publication

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

1.5.1 Conferences

- **Robust Visual Odometry in Underwater Environment.**
  Jun Zhang, Viorela Ila and Laurent Kneip.
  2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO), Kobe, 2018, pp. 1-9.

- **Multi-frame Motion Segmentation for Dynamic Scene Modelling.**
  Jun Zhang and Viorela Ila.
  In the 20th Australasian Conference on Robotics and Automation (ACRA), 2018.

- **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, pp. 2123-2129.

- **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, pp. 5017-5023.

1.5.2 Journals

- **VDO-SLAM: A Visual Dynamic Object-aware SLAM System.**
  Jun Zhang*, Mina Henein*, Robert Mahony and Viorela Ila.
  ArXiv Preprint arXiv:2005.11052.
  ($*$ equal contribution)

1.5.3 Misc

Publicly available source code:

- **Multiple Motion Tracking**
  https://github.com/halajun/multimot_track
- **VDO-SLAM**
  https://github.com/halajun/VDO_SLAM

Chapter 2

Literature Review

In this chapter, a brief introduction towards history of VO/SLAM is first presented, along with the detailed reviews of modern geometric approaches. Then the proposed solutions for underwater and dynamic environments in the literature are discussed.

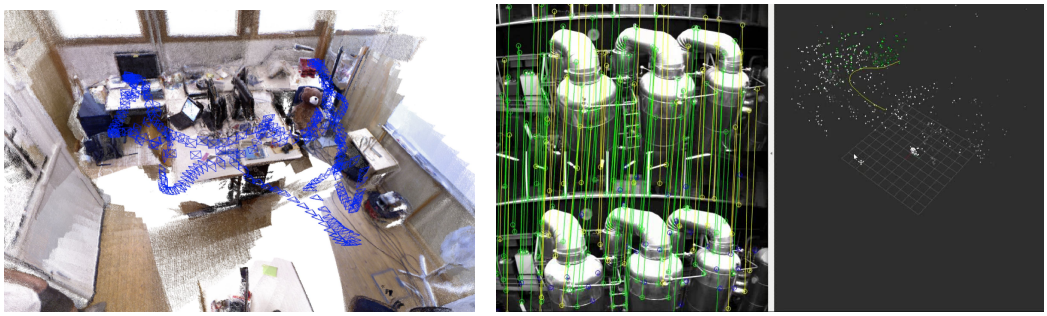
2.1 History of VO/SLAM

The term VO was first used by Srinivasan et al. [1997] to define motion orientation of honey bees. The history of VO, however, dates back to early works Moravec [1980b]; Lacroix et al. [1999] motivated by the NASA Mars exploration program to provide all-terrain rovers with the capability to measure their 6-degree-of-freedom (DoF) motion in the presence of wheel slippage in uneven and rough terrains. In the following two decades, the research on VO was led by NASA in preparation for the 2004 Mars mission Matthies and Shafer [1987]; Matthies [1989]; Lacroix et al. [1999]; Olson et al. [2000]. But it is after the work Visual Odometry from Nistér et al. [2004] was published that the study of VO becomes popular in the areas of robotics and computer vision. The proposed methodology by Nister sooner became almost 'golden method' for feature-based VO, and is still frequently used in the front-end of visual SLAM today.

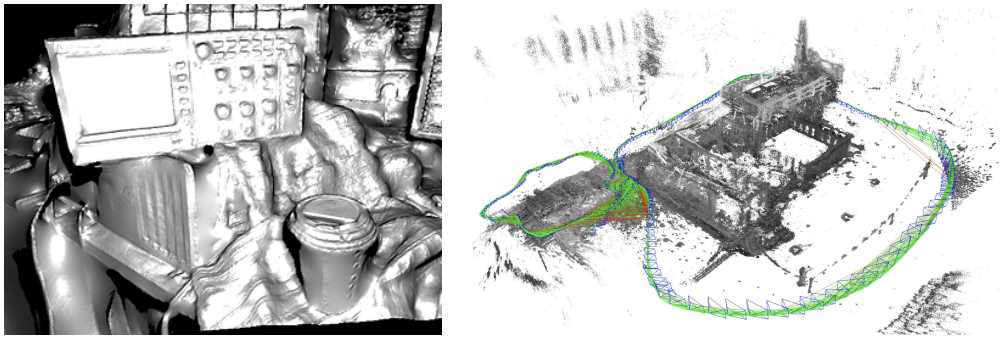
According to Durrant-Whyte and Bailey [2006], the research on SLAM dates back to 1986, when the idea of using estimation-theoretic methods for robot localization and mapping were first discussed. Yet it was until 1995 that the structure of the SLAM problem, the convergence result and the coining of the acronym SLAM were presented. Early SLAM works mostly relied on odometry and laser/ultrasonic sensors as perception input, such as Leonard and Durrant-Whyte [1991b] (with ultrasonic), and Gutmann and Konolige [1999] (with laser). The theoretical fundamental and prototype of traditional Bayesian filtering based SLAM framework emerged in 1990s, and the notable work was from Thrun et al. [1998], as well as Dissanayake et al. [2001] sooner.

Landmark work of bringing vision into SLAM came from A. Davison's Mono-SLAM in early 2000s Davison [2003]; Davison and Murray [2002]; Davison et al. [2007], which in some way set the standard framework for traditional Bayesian filtering-based visual SLAM implementation. The problem of filtering-based SLAM lies in high computation effort that grows quadratically with the number of landmarks increasing. In 2007, Georg Klein and David Murray proposed Parallel Tracking and Mapping (PTAM) Klein and Murray [2007], introducing

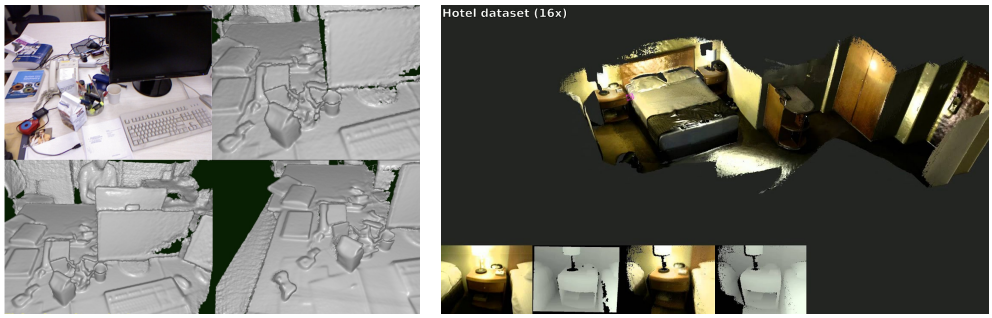
an efficient keyframe-based *Bundle Adjustment* (BA) instead of Bayesian filtering for pose and map refinement, outperforms Mono-SLAM and the like in both efficiency and accuracy. As keyframes and map points are formulated as nodes in a graph and optimized to minimize their measurement errors Kümmerle et al. [2011], keyframe-based map refinement with BA is also referred as graph-based SLAM, which becomes a new standard framework and has great influence in modern visual SLAM research.



(a) ORB-SLAM2. (b) OKVIS.



(c) DTAM. (d) LSD-SLAM.



(e) KinectFusion. (f) ElasticFusion.

Figure 2.1: **Different feature-based (a,b), direct (c,d) and 3D point registration based (e,f) VO/SLAM systems.**

2.2 State-of-the-art VO/SLAM

On the foundation laid by the predecessors, state-of-the-art systems keep making improvement from perspectives of both theory and system engineering. According to different formulations, they could be clustered into three categories: feature-based methods, direct methods, and methods based on 3D point set registration.

2.2.1 Feature-based Methods

Featured-based methods process the raw image information into intermediate representations such as feature points, lines, curves or planes to be used in the estimation as landmarks. The estimation then only depends on the extracted features and normally a geometric cost is used. Typical feature point based methods include ORB-SLAM Mur-Artal et al. [2015], LIBVISO Geiger et al. [2011], OKVIS Leutenegger et al. [2015] and FOVIS Huang et al. [2017], *etc.* In addition to points, other primitives such as lines Pumarola et al. [2017]; Sola et al. [2012], curves Zhou et al. [2018] and planes Henein et al. [2017]; Hsiao et al. [2017] have also been widely studied in VO/SLAM.

Feature-based methods are robust to brightness inconsistencies and large view-point changes among consecutive frames. Nevertheless, feature extraction and matching bring considerable computational cost. Besides, few features can be extracted and matched in low-texture and blurring scenes, which could lead to losing track easily.

2.2.2 Direct Methods

Direct methods Comport et al. [2007, 2010] skip the feature extraction step and directly utilise the raw sensor input such as image pixel intensity, and formulate the optimization cost as photometric error, which has been proven to be more effective in textureless environments. One of the seminal methods is DTAM Newcombe et al. [2011b], which is able to estimate a dense depth map at each keyframe. However, the process of dense pixel points is computationally expensive, resulting in dependency on GPU hardware to achieve real-time performance. LSD-SLAM Engel et al. [2014] is another leading method in this category, which estimates depth at pixels solely near image boundaries and recovers a 'semi-dense' map to speed up the process. This thereby becomes the first direct visual SLAM system that can run in real-time on a CPU. More recently, Engel *et al.* propose the direct sparse odometry (DSO) Engel et al. [2017]. In DSO, sparse points with high intensity gradient are sampled across the whole image and used for estimation. To handle imperfect brightness constancy, a photometric calibration pipeline is introduced to recover the irradiance images and therefore increase the tracking accuracy.

One significant advantage of direct methods is that they can perform relatively dense 3D reconstruction, and more image information is used in tracking. However, they become unstable in rapid light changing environment, as the brightness constancy assumption does not always hold. Moreover, compared to geometric error, photometric error is highly non-linear and non-convex, hence it requires good initialisation of map and camera poses to ensure convergence.

2.2.3 3D Point Registration based Methods

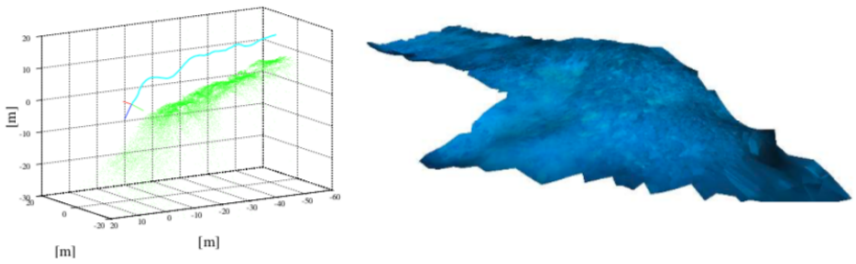
3D point set registration is a traditional problem that has been extensively investigated in the computer vision community. Here, we only limit the discussion to the related VO/SLAM methods that process mainly rigid, geometric information. These methods are commonly based on an *iterative closest point* (ICP) algorithm proposed by Besl and McKay [1992], which registers two 3D point sets through iterative minimization of the *sum of square differences* (SSD) distances between spatial neighbours, and recover the transformation between the sets. One of representative methods is KinectFusion Newcombe et al. [2011a]. The method proposes to represent the 3D structure of the environment in a voxel space. The 3D structure is reconstructed by combining obtained depth maps in the voxel space, and camera motion is estimated by ICP using an estimated 3D structure and the input depth map. The ICP algorithm and its close derivatives Pomerleau et al. [2011, 2013] still represent the algorithm of choice for 3D point registration based VO/SLAM Whelan et al. [2012, 2015]; Ireta Muñoz and Comport [2018] today.

2.3 VO/SLAM in Underwater Environment

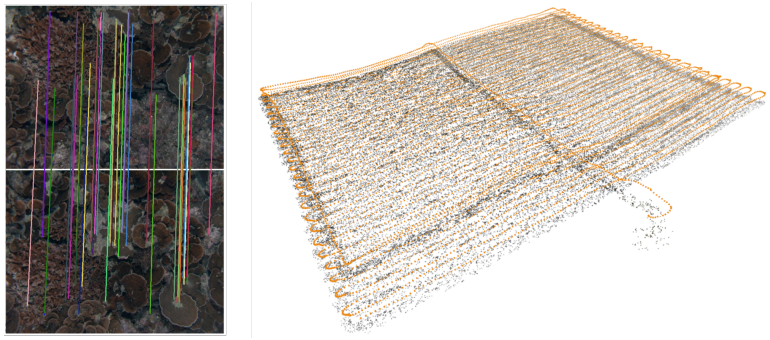
Acoustic sensors have been the first choice for underwater SLAM and navigation for a long time. Using such sensors, *e.g.*, Doppler Velocity Log (DVL), ultra-short baseline (USBL), and multi-beam imaging sonar, many underwater navigation algorithms Leonard and Durrant-Whyte [2012]; Snyder [2010]; Johannsson et al. [2010]; Rigby et al. [2006] have been developed. Acoustic positioning systems, however, require careful calibration of the sound velocity, as they suffer from multipath Doppler effects and susceptibility to thermocline; they also have a limited range and accuracy Wu et al. [2015].

Eustice et al. [2008] are among the first to present a successful use of visual information as a complementary sensor for underwater robots localization. They use an Extended Information Filter (EIF) to process dead-reckoning sensors and insert visual constraints based on the 2D-2D matching of points coming from overlapping monocular images. Kim and Eustice [2013] extended their work by adding loop-closure capability through the computation of a visual saliency metric using SIFT Lowe [2004] features. However, in both methods the visual information is only used to bound the localization drift using low-overlap imagery systems, but their systems mainly rely on expensive navigational sensors.

Representative underwater monocular-based methods using cameras at high frame rate are Shkurti et al. [2011] and Burguera et al. [2015]. In their approaches, they fuse visual motion estimation in an *Extended Kalman Filter* (EKF) along with an IMU and a pressure sensor. More recently, Creuze [2017] propose a monocular underwater localization method that does not rely on an EKF framework but iteratively estimates ego-motion by integration of optical flow measurements corrected by an IMU and a pressure sensor. This latter is used to compute the scale factor of the observed scene. Many stereo-vision based systems are also proposed to avoid scale ambiguity in monocular systems. Salvi et al. [2008] introduce a real-time EKF-SLAM system that incorporates a sparsely distributed robust feature selection and 6-DOF pose estimation using only calibrated stereo cameras. Beall et al. [2011] present an accurate 3D reconstruction on a large-scale underwater dataset by performing bundle adjustment



(a) Selective SLAM.



(b) Beall et al. [2011].

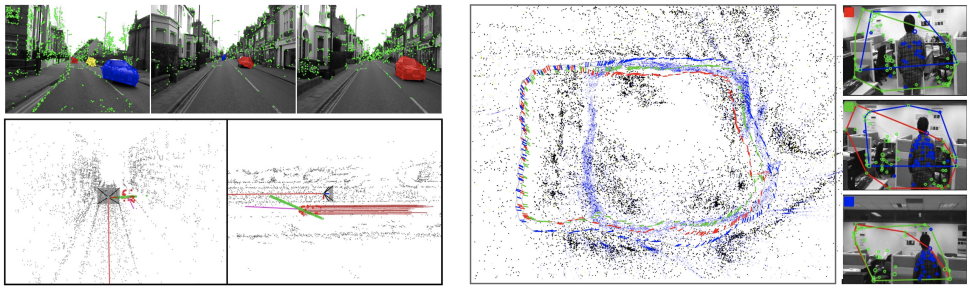
Figure 2.2: **Different underwater VO/SLAM systems.**

over all cameras and a subset of features rather than using a traditional filtering technique. A stereo SLAM framework named Selective SLAM (SSLAM) for autonomous underwater vehicle localization is proposed in Bellavia et al. [2017]. The above-mentioned methods all achieve accurate trajectory estimation in underwater environment. However, the typical problems in underwater scenes: poor imaging quality and inconsistent motion, are not extensively studied and evaluated in details.

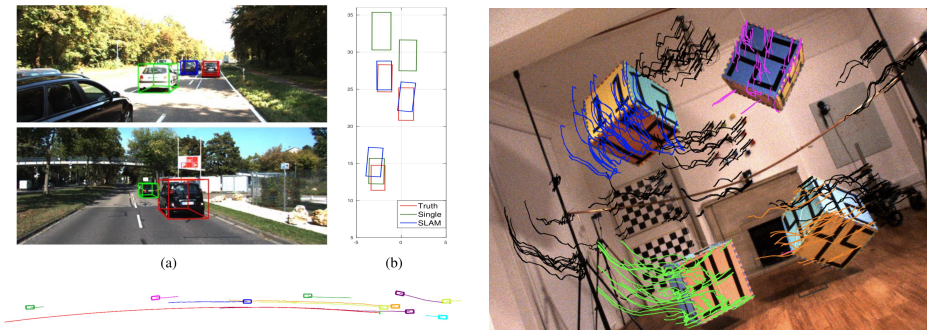
2.4 VO/SLAM in Dynamic Environment

In the past two decades, the study of VO/SLAM for dynamic environments has become more and more popular in the community, with a considerable amount of algorithms being proposed to solve the dynamic VO/SLAM problems. Motivated by different goals to achieve, solutions in the literature can be mainly divided into four categories.

The first category aims to explore robust SLAM against dynamic environments. Early methods in this category (Hahnel et al. [2003]; Alcantarilla et al. [2012]; Tan et al. [2013]) normally detect and remove the information drawn from dynamic foreground, which is seen as degrading the SLAM performance. More recent methods on this track tend to go further by not just removing the dynamic foreground, but also inpainting or reconstructing the static background that is occluded by moving targets. Bescos and Neira [2018] present dynaSLAM that combines classic geometry and deep learning-based models to detect and remove dynamic objects, then inpaint the occluded background with multi-view information of the scene. Simi-



(a) Kundu et al. [2011]. (b) CoSLAM.



(a) (b)

(c) CubeSLAM. (d) Judd et al. [2018].

Figure 2.3: **Different dynamic VO/SLAM systems.**

larly, a Light Field SLAM front-end is proposed by Kaveti and Singh [2020] to reconstruct the occluded static scene via Synthetic Aperture Imaging (SAI) technics. Different from Bescos and Neira [2018], features on the reconstructed static background are also tracked and used to achieve better SLAM performance. The above state-of-the-art solutions achieve robust and accurate estimation by discarding the dynamic information. However, we argue that this information has potential benefits for SLAM if it is properly modelled. Furthermore, understanding dynamic scenes in addition to SLAM is crucial for many other robotics tasks such as planning, control and obstacle avoidance, to name a few.

Approaches of the second category performs SLAM and Moving Objects Tracking (MOT) separately, as an extension to conventional SLAM for dynamic scene understanding (Wang et al. [2007]; Kundu et al. [2011]; Reddy et al. [2015]; Bârsan et al. [2018]). Wang et al. [2007] developed a theory for performing SLAM with Moving Objects Tracking (SLAMMOT). 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. [2011] 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 containing the static structure and the trajectories of moving objects. Later in Reddy et al. [2015], the authors propose to integrate semantic constraints to further improve the 3D reconstruction. The more recent work Bârsan et al. [2018] present a stereo-based dense mapping algorithm in a SLAM framework, with the advantage of accurately and efficiently reconstructing both static

background and moving objects in large scale dynamic environments. The listed algorithms above have proven that combining multiple objects tracking with SLAM is doable and applicable for dynamic scene exploration. To take a step further by proper exploiting and establishing the spatial and temporal relationships between the robot, static background, stationary and dynamic objects, we show in this thesis that the problems of SLAM and multi-object tracking are mutually beneficial.

The third and most active category is object SLAM, which usually includes both static and dynamic objects. Algorithms in this class normally require specific modelling and representation of 3D object, such as point sets (Dharmasiri et al. [2016]; Henein et al. [2020]), 3D shape (Salas-Moreno et al. [2013]; Tateno et al. [2016]; Sucar et al. [2020]), surfel Runz et al. [2018] or volumetric Xu et al. [2019] model, geometric model such as ellipsoid (Hosseinzadeh et al. [2019]; Nicholson et al. [2018b]) or 3D bounding box (Yang and Scherer [2019]; Li et al. [2018, 2020]; Bescos et al. [2020]), etc., to extract high-level primitive (e.g., object pose) and integrate into a SLAM framework. Salas-Moreno et al. [2013] is one of the earliest works to introduce an object-oriented SLAM paradigm, which represents cluttered scene in object level and constructs an explicit graph between camera and object poses to achieve joint pose-graph optimisation. Later, Tateno et al. [2016] propose a novel 3D object recognition algorithm to ensure the system robustness and improve the accuracy of estimated object pose. The high-level scene representation enables real-time 3D recognition and significant compression of map storage for SLAM. Nevertheless, a database of pre-scanned or pre-trained object models has to be created in advance. To avoid prebuilt database, representing objects using surfel or voxel element in a dense manner starts to gain popularity, along with RGB-D cameras becoming widely used. Runz et al. [2018] present MaskFusion that adopts surfel representation to model, track and reconstruct objects in the scene, while Xu et al. [2019] apply an octree-based volumetric model to objects and build multi-object dynamic SLAM system. Both methods succeed to exploit object information in a dense RGB-D SLAM framework, without prior knowledge of object model. Their main interest, however, is the 3D object segmentation and consistent fusion of the dense map rather than the estimation of the motion of the objects.

Lately, the use of basic geometric models to represent objects becomes a popular solution due to the less complexity and easy integration into a SLAM framework. In Quadric-SLAM Nicholson et al. [2018b], detected objects are represented as ellipsoids to compactly parametrise the size and 3D pose of an object. In this way, the quadric parameters are directly constrained as geometric error and formulated together with camera poses in a factor graph SLAM for joint estimation. Yang and Scherer [2019] propose to combine 2D and 3D object detection with SLAM for both static and dynamic environments. Objects are represented as high-quality cuboids and optimized together with points and cameras through multi-view bundle adjustment. While both methods prove the mutual benefit between detected object and SLAM, their main focus is on object detection and SLAM primarily for static scenarios. In this thesis, we take this direction further to tackle the challenging problem of dynamic object tracking within a SLAM framework, and exploit the relationships between moving objects and agent robot, static and dynamic structures for potential advantages.

Apart from the dynamic SLAM categories, the literature of Multi-motion Visual Odometry (MVO) is also crucial for dynamic VO/SLAM problems. Quite a few methods have been proposed in the literature to estimate $\mathrm{SE}(3)$ motion of objects in a visual odometry or

SLAM framework (Dewan et al. [2016]; Judd et al. [2018]; Huang et al. [2020]). Dewan et al. [2016] present a model-free method for detecting and tracking moving objects in 3D LiDAR scans. The method sequentially estimates motion models using RANSAC Fischler and Bolles [1981a], then segments and tracks multiple objects based on the models by a proposed Bayesian approach. In Judd et al. [2018], the authors address the problem of simultaneous estimation of ego and third-party $\mathrm{SE}(3)$ motions in complex dynamic scenes using cameras. They apply multi-model fitting techniques into a visual odometry pipeline and estimate all rigid motions within a scene. In later work, Huang et al. [2020] present ClusterVO that is able to perform online processing for multiple motion estimations. To achieve this, a multi-level probabilistic association mechanism is proposed to efficiently track features and detections, then a heterogeneous Conditional Random Field (CRF) clustering approach is applied to jointly infer cluster segmentations, with a sliding-window optimization for clusters in the end. While the above proposed methods represent an important step forward to the MVO task, the study of spacial and temporal relationships is not fully explored but is arguably important. Therefore, by carefully considering the pros and cons in the literature of SLAM+MOT, object SLAM and MVO, this thesis proposes a visual dynamic object-aware SLAM system that is able to achieve robust ego and object motion tracking, as well as consistent static and dynamic mapping in a novel SLAM formulation.

Chapter 3

Preliminaries

In this chapter, the preliminaries related to VO/SLAM problems are described. A brief introduction of camera modelling and calibration is first presented. Then the formulation of VO and SLAM problems are discussed, respectively, following the details of optimization-based solutions towards the two problems. After that, Lie groups that are used in the optimization methods for VO/SLAM are covered. Finally, a number of commonly-used optimization tools for VO/SLAM are introduced.

3.1 Camera Modelling and Calibration

A *camera model* is a function that maps the 3D world onto a 2D image plane and is designed to closely model a real-world physical camera. There are many camera models of varying complexity, such as perspective camera, omnidirectional camera and spherical models. This thesis only focuses on the basic and most common model: the perspective camera model.

3.1.1 Perspective Projection

Perspective is the property that objects that are far away appear smaller than closer objects, which is the case with human vision and most real world cameras. The most used model for perspective camera is the pinhole projection system, which assumes that the image is formed by the intersecting light rays from the objects passing through the centre of lens with the focal plane. Let $\mathbf{m} = [m_x, m_y, m_z, 1]^\top \in \mathbb{E}^3$, be the homogeneous coordinates of a 3D point in the camera reference frame, and $\mathbf{p} = [u, v, 1]^\top \in \mathbb{E}^2$ its projection onto the image plane in homogeneous coordinates. The perspective projection equation that connects the 3D world and the 2D image can be written as:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\tilde{\mathbf{m}} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}, \tag{3.1}$$

where $\tilde{\mathbf{m}} \in \mathbb{R}^3$ contains only the 3D point coordinate; λ is a depth factor, $\mathbf{K}$ is the intrinsic calibration matrix and contains the intrinsic parameters: f_u and f_v are the focal lengths, and c_u and c_v are the image coordinates of the projection centre. Note that, when the camera's field

of view is larger than 45 degree, the effects of the radial distortion may become visible and can be modelled using a second or higher order polynomial Hartley and Zisserman [2003]. An illustration of the perspective projection is demonstrated in Figure 3.1.

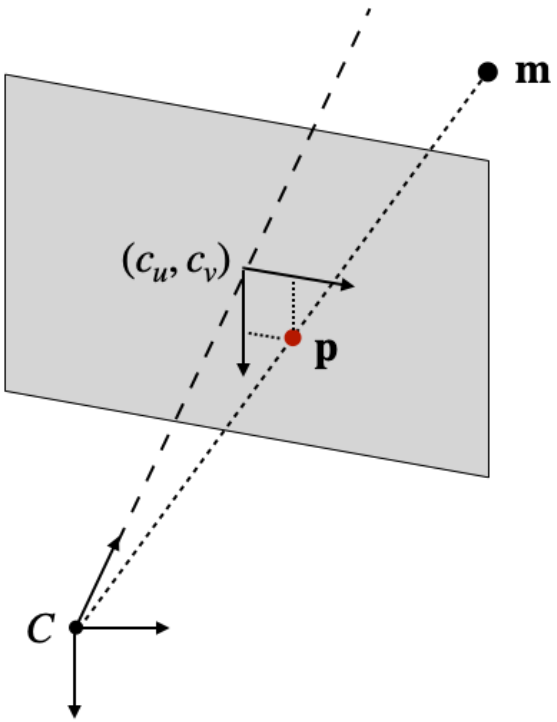


Figure 3.1: **Illustration of the perspective projection camera model.** Light rays from point **m** intersect the image plane at point **p** when passing through the center of the projection.

3.1.2 Camera Calibration

The goal of calibration is to accurately measure the intrinsic and/or extrinsic parameters of the camera system. In a multi-camera system (*e.g.*, stereo and trinocular), the extrinsic parameters describe the mutual position and orientation between each camera pair. Precise camera calibration is very important for 3D interpretation of the image, reconstruction of physical world and the robot's interaction with the reality world.

The most popular method is via using a planar checkerboard-like pattern. To compute the calibration parameters accurately, the user is required to take several pictures of the board shown from different positions and orientations, to ensure that the field of view of the camera is filled as much as possible. The intrinsic and/or extrinsic parameters are then estimated through a least-square minimization method, where the input data are the 3D positions of the corners of the squares of the board and their corresponding pixel coordinates in each image.

Many open source toolboxes are available for estimating the camera parameters, such as the MATLAB camera calibration toolbox Bouguet [1996] and the C/C++ OpenCV calibration library Bradski [2000].

3.2 Formulation of Visual Odometry Problem

As discussed in Chapter 2, there are three classes of VO methods according to different formulation: feature based, direct and 3D point set registration based methods. The proposed algorithms in this thesis are all within the category of feature based, and more specifically, point feature based method. In the following, therefore, only the formulation of this category is covered.

Motion estimation is the core processing step performed for every image in a VO system. Depending on the dimension of the feature correspondences used to estimate motion, there are three different formulations: 2D to 2D, 3D to 3D and 3D to 2D.

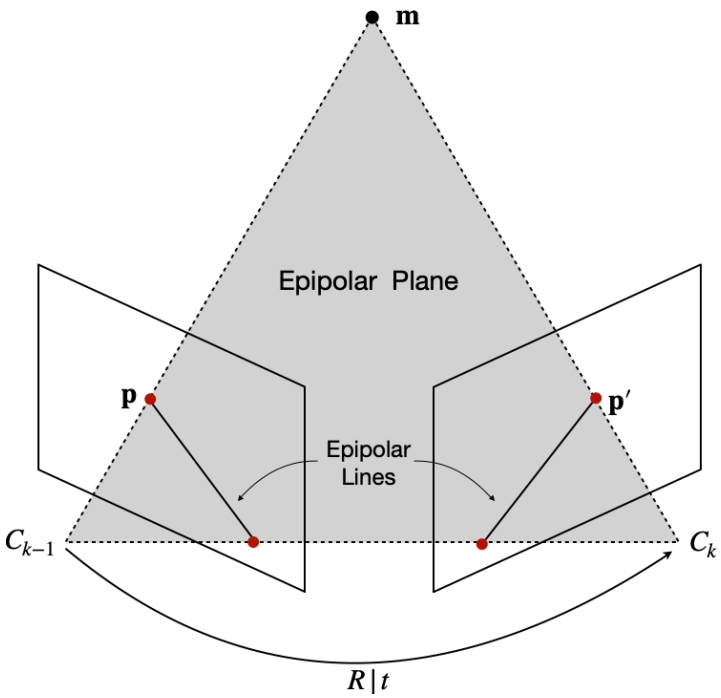


Figure 3.2: **An illustration of the Epipolar constraint.** The two image frames are indicated by their centres C_{k-1} and C_k and the image planes. The two centres, the 3D point **m** and its projections on both images lie in a common plane (Epipolar plane). The Epipolar plane intersects each image plane where it forms lines (Epipolar line). The projection of **m** on image plane k must be contained in the Epipolar line.

3.2.1 2D to 2D: Estimation from 2D Feature Correspondences

The geometric relations between two frames of a calibrated camera are described by *Epipolar* geometry, as illustrated in Figure 3.2. The Epipolar constraint determines the Epipolar lines between the 2D point **p** in image frame k-1 and its correspondence $\mathbf{p}' \in \mathbb{E}^3$ in frame k. This constraint can be formulated as:

$$\tilde{\mathbf{p}}'^{\top} \mathbf{E} \tilde{\mathbf{p}} = 0\,, \tag{3.2}$$

where $\tilde{\mathbf{p}} \in \mathbb{R}^2$ and $\tilde{\mathbf{p}}' \in \mathbb{R}^2$ contain only the 2D coordinates; $\mathbf{E}$ is the *essential matrix*, which contains the camera motion parameters up to an unknown scale factor for the translation and is the following form:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}\,. \tag{3.3}$$

Here $\mathbf{R} \in \mathrm{SO}(3)$ is the rotation matrix, $\mathbf{t} = [t_x, t_y, t_z]^\top \in \mathbb{R}^3$ is translation vector and $[\mathbf{t}]_\times$ is the *skew symmetric matrix* that is given by:

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & t_y \\ -t_y & t_x & 0 \end{bmatrix}. \tag{3.4}$$

The essential matrix can be computed from 2D feature correspondences, and rotation and translation are directly extracted from $\mathbf{E}$. Detailed descriptions of different ways to solve the essential matrix using the Epipolar constraint can be found in Longuet-Higgins [1981]; Zhang [1998]; Nistér [2004].

3.2.2 3D to 3D: Estimation from 3D Point Correspondences

In this approach, the camera motion $\mathbf{T} \in \mathrm{SE}(3)$ from frame k-1 to frame k is estimated via determining the aligning transformation of two 3D point sets, which are normally obtained from stereo or RGB-D sensors. The general solution of finding the transformation is by minimizing the L_2 distance

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i^n |\mathbf{m}'_i - \mathbf{T}\mathbf{m}_i| \tag{3.5}$$

for all n 3D point correspondences. Here $\mathbf{m}_i \in \mathbb{E}^3$ is a 3D point observed from camera frame k-1, and $\mathbf{m}'_i \in \mathbb{E}^3$ from frame k. According to Goldstein [1980], the minimal number of points required to solve the above problem is n=3, which can be used for robust estimation in the presence of outliers. For the case of n>3, one possible solution is to calculate the translation as the difference of the centroids of the 3D point sets, and the rotation using *singular value decomposition* (SVD). More details can be referred to Arun et al. [1987].

3.2.3 3D to 2D: Estimation from 3D Point and 2D Feature Correspondences

Analogous to the 3D-to-3D approach, the camera motion $\mathbf{T}$ between frame k-1 and k is found by formulating a non-linear least squares problem using 3D-to-2D correspondences, and solving the problem by minimizing the image re-projection errors:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i^n |\mathbf{p}'_i - \pi(\mathbf{T}\mathbf{m}_i)|\,, \tag{3.6}$$

where $\pi(\cdot)$ is the projection function. The above problem is also known as a *perspective from n points* (PnP) problem, and many different solutions have been proposed in the literature Moreno-Noguer et al. [2007]; Lepetit et al. [2009]; Kneip et al. [2011]. As described in Fischler and Bolles [1981b], the minimum case involves three 3D-to-2D correspondences, which is called *perspective from three points* (P3P). There are four solutions returned by P3P, and they can be disambiguated with one or more additional corresponding points. In the 3D-to-2D approach, P3P is the standard method for robust motion estimation in the presence of outliers Fischler and Bolles [1981b].

3.3 Formulation of SLAM Problem

Due to the inherent noise in the sensor measurements, the problem of SLAM is normally described by means of probabilistic formulation. Assume a robot moves in an unknown environment, generating a trajectory denoted as a set of random variables $\mathbf{X} = \{\mathbf{x}_k \mid k \in \mathscr{T}\}$, with $\mathscr{T} \subset \mathbb{N}^+$ the cardinality of the set. While moving, the robot acquires a set of odometry measurements $\mathbf{O} = \{\mathbf{o}_k \mid k \in \mathscr{T}\}$ and perceptions of the environment $\mathbf{Z} = \{\mathbf{z}_j \mid j \in \mathscr{N}\}$. Solving the full SLAM problem consists of estimating the posterior probability of the robot's trajectory $\mathbf{X}$ and the map $\mathbf{M} = \{\mathbf{m}_i \mid i \in \mathscr{M}\}$ of the environment, given all the measurements and an initial position $\mathbf{x}_0$:

$$p(\mathbf{X}, \mathbf{M} \,|\, \mathbf{O}, \mathbf{Z}, \mathbf{x}_0)\,. \tag{3.7}$$

The initial position $\mathbf{x}_0$ defines the position of the map and can be chosen arbitrarily. The pose $\mathbf{x}_k$ and the odometry $\mathbf{o}_k$ are normally represented as 2D or 3D transformations in $\mathrm{SE}(2)$ or $\mathrm{SE}(3)$, while the map can be represented in different ways, such as spatially located landmarks, occupancy grids, surface maps or raw sensor measurements.

3.3.1 SLAM as Dynamic Bayesian Network

Estimating the posterior given in (3.7) involves operation in high dimensional state space, which would not be solvable if the SLAM problem does not have a well-defined structure. This structure arises from commonly established assumptions, namely the static world assumption and the Markov assumption.

One intuitive way to describe the structure is through the *Dynamic Bayesian Network* (DBN). A Bayesian network is a graphical model that describes a stochastic process as a directed graph illustrated in Figure 3.3. The graph has one node for each random variable in the process, and a directed edge between two nodes models a conditional dependence between them. Based on the directed graph, the joint probability model corresponding to this network is given as:

$$p(\mathbf{X}, \mathbf{M}, \mathbf{O}, \mathbf{Z}, \mathbf{x}_0) = p(\mathbf{x}_0) \prod_{k \in \mathscr{T}} p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}, \mathbf{o}_k) \prod_{j \in \mathscr{N}} p(\mathbf{z}_j \,|\, \mathbf{x}_k, \mathbf{m}_i)\,, \tag{3.8}$$

where $p(\mathbf{x}_0)$ is a prior on the initial state, $p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}, \mathbf{o}_k)$ is the state transition model and $p(\mathbf{z}_j \,|\, \mathbf{x}_k, \mathbf{m}_i)$ is the observation model.

The transition model $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{o}_k)$ is indicated as the two edges leading to $\mathbf{x}_k$ and represents the probability that the robot at time k is in $\mathbf{x}_k$ given that at time $k-1$ it was in $\mathbf{x}_{k-1}$ and it acquired an odometry measurement $\mathbf{o}_k$. The observation model $p(\mathbf{z}_j \mid \mathbf{x}_k, \mathbf{m}_i)$ models the probability of performing the observation $\mathbf{z}_j$ given that the robot is at location $\mathbf{x}_k$ in the map. It is represented by the arrows entering in $\mathbf{z}_j$. The exteroceptive observation $\mathbf{z}_j$ depends only on the current location $\mathbf{x}_k$ of the robot and on the map that is observed at time k ($\mathbf{m}_i$ in Figure 3.3). Here it assumes that the data-association problem has been solved, *i.e.*, the association between the map $\mathbf{M}$ and the observation measurements $\mathbf{Z}$ is known.

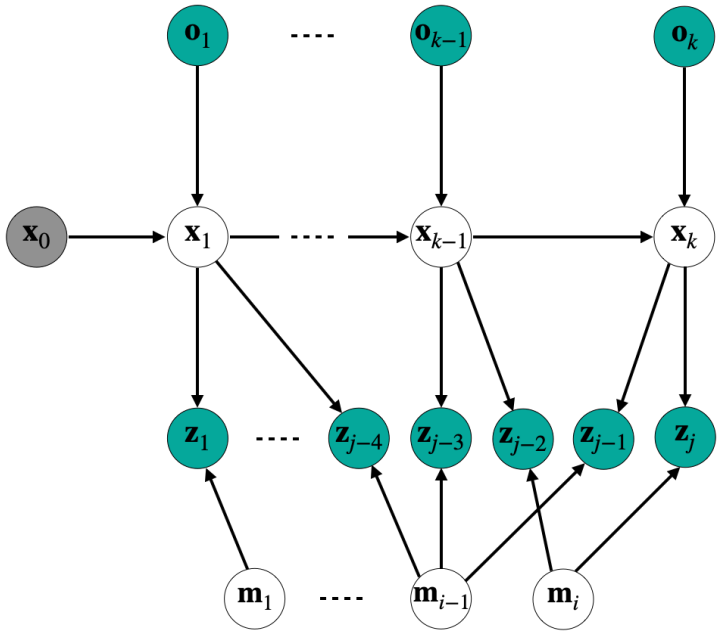


Figure 3.3: **Dynamic Bayesian Network formulation of the SLAM problem.** The grey node indicates an initial prior. Green nodes refer to the observed variables and white nodes the hidden variables.

As is standard in the SLAM literature Smith et al. [1990]; Leonard et al. [1992]; Dissanayake et al. [2001], Gaussian transition and observation models Maybeck [1982] is assumed and defined by

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \mathbf{o}_k) + w_k \;\Leftrightarrow\; p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{o}_k) \propto \exp -\frac{1}{2} ||f_k(\mathbf{x}_{k-1}, \mathbf{o}_k) - \mathbf{x}_k||^2_{\Sigma_k}, \tag{3.9}$$

where $f_k(\cdot)$ is a transition measurement model, and w_k is normally distributed zero-mean process noise with covariance matrix Σ_k, and

$$\mathbf{z}_j = h_j(\mathbf{x}_k, \mathbf{m}_i) + v_j \;\Leftrightarrow\; p(\mathbf{z}_j \mid \mathbf{x}_k, \mathbf{m}_i) \propto \exp -\frac{1}{2} ||h_j(\mathbf{x}_k, \mathbf{m}_i) - \mathbf{z}_j||^2_{\Sigma_j}, \tag{3.10}$$

where $h_j(\cdot)$ is a observation measurement model, and v_j is normally distributed zero-mean measurement noise with covariance Σ_j. Here $||\mathbf{e}||^2_\Sigma = \mathbf{e}^\top \Sigma^{-1} \mathbf{e}$ is defined as the squared Mahalanobis distance given a covariance matrix Σ. The equations above model the robot's behaviour

in response to control input, and its sensors, respectively.

3.3.2 SLAM as Factor Graph

An alternative representation to the DBN is via the so-called 'graph-based' formulation of the SLAM problem, that highlights the underlying spatial structure. A graph-based SLAM approach constructs a simplified estimation problem by abstracting the sensor measurements. These measurements are replaced by the edges in the graph which can then be considered as parameters of the joint probability factors over the actual variables. This naturally leads to the well known *factor graph* representation, a class of bipartite graphical models that can be used to represent such factored densities Kschischang et al. [2001].

In a factor graph there are nodes for variables, edges for the probability factors defined on them, and the graph structure expresses which variables are involved in each factor. The factor graph for the example from Figure 3.3 is shown in Figure 3.4. As can be seen, there are factor edges for both odometry measurements $\mathbf{O}$ and observation measurements $\mathbf{Z}$.

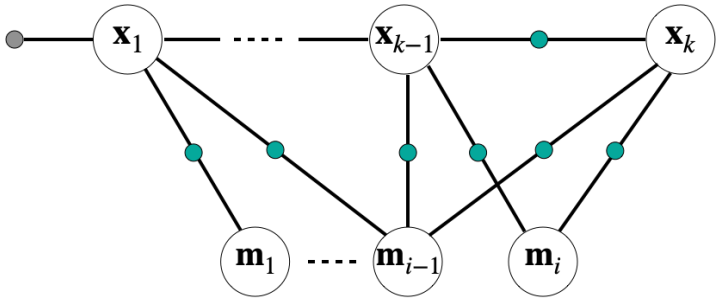


Figure 3.4: **Factor graph formulation of the SLAM problem.** The white nodes represent the actual variables. The edge with grey dot indicates an initial state, and the edges with green dots refer to the observed measurements.

For a typical graph-based SLAM problem, only single and pairwise cliques (constraints) are considered for succinct explanation. Given a set of variables Θ, which includes both the robot's poses and the map, the factor graph expression of the joint density over Θ can be considered as:

$$p(\Theta) = \prod_i \phi_i(\theta_i) \prod_{\{i,j\},i<j} \psi_{ij}(\theta_i, \theta_j)\,, \tag{3.11}$$

Normally the potentials $\phi_i(\theta_i)$ encode a prior or a single measurement constraint at an variable $\theta_i \in \Theta$, whereas the pairwise potentials $\psi_{ij}(\theta_i, \theta_j)$ refer to measurements or constraints that involve the relationships between the variables θ_i and θ_i. Note that the second product is over pairwise cliques $\{i, j\}$ and is counted only once. The equivalence between equations (3.8) and (3.11) can be readily established by taking

$$\phi_0(\mathbf{x}_0) \propto p(\mathbf{x}_0)\,, \tag{3.12}$$

$$\psi_{(k-1)k}(\mathbf{x}_{k-1}, \mathbf{x}_k) \propto p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}, \mathbf{o}_k)\,, \tag{3.13}$$

$$\psi_{ki}(\mathbf{x}_k, \mathbf{m}_i) \propto p(\mathbf{z}_j \,|\, \mathbf{x}_k, \mathbf{m}_i)\,. \tag{3.14}$$

Based on the factor graph modelling, an optimal estimate for the set of variables given all the measurements can be obtained. Specifically, given the observation measurements $\mathbf{Z}$ and odometry measurements $\mathbf{O}$, the goal is to recover the *Maximum a Posteriori* (MAP) estimate for the entire trajectory $\mathbf{X}$ and the map $\mathbf{M}$. Let $\Theta := (\mathbf{X}, \mathbf{M})$ be the vector collecting all the variables in $\mathbf{X}$ and $\mathbf{M}$, the MAP estimate is obtained via maximizing the joint probability $p(\mathbf{X}, \mathbf{M}, \mathbf{O}, \mathbf{Z}, \mathbf{x}_0)$ from (3.8),

$$\begin{aligned}\Theta^* = \underset{\Theta}{\operatorname{argmax}}\ p(\mathbf{X}, \mathbf{M} \,|\, \mathbf{O}, \mathbf{Z}, \mathbf{x}_0) &= \underset{\Theta}{\operatorname{argmax}}\ p(\mathbf{X}, \mathbf{M}, \mathbf{O}, \mathbf{Z}, \mathbf{x}_0) \\ &= \underset{\Theta}{\operatorname{argmin}} - \log p(\mathbf{X}, \mathbf{M}, \mathbf{O}, \mathbf{Z}, \mathbf{x}_0)\,, \end{aligned} \tag{3.15}$$

which comes to the following non-linear least-squares problem combining (3.9) and (3.10),

$$\Theta^* := \underset{\Theta}{\operatorname{argmin}} \Big\{ \sum_{k \in \mathscr{T}} ||f_k(\mathbf{x}_{k-1}, \mathbf{o}_k) - \mathbf{x}_k||_{\Sigma_k} + \sum_{j \in \mathscr{N}} ||h_j(\mathbf{x}_k, \mathbf{m}_i) - \mathbf{z}_j||_{\Sigma_j} \Big\} \tag{3.16}$$

Regarding the prior $p(\mathbf{x}_0)$, it is assumed that $\mathbf{x}_0$ is given and hence is treated as a constant constraint. This is what is often done in practice: the origin of the coordinate system is arbitrary, and thereby $\mathbf{x}_0$ can be fixed at the origin as well.

3.4 Optimization for VO/SLAM

As discussed in the previous two sections, both VO and SLAM are normally formulated as non-linear least-squares problems and solved through optimization. In this section, basic preliminaries regarding to optimization is introduced, along with classic solutions that are applied to VO/SLAM problems.

3.4.1 Taylor Series

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

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{3.17}$$

is called the *Taylor series* of f at a. Here $f^{(n)}(a)$ denotes the n-th derivative of f evaluated at the point a, and the derivative of order zero of f is defined to be f itself.

If $f(x)$ is given by a convergent power series in an open disc centred at b in the complex plane, it is said to be *analytic* in this disc. Thus for x in this disc, f is given by a convergent power series:

$$f(x) = \sum_{n=0}^{\infty} a_n (x-b)^n\,. \tag{3.18}$$

Differentiating by x the above formula n times, then setting $x = b$ gives:

$$\frac{f^{(n)}(b)}{n!} = a_n, \tag{3.19}$$

and hence the power series expansion agrees with the Taylor series. Thus a function is analytic in an open disc centred at b if and only if its Taylor series converges to the value of the function at each point of the disc.

The usage of the Taylor series for analytic function is important. For instance, the partial Taylor polynomials of the series can be used as approximations of the function. These approximations are good if sufficiently many terms are included. Approximations using the first few terms can also make otherwise unsolvable problems possible for a restricted domain.

3.4.2 Multivariate Differentiation

Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ be a function such that each of its first-order partial derivatives exists on $\mathbb{R}^n$. This function takes a point $\mathbf{x} \in \mathbb{R}^n$ as input and produces the vector $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^n$ as output. Then the *Jacobian matrix* of $\mathbf{f}$ is defined to be an $n \times m$ matrix, denoted by $\mathbf{J_f}$ (or $\nabla\mathbf{f}$) as:

$$\mathbf{J_f} := \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}. \tag{3.20}$$

If all second partial derivatives of $\mathbf{f}$ exist and are continuous over the domain of the function, then the *Hessian matrix* $\mathbf{H_f}$ of $\mathbf{f}$ is a square $n \times m \times n$ matrix, which is the Jacobian matrix of the gradient of the $\mathbf{f}$ function: $\mathbf{H}(\mathbf{f}(\mathbf{x})) = \mathbf{J}(\nabla\mathbf{f}(\mathbf{x}))$.

3.4.3 Optimization Problem and Typical Solutions

An optimization problem is the problem of finding the best solution from all feasible solutions, which can be represented in the following way:

- *Given:* a function $F : \mathbb{R}^n \to \mathbb{R}$ mapping from a vector onto a scalar field,
- *Sought:* an element $\mathbf{x}^* \in \mathbb{R}^n$ such that $F(\mathbf{x}^*) \leq F(\mathbf{x})$ for all possible $\mathbf{x} \in \mathbb{R}^n$,

or in a simpler form as:

$$\mathbf{x}^* = \min_{\mathbf{x} \in \mathbb{R}^n} F(\mathbf{x}). \tag{3.21}$$

The function F is normally called an *objective* function or *cost* function, and a feasible solution that minimizes the objective function is called an *optimal* solution.

For a general scalar field $F : \mathbb{R}^n \to \mathbb{R}$, there is no guarantee to find such a global minimum in a finite number of steps, even if it is assumed to be infinitely differentiable. Therefore, most proposed methods often focus on finding a local minima as actual solutions to the original problem. A local minimum $\mathbf{x}^*$ is defined as an element for which there exists some $\delta > 0$ such that

$$\forall \mathbf{x} \in \mathbb{R}^n \text{ where } ||\mathbf{x} - \mathbf{x}^*|| \leq \delta, \tag{3.22}$$

the expression $F(\mathbf{x}^*) \leq F(\mathbf{x})$ holds; that is to say, on some region around $\mathbf{x}^*$ all of the function values are greater than or equal to the value at that element.

3.4.3.1 Gradient Descent Method

The simplest way to find the local minimum of F is via *gradient descent* algorithm Nocedal and Wright [2006]. The method is based on the observation that if $F(\mathbf{x})$ is defined and differentiable in a neighbourhood of a point $\mathbf{x}$, then $F(\mathbf{x})$ decreases fastest if one goes from $\mathbf{x}$ in the direction of the negative gradient of F at $\mathbf{x}$, *i.e.*, $-\nabla F(\mathbf{x})$. It follows that, if

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \gamma_k \nabla F(\mathbf{x}^{(k)}),\ k \geq 0 \tag{3.23}$$

for $\gamma_k \in \mathbb{R}^+$ small enough, then $F(\mathbf{x}^{(k+1)}) \leq F(\mathbf{x}^{(k)})$. With this observation in mind, one starts with a guess $\mathbf{x}^{(0)}$ for a local minimum of F, and considers the sequence $\mathbf{x}^{(0)}$, $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$,..., such that

$$F(\mathbf{x}^{(0)}) \geq F(\mathbf{x}^{(1)}) \geq F(\mathbf{x}^{(2)}) \geq \ldots, \tag{3.24}$$

and if no such γ_k exists to satisfy the above condition, a desired local minimum is reached. The gradient descent approach 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.

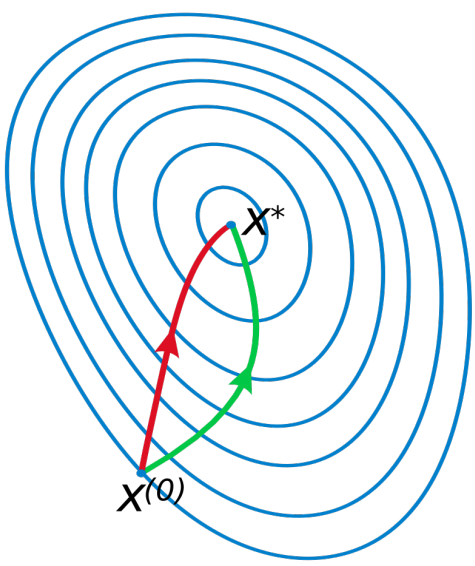


Figure 3.5: **A comparison of Newton's method (red) and gradient descent method (green) for minimizing a function with small step sizes.** Newton's method uses curvature information (i.e. the second derivative) to take a more direct route.

3.4.3.2 Newton's Method

A more efficient approach is *Newton's method*, which attempts to solve this problem by constructing a sequence $\{\mathbf{x}^{(k)}\}$ from an initial guess $\mathbf{x}^{(0)}$ that converges towards a minimum $\mathbf{x}^*$ of F by using a sequence of second-order Taylor approximations of F around the iteration point.

The second-order Taylor expansion of F around $\mathbf{x}^{(k)}$ is

$$F(\mathbf{x}^{(k)}+\Delta\mathbf{x}) \approx F(\mathbf{x}^{(k)}) + \nabla F(\mathbf{x}^{(k)})\Delta\mathbf{x} + \frac{1}{2}\mathbf{H}_F(\mathbf{x}^{(k)})\Delta\mathbf{x}^2 . \tag{3.25}$$

The next iterate $\mathbf{x}^{(k+1)}$ is defined so as to minimize this quadratic approximation in $\Delta\mathbf{x}$, with $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}$. If the second derivative is positive, the quadratic approximation is a convex function of $\Delta\mathbf{x}$, and its minimum can be found by setting the derivative to zero.
Since

$$0 = \frac{\mathrm{d}}{\mathrm{d}\Delta\mathbf{x}}\left(F(\mathbf{x}^{(k)}) + \nabla F(\mathbf{x}^{(k)})\Delta\mathbf{x} + \frac{1}{2}\mathbf{H}_F(\mathbf{x}^{(k)})\Delta\mathbf{x}^2\right) = \nabla F(\mathbf{x}^{(k)}) + \mathbf{H}_F(\mathbf{x}^{(k)})\Delta\mathbf{x}, \tag{3.26}$$

the minimum is achieved for

$$\Delta\mathbf{x} = -\frac{\nabla F(\mathbf{x}^{(k)})}{\mathbf{H}_F(\mathbf{x}^{(k)})} . \tag{3.27}$$

Putting everything together, Newton's method performs the iteration

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x} = \mathbf{x}^{(k)} - \frac{\nabla F(\mathbf{x}^{(k)})}{\mathbf{H}_F(\mathbf{x}^{(k)})} . \tag{3.28}$$

In contrast to the gradient descent method, Newton's method converges especially fast in the neighbourhood of the minimum. For high-dimensional problems, however, computing the Hessian $\mathbf{H}_F(\mathbf{x})$ is an expensive operation.

3.4.3.3 Gauss-Newton Method

The *Gauss-Newton* algorithm is used to solve non-linear least squares problems. Unlike Newton's method, the Gauss-Newton algorithm can only be used to minimize a sum of squared function values, but it has the advantage that second derivatives, which can be challenging to compute, are not required.
Assume $\mathbf{F}$ to be of the following form:

$$\mathbf{F}(\mathbf{x}) = \mathbf{r}(\mathbf{x})^\top \Lambda \mathbf{r}(\mathbf{x}) , \tag{3.29}$$

with $\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m, n \geq m$ (often called residuals) a twice differentiable vector field, and $\Lambda \in \mathbb{R}^{m\times m}$ being a symmetric, positive semi-definite matrix.
The first derivative of $\mathbf{F}$ is given by,

$$\nabla\mathbf{F}(\mathbf{x}) = \left(\mathbf{r}(\mathbf{x})^\top \Lambda \mathbf{J}_\mathbf{r}(\mathbf{x})\right)^\top + \mathbf{J}_\mathbf{r}(\mathbf{x})^\top \Lambda \mathbf{r}(\mathbf{x}) = 2\,\mathbf{J}_\mathbf{r}(\mathbf{x})^\top \Lambda \mathbf{r}(\mathbf{x}) , \tag{3.30}$$

and the second derivative of $\mathbf{F}$ is obtained as,

$$\mathbf{H}_\mathbf{F}(\mathbf{x}) = \mathbf{J}_\mathbf{r}(\mathbf{x})^\top \Lambda \mathbf{J}_\mathbf{r}(\mathbf{x}) + \mathbf{H}_\mathbf{r} \Lambda \mathbf{r}(\mathbf{x}) , \tag{3.31}$$

with $\mathbf{H_r}$ being the Hessian tensor of $\mathbf{r}$.
The Gauss-Newton method approximates the Hessian of F as

$$\mathbf{H_F}(\mathbf{x}) \approx \mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{J_r}(\mathbf{x}), \tag{3.32}$$

which ignores the second-order derivative terms.
The above $\nabla\mathbf{F}(\mathbf{x})$ and $\mathbf{H_F}(\mathbf{x})$ are substituted into the recurrence relation (3.28) to obtain the follow iterative equation:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x} = \mathbf{x}^{(k)} - \left(\mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{J_r}(\mathbf{x})\right)^{-1} \mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{r}(\mathbf{x}). \tag{3.33}$$

Convergence of the Gauss-Newton method is not guaranteed in all instances. The approximation of (3.32) may be valid in two cases, for which convergence is to be expected Wright and Nocedal [1999]:

1. The residues in $\mathbf{r}$ are small in magnitude, at least around the minimum;

2. The functions of $\mathbf{r}$ are only "mildly" nonlinear, so that $\mathbf{H_r}$ is relatively small in magnitude.

3.4.3.4 Levenberg-Marquardt Method

As discussed above, the gradient descent method guarantees local convergence but suffers poor performance close to the minimum. On the contrary, the Gauss-Newton method works particularly well close to the minimum, but the cost may not decrease for convergence as the quadratic approximation may not always be a good approximation if the start point is far off the final minimum.
An alternative method to combine the advantages of both is the *Levenberg* algorithm which interpolates gradient descent and Gauss-Newton by altering the normal equations as follows:

$$\left(\mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{J_r}(\mathbf{x}) + \lambda \mathbf{I}\right) \Delta\mathbf{x} = -\mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{r}(\mathbf{x}), \tag{3.34}$$

where λ is a damping ratio and $\mathbf{I}$ is the identity matrix.
The parameter $\lambda > 0$ steers the update vector $\Delta\mathbf{x}$ towards the direction of the steepest descent. As λ approaches zero, Levenberg method becomes the standard Gauss-Newton method. On the other hand, if λ approaches infinity, the matrix $\left(\mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{J_r}(\mathbf{x}) + \lambda \mathbf{I}\right)$ approaches a diagonal matrix with infinite trace. Thus, as $\lambda \to \text{inf}$,

$$\Delta\mathbf{x} = -\frac{1}{\lambda} \mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{r}(\mathbf{x}), \tag{3.35}$$

and the Levenberg method becomes a gradient descent update.
The damping ratio is adjusted in each optimisation iteration. Only if the update $\mathbf{x}^{(k)} + \Delta\mathbf{x}$ reduces the cost ($F(\mathbf{x}^{(k)} + \Delta\mathbf{x}) \ll F(\mathbf{x}^{(k)})$), the update is accepted, which indicates the algorithm is approaching the local minimum and hence λ is reduced to strengthen the influence of Gauss-Newton. However, if the update does not reduce the cost, it is rejected, and a larger λ (smaller step size) and an update more oriented towards the steepest descent direction is attempted.

A refinement of the Levenberg method is the *Levenberg-Marquardt* Method that changes how the matrix $\left(\mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{J_r}(\mathbf{x}) + \lambda \mathbf{I}\right)$ is defined. Specifically, instead of damping all parameter dimensions equally with $\mathbf{I}$, a scaled version of of the diagonal of the information matrix itself can be added, and this matrix becomes:

$$\mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{J_r}(\mathbf{x}) + \lambda \operatorname{diag}(\mathbf{J_r}(\mathbf{x})^\top \Lambda \mathbf{J_r}(\mathbf{x}))\,. \tag{3.36}$$

As λ grows, $\Delta\mathbf{x}$ also tends to wards a gradient descent update, but with each dimension scaled according to the diagonal of the information matrix. This can lead to faster convergence than the Levenberg damping term when some dimensions of the error surface have much different curvature than others Madsen et al. [2004].

3.5 Lie Groups in VO/SLAM

In the optimisation methods presented in Section 3.4.3, the variables $\mathbf{x}$ to be optimized are normally defined on Euclidean vector spaces $\mathbb{R}^n$. Solving the VO/SLAM problems, however, involves manipulation and estimation of 3D transformation in non-Euclidean spaces. Without a coherent and robust framework for representing and working with 3D transformations, these tasks can be onerous and intractable. Transformations must be composed, inverted, differentiated and interpolated. *Lie groups* and their associated machinery (*Lie algebra*) address all of these operations, and do so in a principled way, so that once intuition is developed, it can be followed with confidence.

This section aims at revisiting the general update rule for optimization in non-Euclidean spaces. For more details about the mathematical definition and derivation of Lie groups and Lie algebra, the reader can refer to Blanco [2010]; Eade [2013].

3.5.1 Special Orthogonal Group SO(3)

The *special orthogonal group* $\mathrm{SO}(3)$ represents the group of 3D rotations by rotation matrix. Composition and inversion in the group correspond to matrix multiplication and inversion. As rotation matrices are orthogonal, inversion is equivalent to transposition.
The associated Lie algebra $\mathrm{so}(3)$ is a set of 3×3 skew-symmetric matrices, which are constructed by three basis *generator* matrices corresponding to the derivatives of rotation around each of the standard axes, evaluated at the identity:

$$G_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \; G_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \; G_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.37}$$

An element of $\mathrm{so}(3)$ is then represented as a linear combination of the generators:

$$\boldsymbol{\omega}_\times = \omega_1 G_1 + \omega_2 G_2 + \omega_3 G_3 = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}, \tag{3.38}$$

where $\boldsymbol{\omega} = [\omega_1\ \omega_2\ \omega_3]^\top$, and $\boldsymbol{\omega}_\times \in \mathrm{so}(3)$ represents the skew symmetric matrix.
The exponential map that takes skew symmetric matrices to rotation matrices is simply the matrix exponential over a linear combination of the generators:

$$\exp : \mathrm{so}(3) \to \mathrm{SO}(3)\,, \tag{3.39}$$

and has the closed form solution using *Rogridues* rotation formula:

$$\exp(\boldsymbol{\omega}_\times) = \mathbf{I}_3 + \frac{\sin\theta}{\theta}\boldsymbol{\omega}_\times + \frac{1-\cos\theta}{\theta^2}\boldsymbol{\omega}_\times^2\,. \tag{3.40}$$

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

$$\ln(\mathbf{R}) = \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^\top)\,. \tag{3.41}$$

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

3.5.2 Special Euclidean Group SE(3)

The *special Euclidean group*, $\mathrm{SE}(3)$, represents the group of rigid transformations in 3D space. An element of rigid transformation $\mathbf{T} \in \mathrm{SE}(3)$ is denoted as,

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tag{3.42}$$

with $\mathbf{R} \in \mathrm{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$.
The Lie algebra $\mathrm{se}(3) \in \mathbb{R}^6$ is the set of 4×4 matrices corresponding to differential translations and rotations ($\mathrm{so}(3)$). There are thus six generators of the algebra:

$$G_1 = \begin{pmatrix} 0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}, G_2 = \begin{pmatrix} 0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0 \end{pmatrix}, G_3 = \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0 \end{pmatrix},$$
$$G_4 = \begin{pmatrix} 0&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&0 \end{pmatrix}, G_5 = \begin{pmatrix} 0&0&1&0\\0&0&0&0\\-1&0&0&0\\0&0&0&0 \end{pmatrix}, G_6 = \begin{pmatrix} 0&-1&0&0\\1&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix}. \tag{3.43}$$

Given $(\mathbf{v}\ \boldsymbol{\omega})^\top \in \mathbb{R}^6$, an element of $\mathrm{se}(3)$ is represented as multiples of the generators:

$$v_1\,G1 + v_2\,G2 + v_3\,G3 + \omega_1\,G_4 + \omega_2\,G_5 + \omega_3\,G_6 = \begin{pmatrix} \boldsymbol{\omega}_\times & \mathbf{v} \\ \mathbf{0}^\top & 0 \end{pmatrix}. \tag{3.44}$$

The exponential map from $\mathrm{se}(3)$ to $\mathrm{SE}(3)$ is the matrix exponential over a linear combination of the generators:

$$\exp : \mathrm{se}(3) \to \mathrm{SE}(3), \tag{3.45}$$

and has the closed form solution as:

$$\exp \begin{pmatrix} \boldsymbol{\omega}_\times & \mathbf{v} \\ \mathbf{0}^\top & 0 \end{pmatrix} = \begin{pmatrix} \exp(\boldsymbol{\omega}_\times) & \mathbf{V}\mathbf{v} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tag{3.46}$$

where

$$V = \mathbf{I}_3 + \frac{1-\cos\theta}{\theta^2}\boldsymbol{\omega}_\times + \frac{\theta - \sin\theta}{\theta^3}\boldsymbol{\omega}_\times . \tag{3.47}$$

The logarithm map of $\mathrm{SE}(3)$ can be operated by first finding $\ln(\mathbf{R})$ as in (3.41), then computing $\mathbf{v} = \mathbf{V}^{-1}\mathbf{t}$.

3.5.3 Optimisation on SE(3)

The $\mathrm{SE}(3)$ in VO/SLAM optimisation problems can be solved naturally now. Generally, the variables in a classical VO/SLAM problem include the camera/robot poses and point positions. A camera pose lies in the non-Euclidean $\mathrm{SE}(3)$ space, therefore, it needs to be parameterised using the Lie algebra explained in the last section. Assume an update as $\Delta\mathbf{x}$ is performed around the current variable $\mathbf{x}$. The variable update formula now changes to a general form:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x} \Longrightarrow \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \boxplus \Delta\mathbf{x}. \tag{3.48}$$

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

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \boxplus \Delta\mathbf{x} \Longrightarrow \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} \cdot \exp(\Delta\mathbf{x}). \tag{3.49}$$

Analogously, the Jacobian of a residue function $\mathbf{r}$ also needs to extend to the general form:

$$\mathbf{J}_\mathbf{r}(\mathbf{x}) = \frac{\partial \mathbf{r}(\mathbf{x} \boxplus \Delta\mathbf{x})}{\partial \Delta\mathbf{x}}|_{\Delta\mathbf{x}=0}. \tag{3.50}$$

3.6 General Optimisation Libraries for VO/SLAM

There are several existing libraries for solving non-linear least squares problems that are commonly defined in VO and SLAM communities. Those include g2o Kümmerle et al. [2011], GTSAM Dellaert [2012], Ceres Agarwal et al. [2012] and SLAM++ Ila et al. [2017], *etc.*

Specifically, g2o (General Graph Optimization) is a C++ framework for performing the optimization of non-linear least squares problems that can be embedded as a graph. The g2o has the advantage of being easily extensible to a wide range of problems, and the current implementation provides solutions to several variants of SLAM and bundle adjustment (BA) problems. GTSAM (Georgia Tech Smoothing and Mapping) is graph-based framework implemented in C++, which uses factor graph to model the non-linear least square problems, and solves the problems using graphical algorithms. Ceres is also an open source C++ library for modelling and solving large, complicated optimization problems, including non-linear Least

Squares problems with bounds constraints and general unconstrained optimization problems. SLAM++ is another general C++ framework for incremental maximum likelihood estimation. This framework aims to maintain a sparse and scalable state representation for large scale mapping, to achieve fast matrix manipulation and arithmetic operations used in non-linear least squares.

In this thesis, optimization problems are mainly solved using the g2o framework, including camera pose refinement and bundle adjustment in Chapter 4, joint optic-flow and 6-DoF motion estimation in Chapter 5, and graph optimization for dynamic SLAM in Chapter 6.

Chapter 4

Robust Visual Odometry in Underwater Environments

The accurate estimation of pose and velocity of an autonomous underwater vehicle (AUV) is critical to ensure the repeatability and validity of scientific data that is captured using sensors onboard the AUV. A low-cost and effective way is by using stereo camera sensors to perform visual odometry (VO). However, this is a difficult problem in underwater due to poor imaging conditions and inconsistent motion caused by water flow. In this chapter, a robust and effective stereo underwater VO system is proposed to overcome aforementioned difficulties and accurately localize the AUV. Experimental results demonstrate that the proposed pipeline outperforms existing VO systems in underwater environments, as well as obtains a comparative performance on the KITTI benchmark dataset Geiger et al. [2012].

4.1 Motivation

In recent years, the demand for exploration in underwater environments using autonomous underwater vehicle is constantly increasing. Positioning sensors such as Doppler Velocity Logs (DVL) or acoustic transponders like long baseline system (LBL) and ultra short baseline system (USBL) are commonly used in localizing AUVs. Although these methods can provide accurate pose estimation, they are very expensive and not easy to integrate or deploy in many cases.

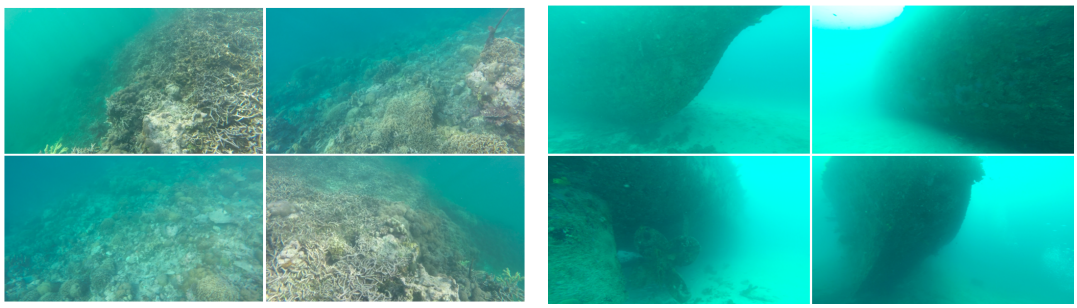
A cost effective alternative is to use visual sensors and perform visual odometry. This technique is becoming very popular in computer vision and robotics Mur-Artal and Tardós [2017]; Forster et al. [2014]; Engel et al. [2014], and provides a low-cost and effective solution to estimate the robot trajectory. Nevertheless, this becomes comparatively challenging in underwater environments due to the following issues: a) As is demonstrated in Figure 4.1, the imaging conditions in water are poor due to light attenuation, poor/artificial illumination, haze and scattering. When the AUV operates in shallow water, scattering of the sun light is highly problematic; b) Motion blur can also be present and is due to motion of the robot while the camera shutter is open; c) The vehicle/camera's motion is inconsistent with oscillation, especially in shallow underwater area, due to the water waves. All these problems greatly increases the difficulty in estimating the robot location.

In this chapter, possible solutions of the mentioned problems are intensively tested and

evaluated. A stereo underwater VO system is proposed, that is able to robustly and accurately localize the AUV. The proposed system is built upon a popular visual-based localization system in robotics called ORB-SLAM2 Mur-Artal and Tardós [2017]. Careful modifications are made for this system to accommodate the above mentioned challenging conditions, and experimental results demonstrate that the proposed pipeline outperforms existing VO systems in underwater environments.

4.2 Datasets

Two underwater datasets will be tested and analysed in this chapter. One of them is called Underwater Coral Dataset, which is a stereo video sequence dataset[1] that is captured manually over the undersea coral reef using a GoPro stereo rig. The video contains 2500 frames of image size 1920×1080. It starts at one marked place, and heads forward in the length of approximately 60 meters to another marked spot, then turns back to its starting point. As is shown in Figure 4.1 (a), this dataset is affected by certain level of hazing, and most of the images are half invisible because of the camera viewing angle. The most challenging part of this dataset is the inconsistent motion in most of the frames caused by the shallow water waves, which not only results in motion blur that affects the feature extraction and matching, but also makes any constant motion model assumption fail.



(a) (b)

Figure 4.1: **Selected typical sample images**. (a) The Underwater Coral Dataset and (b) The Underwater Shipwreck Dataset.

The other dataset is the Underwater Shipwreck Dataset Li et al. [2016] that is also a stereo dataset capturing around an underwater shipwreck by a GoPro stereo rig. This video (1800 frames in total and 1920×1080 in size) starts at the front of the shipwreck, and circles around it for a lap. Although the camera motion in this dataset is relatively smoother comparing to Underwater Coral Dataset, some parts of the image sequence are encountered with severe shaking. The big obstacle in this dataset is the poor imaging condition that is caused by the turbid water and light attenuation, leading to only a limited number of valid features being detected and tracked in every frame. See Figure 4.1 (b).

[1] Collected by Dr. Feras Dayoub and Dr. Matthew Dunbabin from Queensland University of Technology.

4.3 Pipeline

The state-of-the-art ORB-SLAM2 Mur-Artal and Tardós [2017] VO pipeline failed to work with the tested underwater datasets due to the challenging problems discussed in Section 4.1, see Figure 4.5 (b). Therefore, in this work, it is carefully evaluated and modified as follows. Figure 4.2 shows the proposed visual odometry pipeline in comparison with the VO part of ORB-SLAM2. In the feature extraction, three state-of-the-art feature techniques are experimentally compared and analysed to find the optimal solution that is most robust against challenging image condition to conduct underwater visual odometry. Inconsistent motion is handled by two techniques in feature association and motion estimation, respectively: a) Instead of using constant motion model (in frame-to-frame tracking) to match features, an effective loop search is performed between current left, right and previous left, right frames (quad matching) to find the matchings. The proposed quad matching approach is simple, fast and not affected by unpredictable motion. b) Rather than directly optimizing (motion-only bundle adjustment) with motion model as initial value, a straightforward pose estimation from 2D to 3D correspondences is conducted using P3P Kneip et al. [2013] with RANSAC to get an initial pose for each new camera frame.

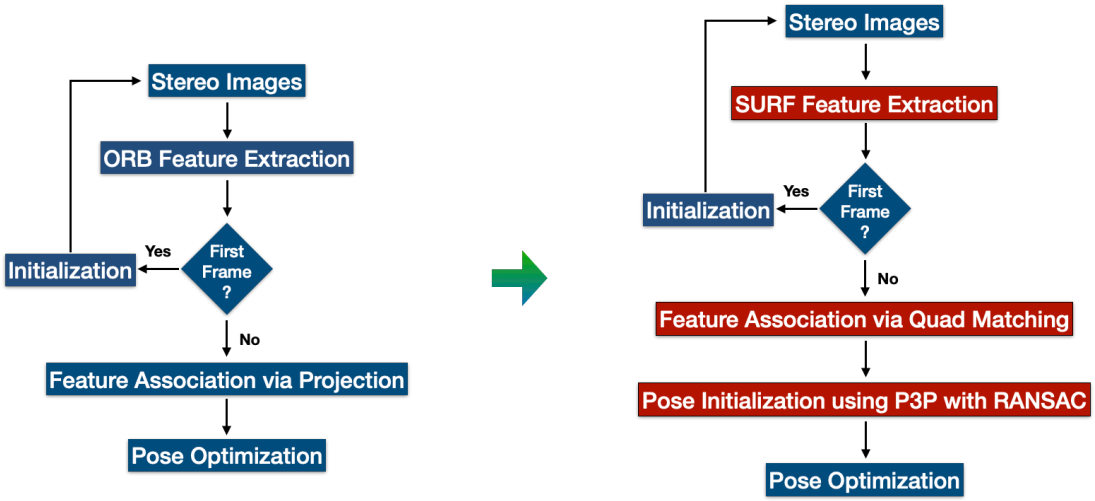


Figure 4.2: **The visual odometry pipeline of ORB-SLAM2 (left) and the proposed (right).** Red rectangular boxes indicate the new added components.

Algorithm. 1 presents the pseudo code of the proposed VO pipeline. The algorithm inputs the stereo images sequence $\mathbf{I}$ and outputs the camera poses $\mathbf{X}$ in the world coordinate system (denoted with upper left index 0). For each incoming stereo frame $\{{}^k\mathbf{I}_l, {}^k\mathbf{I}_r\}$, where the lower right indexs l and r stand for left and right images, feature extraction (denoted as Θ) is performed with keypoints $\{{}^k\mathbf{P}_l, {}^k\mathbf{P}_r\}$ and their descriptors $\{{}^k\mathbf{D}_l, {}^k\mathbf{D}_r\}$ as outcome. They are then used to perform stereo matching (Ψ_s) via Epipolar search, resulting in a set of 2D stereo correspondences ${}^k\mathbf{C}_s$. The first frame is initialized as the world coordinate frame and an initial map $\mathbf{M} = {}^k\mathbf{M}$ is created with all the stereo correspondences via back-projection (π^{-1}) using intrinsic matrix $\mathbf{K}$. The map $\mathbf{M}$ is maintained and updated in the proposed pipeline for the

Algorithm 1 Proposed Visual Odometry Pipeline

Require: $\mathbf{I} = \{{}^1\mathbf{I}_l, {}^1\mathbf{I}_r, ..., {}^n\mathbf{I}_l, {}^n\mathbf{I}_r\}$
Ensure: $\mathbf{X} = \{{}^0\mathbf{X}_1, ..., {}^0\mathbf{X}_n\}$
1: **for** $(k = 1; k \leq n; k{+}{+})$ **do**
2: $\{{}^k\mathbf{P}_l, {}^k\mathbf{D}_l\} \leftarrow \Theta({}^k\mathbf{I}_l)$, $\{{}^k\mathbf{P}_r, {}^k\mathbf{D}_r\} \leftarrow \Theta({}^k\mathbf{I}_r)$;
3: ${}^k\mathbf{C}_s \leftarrow \Psi_s({}^k\mathbf{P}_l, {}^k\mathbf{D}_l, {}^k\mathbf{P}_r, {}^k\mathbf{D}_r)$;
4: **if** $(k = 1)$ **then**
5: ${}^0\mathbf{X}_k = \mathbf{I}_4$;
6: $\mathbf{M} = {}^k\mathbf{M} \leftarrow \Pi^{-1}({}^k\mathbf{C}_s, \mathbf{K}, {}^0\mathbf{X}_k)$;
7: continue ;
8: **else**
9: ${}^{k-1}\mathbf{M}_{temp} \leftarrow \Pi^{-1}({}^{k-1}\mathbf{S}, d_{th}, \mathbf{K}, {}^0\mathbf{C}_{k-1})$;
10: ${}^k\mathbf{M} \leftarrow {}^{k-1}\mathbf{M} \cup {}^{k-1}\mathbf{M}_{temp}$;
11: **end if**
12: ${}^{k-1}_{k}\mathbf{C}_q \leftarrow \Psi_q({}^k\mathbf{M}, {}^{k-1}\mathbf{D}, {}^k\mathbf{D}, {}^{k-1}\mathbf{S}, {}^k\mathbf{S})$;
13: $\{{}^0\hat{\mathbf{X}}_k, {}^k\mathbf{M}_{inlier}\} \leftarrow \text{P3PRANSAC}({}^{k-1}_{k}\mathbf{C}_q, {}^k\mathbf{M})$;
14: ${}^0\mathbf{X}^*_k \leftarrow \Phi({}^{k-1}_{k}\mathbf{C}_q, {}^k\mathbf{M}_{inlier}, {}^0\hat{\mathbf{X}}_k)$;
15: $\mathbf{M} \leftarrow \mathbf{M} \cup {}^k\mathbf{M}_{inlier}$;
16: **end for**
17: **return X ;**

purpose of performing local bundle adjustment (LBA). Furthermore, the LBA process is used in outlier rejection process by removing the points with high re-projection error.

Except for the initialization, as shown in step 9 and 10, not all the stereo correspondences (${}^k\mathbf{C}_s$) are used to to find the temporal correspondences ${}^{k-1}_{k}\mathbf{C}_q$. Instead, only the union set of 1) ${}^{k-1}\mathbf{M} \in \mathbf{M}$, the 3D points that can be found in frame $\mathbf{I}_{i-1}$, and 2) a set of new back-projected 3D points $\mathbf{M}_{temp}$ from $\mathbf{I}_{i-1}$ that are obtained according to their depth values in increasing order up to a pre-set depth threshold that is based on a coarse estimate of the scale of visibility in underwater. In this way, only reliable 3D points are used and the motion estimation is more robust and accurate.

Now, the 2D stereo features in ${}^{k-1}\mathbf{I}$ are used to find their temporal correspondences in ${}^k\mathbf{I}$ via Quad matching (Ψ_q). Figure 4.3 and Algorithm 2 illustrates the details of quad matching, which is the implementation of step 12 in Algorithm 1. For each frame step, stereo correspondences are computed by Epipolar line search. Then each keypoint of stereo correspondence in previous left and right frame performs a window search ($\mathbf{W}$) in current left and right frame, to get their optimal correspondences by comparing their descriptor distances ($\mathbf{D}_{dist}$), respectively. If their optimal correspondences happen to be the stereo correspondences in the current frame (${}^k\mathbf{c}^{i*}_s \in {}^k\mathbf{C}_s$), these four keypoints are accepted as a set of quad correspondences. The choice of window size can not be too small to have insufficient correspondences, or too big to increase the computational cost, as well as decrease the number of valid correspondences due to having more false matches. As demonstrated in Figure 4.4, we empirically set it to 100x100 for all the tested dataset in this chapter, which shows a reasonable balance between computational cost and quantity of correspondences.

At the end of the loop (step 13 and 14 in Algorithm 1), a P3P estimation with RANSAC is

Algorithm 2 Quad Matching

Require: :
1: ${}^{k-1}\mathbf{C}_s=\left\{{}^{k-1}\mathbf{c}_s^i \mid {}^{k-1}\mathbf{c}_s^i=\{{}^{k-1}\mathbf{p}_l^i, {}^{k-1}\mathbf{p}_r^i\}\right\}$, stereo correspondence in ${}^{k-1}\mathbf{I}$;
2: ${}^{k}\mathbf{C}_s=\left\{{}^{k}\mathbf{c}_s^i \mid {}^{k}\mathbf{c}_s^i=\{{}^{k}\mathbf{p}_l^i, {}^{k}\mathbf{p}_r^i\}\right\}$, stereo correspondence in ${}^{k}\mathbf{I}$;
Ensure: ${}^{k-1}_{k}\mathbf{C}_q$, temporal correspondences between ${}^{k-1}\mathbf{I}$ and ${}^{k}\mathbf{I}$;
3: **for** (each ${}^{k-1}\mathbf{c}_s^i=\{{}^{k-1}\mathbf{p}_l^i, {}^{k-1}\mathbf{p}_r^i\} \in {}^{k-1}\mathbf{C}_s$) **do**
4: $\mathbf{S}_l \leftarrow \mathbf{W}({}^{k}\mathbf{I}_l)$, $\mathbf{S}_r \leftarrow \mathbf{W}({}^{k}\mathbf{I}_r)$;
5: ${}^{k}\mathbf{p}_l^{i*}=\{{}^{k}\mathbf{p}_l^i \in \mathbf{S}_l \mid \mathbf{D}_{dist}({}^{k}\mathbf{p}_l^i, {}^{k-1}\mathbf{p}_l^i)$ is minimum$\}$;
6: ${}^{k}\mathbf{p}_r^{i*}=\{{}^{k}\mathbf{p}_r^i \in \mathbf{S}_r \mid \mathbf{D}_{dist}({}^{k}\mathbf{p}_r^i, {}^{k-1}\mathbf{p}_r^i)$ is minimum$\}$;
7: **if** $({}^{k}\mathbf{c}_s^{i*}=({}^{k}\mathbf{p}_l^{i*}, {}^{k}\mathbf{p}_r^{i*}) \in {}^{k}\mathbf{C}_s)$ **then**
8: ${}^{k-1}_{k}\mathbf{C}_q \leftarrow \{{}^{k-1}\mathbf{c}_s^i, {}^{k}\mathbf{c}_s^{i*}\}$;
9: **end if**
10: **end for**
11: **return** ${}^{k-1}_{k}\mathbf{C}_q$ **;**

performed to get an initial pose and inliers, and then a non-linear optimization (Φ) is applied to refine the pose with all the inliers, which is based on Levenberg-Marquardt method implemented in g2o Kümmerle et al. [2011].

In comparison, the VO pipeline in ORB-SLAM2 is demonstrated in Algorithm 3, where the differences can be summarized as follows:

- During initialization, as no motion has been recovered yet, Bag-of-Words (BoW) matching Gálvez-López and Tardos [2012] (Ψ_b) is utilized in the first two frame to accomplish feature association in ORB-SLAM2 (Details of BoW construction are omitted for simplification), see step 12 and 14 of Algorithm 3, whereas it is unnecessary in the proposed pipeline.

- In the temporal matching step, a coarse current camera pose is calculated using constant motion model $\mathbf{T}$, then the temporal correspondences are found by projecting the selected points ${}^{k}\mathbf{M}$ into current frame ${}^{k}\mathbf{I}$ and search locally (Ψ_p). Nevertheless, this becomes invalid in underwater environments because of the motion inconsistency caused by water medium and the difficulty of AUV control in underwater. One possible solution would be to integrate a constant acceleration model to predict a new motion model. However, the matching step becomes complicated with additional parameters being introduced. Instead, we propose a simple but effective Quad matching approach to find the correspondences, which does not rely on motion model, hence being robust to unpredictable motion.

- Only non-linear optimization is performed to recover the camera pose. In this case, the result is easily affected by the initial value, *i.e.*, coarse result computed from constant motion model. If the motion model is not reliable, the optimization may not get the optimal solution or even end up with failure. This has been observed from the experiment that the system loses track easily only by non-linear optimization.

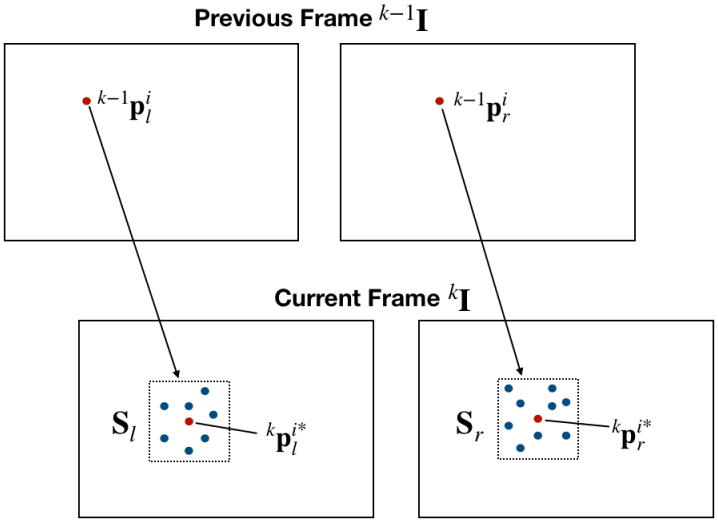


Figure 4.3: **Sketch map of the proposed quad matching method.** Dashed square refers to the search area, and **S** denotes the set of feature points found within the search area.

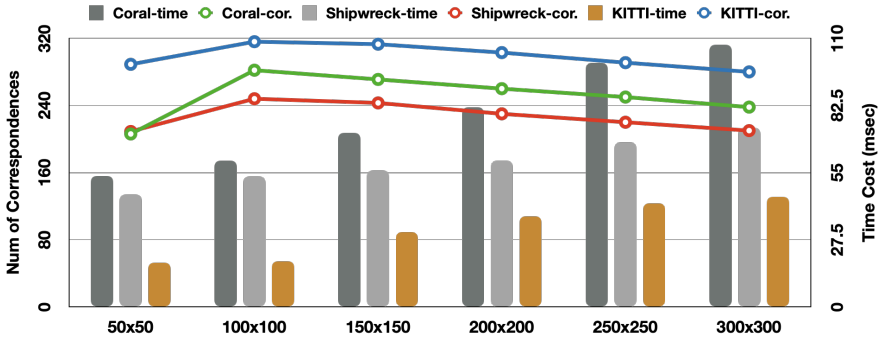


Figure 4.4: **The performance of quad matching with regard to different window sizes.** The curves refer to number of quad correspondences, which corresponds to the left Y-axis. The color bars refer to computational cost that is corresponding to the right Y-axis.

4.4 Experimental Results

In the following, qualitative evaluations of the proposed method on the two underwater datasets, as well as quantitative results on KITTI dataset are demonstrated. Results of two state-of-the-art methods are also shown for comparison.

4.4.1 Underwater Datasets

First, results of the proposed system performing visual odometry task on the Underwater Coral Dataset is shown. As is illustrated in Figure 4.5 (a), the proposed method successfully recovers the whole camera trajectory, which is very close to the real trajectory, as well as the 3D structure. The proposed method is also compared with two state-of-the-art VO systems: LIBVISO2 Geiger et al. [2011] and ORB-SLAM2 Mur-Artal and Tardós [2017]. For fair comparison, the loop closure module (including global bundle adjustment) in ORB-SLAM2 is

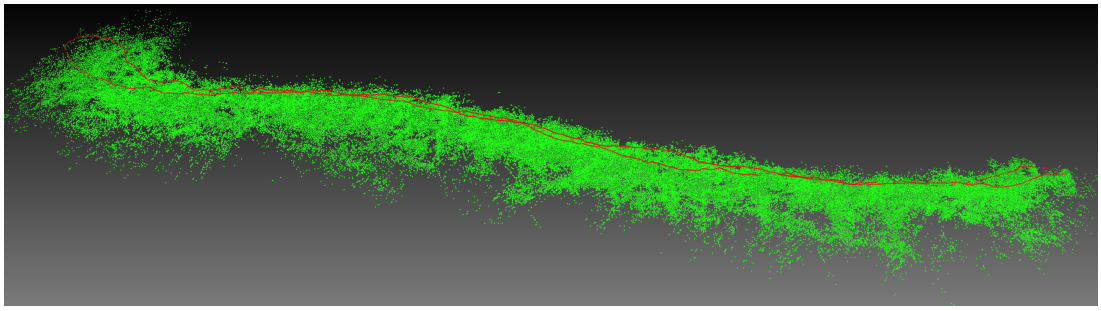
Algorithm 3 Visual Odometry pipeline in ORB-SLAM2

Require: $\mathbf{I} = \{{}^1\mathbf{I}_l, {}^1\mathbf{I}_r, ..., {}^n\mathbf{I}_l, {}^n\mathbf{I}_r\}$
Ensure: $\mathbf{X} = \{{}^0\mathbf{X}_1, ..., {}^0\mathbf{X}_n\}$
for $(k = 1; k \leq n; k{+}{+})$ **do**
  $\{{}^k\mathbf{P}_l, {}^k\mathbf{D}_l\} \leftarrow \Theta({}^k\mathbf{I}_l)$, $\{{}^k\mathbf{P}_r, {}^k\mathbf{D}_r\} \leftarrow \Theta({}^k\mathbf{I}_r)$;
  ${}^k\mathbf{C}_s \leftarrow \Psi_s({}^k\mathbf{P}_l, {}^k\mathbf{D}_l, {}^k\mathbf{P}_r, {}^k\mathbf{D}_r)$;
  if $(k = 1)$ **then**
    ${}^0\mathbf{X}_k = \mathbf{I}_4$;
    $\mathbf{M} = {}^k\mathbf{M} \leftarrow \Pi^{-1}({}^k\mathbf{C}_s, \mathbf{K}, {}^0\mathbf{X}_k)$;
    continue ;
  else
    ${}^{k-1}\mathbf{M}_{temp} \leftarrow \Pi^{-1}({}^{k-1}\mathbf{S}, d_{th}, \mathbf{K}, {}^0\mathbf{C}_{k-1})$;
    ${}^k\mathbf{M} \leftarrow {}^{k-1}\mathbf{M} \cup {}^{k-1}\mathbf{M}_{temp}$;
  end if
  if $(k = 2)$ **then**
    ${}^{k-1}_{k}\mathbf{C}_q \leftarrow \Psi_b({}^k\mathbf{M}, {}^{k-1}\mathbf{D}, {}^k\mathbf{D}, {}^{k-1}\mathbf{S}, {}^k\mathbf{S})$;
    $\{{}^0\mathbf{X}_k, \mathbf{T}\} \leftarrow \Phi({}^{k-1}_{k}\mathbf{C}_q, {}^k\mathbf{M}, {}^0\hat{\mathbf{X}}_k)$;
  else
    ${}^0\hat{\mathbf{X}}_k = \mathbf{T}\,{}^0\mathbf{X}_{k-1}$;
    ${}^{k-1}_{k}\mathbf{C}_q \leftarrow \Psi_p({}^k\mathbf{M}, {}^0\hat{\mathbf{X}}_k, {}^{k-1}\mathbf{D}, {}^k\mathbf{D}, {}^{k-1}\mathbf{S}, {}^k\mathbf{S})$;
    $\{{}^0\mathbf{X}^*_k, \mathbf{T}\} \leftarrow \Phi({}^{k-1}_{k}\mathbf{C}_q, {}^k\mathbf{M}, {}^0\hat{\mathbf{X}}_k)$;
    $\mathbf{M} \leftarrow \mathbf{M} \cup {}^k\mathbf{M}$;
  end if
end for
return X ;

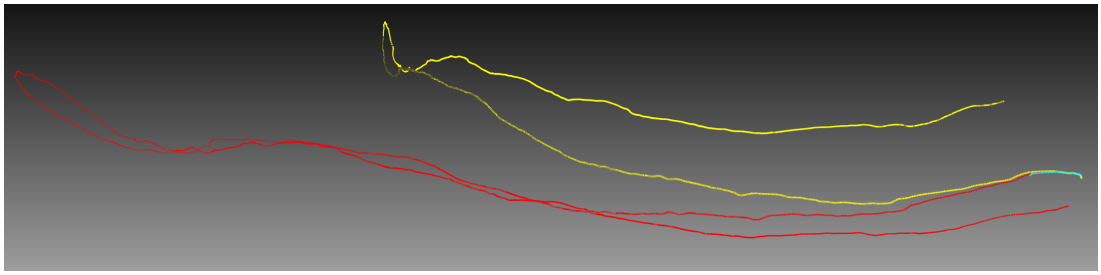
removed to make it a pure visual odometry system. Besides, all the shared parameters in the proposed method and ORB-SLAM2 are set as the same, and the parameters in LIBVISO2 are kept as the default setup. Figure 4.5 (b) shows that LIBVISO2 succeeds to run on the whole dataset, but the trajectory (yellow colour) has drifted away from our trajectory, with the end point being further away from the starting point. ORB-SLAM2 can only survive the first 60 frames (cyan colour), and even with reset, it loses track quickly.

Figure 4.6 (a) shows the whole trajectory and the 3D structure generated from the challenging Underwater Shipwreck Dataset using the proposed method. It can be seen that drift accumulates all along the way, especially when the camera takes a turn and comes back from the stern. Figure 4.6 (b) reveals the inlier distribution along the trajectory. This result intuitively shows the the drift correlates closely with the inlier number. As an example, at the end of the turn (see the black bounding box), where the inlier numbers are lower, the drift of the rotation becomes larger.

The tracking time on the two underwater datasets is around 2.8 seconds per frame when run on an i7 quad-core 2.5Ghz laptop. This is mainly because of the high amount of feature extraction and matching (see Section 4.5.1). This can be improved by employing a GPU-based parallel implementation to achieve real-time performance.



(a)



(b)

Figure 4.5: **Qualitative results on Coral Dataset.** (a) Camera trajectory (red) and the 3D structure (green) produced by the proposed method. (b) Comparison of trajectories generated by the proposed method (red), LIBVISO2 (yellow) and ORB-SLAM2 (cyan).

4.4.2 KITTI Dataset

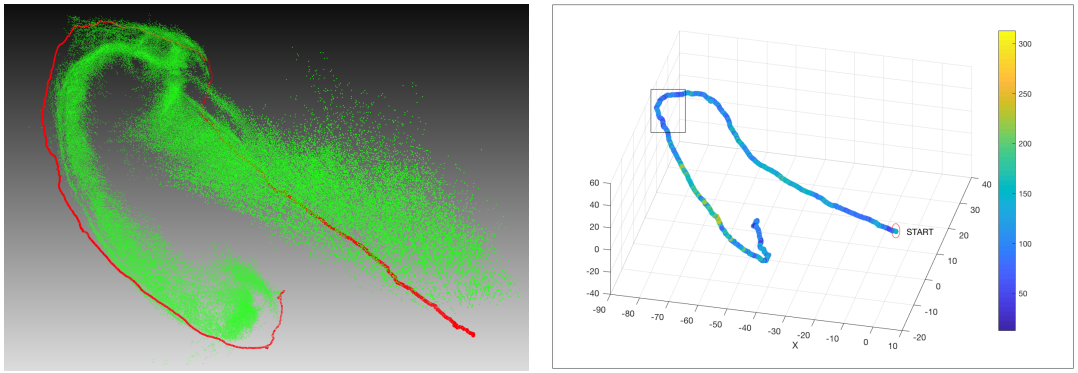
In order to evaluate the proposed VO system, the three methods are also tested on the KITTI benchmark dataset Geiger et al. [2012], which has ground truth trajectories. Table 4.1 demonstrates the quantitative results of the three methods, where the Relative Pose Error Sturm et al. [2012] of rotation (degree) and translation (meter) is computed, respectively. Overall, the proposed method has comparable performance with original version of ORB-SLAM2, and better performance than LIBVISO2. Figure 4.7 illustrates the absolute trajectories compared with the ground truth trajectory. The results of the proposed method is quite close to ground truth trajectory, while LIBVISO2 introduces larger drift to the estimates.

4.5 Discussions

Light is scattered by floating particles in the water, bringing turbidity effects on the captured images, and it is also absorbed by the water itself, causing blur and loss of contrast. Additionally, the light attenuation generates perceived color distortions at different distances and shorten the visual perception to limited distance, making the image less informative. To further exploit valid underwater image information for the VO problem, two important modules are analysed and discussed: feature extraction and image enhancement.

Table 4.1: Comparison of translation (meter) and rotation (degree) RMSE in KITTI dataset.

Method	Sequence	00	01	02	03	04	05	06	07	08	09	10
Proposed	R	**0.6195**	0.2330	**0.2019**	**0.0846**	0.0704	**0.1881**	**0.1267**	**0.1485**	0.1413	**0.1259**	**0.1316**
	t	0.2083	8.8563	0.1908	**0.1047**	0.1395	**0.0923**	0.0995	0.0927	**0.3410**	**0.1534**	0.1179
ORB-SLAM2	R	0.6196	**0.1318**	0.2048	0.0891	**0.0696**	0.1987	0.1303	0.1539	**0.1410**	0.1318	0.1355
	t	0.2032	**0.4546**	0.1933	0.1106	**0.1234**	0.0947	**0.0938**	0.0909	0.3424	0.1656	0.1134
LIBVISO2	R	0.6471	0.2883	0.2384	0.1404	0.1596	0.2512	0.2251	0.2513	0.2149	0.2023	0.1831
	t	**0.1941**	3.2146	**0.1810**	0.1185	0.1715	0.1024	0.1199	**0.0830**	0.3415	0.1694	**0.1076**



(a) (b)

Figure 4.6: **Qualitative results on Shipwreck Dataset.** (a) Camera trajectory (red) and the 3D structure (green) for the shipwreck dataset, produced by the proposed method. (b) Color bar showing the distribution of inlier number on the whole trajectory.

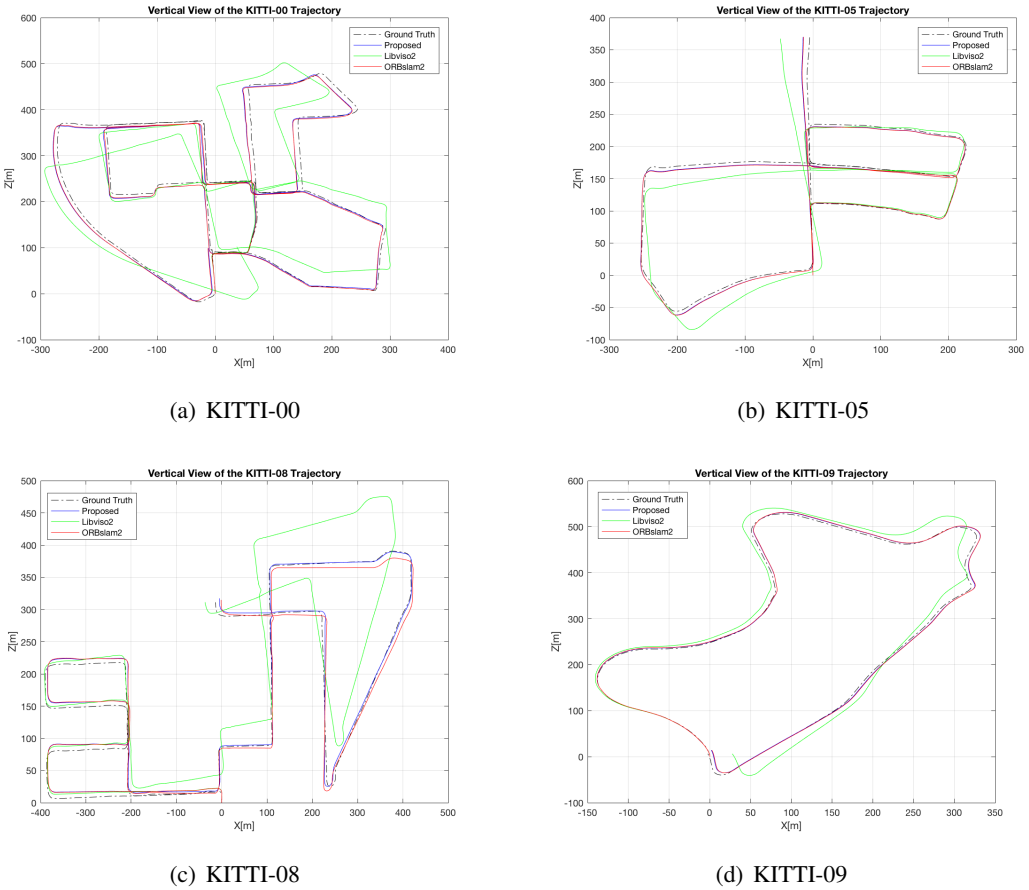
4.5.1 Feature Extraction

In underwater environments, feature should be carefully selected to be robust against poor illumination, haze and scattering. SIFT Lowe [2004] feature is one of the highest quality feature descriptors due to its strong invariance to scale, rotation, illumination change and noise. However, it requires a large computational complexity which is a major drawback for real-time applications such as visual odometry. SURF Bay et al. [2006] feature−an approximation of SIFT−performs faster than SIFT without reducing much quality of the detected points. Alternatively, ORB Rublee et al. [2011] is another efficient choice, which is a binary descriptor requiring less complexity but is still highly distinctive.

Table 4.2: Comparison of average inlier number obtained by three feature techniques.

Feature Techniques	SIFT	SURF	ORB	Mean Value
Coral Dataset	128	**169**	134	144
Shipwreck Dataset	126	**129**	119	125
Time Cost (sec/frame)	2.98	2.53	**0.41**	-

To analyse their performance on underwater images, the three feature techniques are tested on the proposed visual odometry pipeline separately. More in details, FAST Rosten and Drummond [2006] corners at 8 scale levels are detected uniformly distributed on the image. To ensure that enough features are obtained to track the camera, the number of detected features is set to 6000 per image (in size 1920×1080) in both underwater datasets to increase possibility of getting higher number of valid features against the bad imaging quality. Then SIFT, SURF and ORB descriptors are extracted from these corners for comparison. during the matching step, sub pixel correlation, orientation and scale consistency are also considered to remove false correspondences.



(a) KITTI-00 (b) KITTI-05

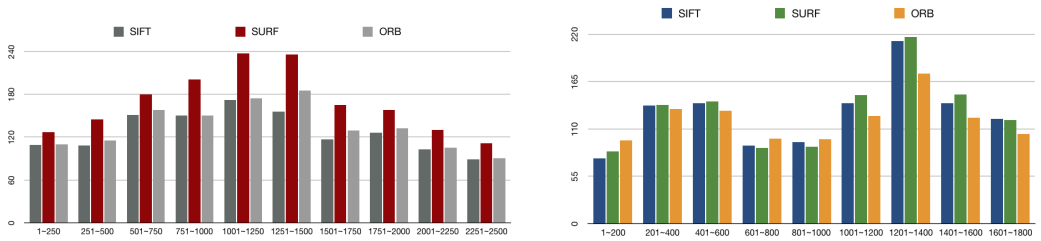
(c) KITTI-08 (d) KITTI-09

Figure 4.7: **Trajectory results on part of the representative sequences of KITTI dataset.** Dashed line represents ground truth trajectory.

Their performance is evaluated by comparing the number of matched inliers, which are obtained after the camera pose between subsequent frames is estimated using the correspondences found by each type of descriptor. This is reasonable because, by getting more inliers, the accuracy of pose estimation is higher. Table 4.2 presents the average inlier number acquired by the tested feature technics on both underwater datasets, which intuitively show that SURF techniques outperforms the other two, especially in the Underwater Coral Dataset. The mean values between datasets shows the Coral Dataset obtains approximately 15% more inliers than that of the Shipwreck Dataset. This accounts for the truth that the imaging condition in Shipwreck Dataset is poorer and less valid features can be used to track. In terms of computational cost, however, the process of extracting SURF requires average 2.5 second per frame, which is over 5 times more than ORB. This can be improved by applying GPU implementations for acceleration.

A more detailed histogram distribution of the inlier number is demonstrated in Figure 4.8, which reflects the characters of both datasets. Specifically, for instance, in (a) there are less inliers at the beginning and towards the end, because the camera is heading forward and most

of the scene during that time is half visible, while in the middle frames (1000-1500) the inliers increase, as the camera is down-looking at the ground and taking a turn, so more features can be detected and tracked. In (b), for another example, the frames between 600 and 1000 gets lowest inliers among the whole sequence, when the camera is approaching the stern and turning back. There is little structure in the scene to be tracked in this area. That is why the drift grows quickly during that time, as shown in Figure 4.6 (b).



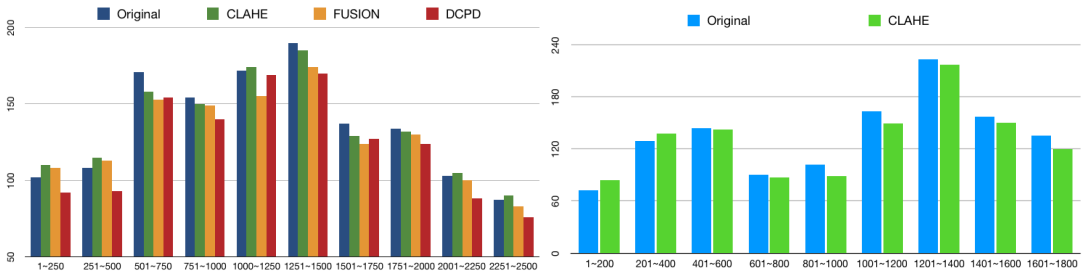
(a) (b)

Figure 4.8: **Comparison of inliers number among the three feature techniques.** (a) Underwater Coral Dataset; (b) Underwater Shipwreck Dataset. The vertical axis refers to inlier number, and the horizontal axis refers to frame index.

4.5.2 Image Enhancement

An extra experiment is conducted to compare different image enhancement algorithms that are used to improve the turbid image quality before feature extraction, and see if more reliable features can be extracted to estimate motion. Many image enhancement or dehazing methods have been proposed to tackle this problem in the literature. Here, three of the state of the arts are selected to compare and evaluate how enhancing or dehazing techniques would affect the visual odometry task. Concretely, contrast-limited adaptive histogram equalization (CLAHE) Zuiderveld [1994] is a widely-used image contrast enhancement algorithm, which divides the images into regions and performs local histogram equalization (HE) and reduces noise by partially reducing the local HE. Underwater images and videos enhancement by fusion (FUSION) Ancuti et al. [2012] is a fusion-based framework that blends different filters to enhance underwater images. Dark channel prior dehazing (DCPD) He et al. [2011] method is based on a kind of statistics of the haze-free outdoor images called dark channel prior. Together with haze imaging model, the thickness of the haze can be directly calculated and finally a high quality haze-free image can be recovered. For parameter setup, we set the clipping term in CLAHE to 4.2, which shows reasonably good performance. The parameters in FUSION and DCPD are kept the same as default.

Figure 4.10 shows a comparison of enhancement results on both underwater datasets. Qualitatively, all the tested algorithms have improved the visibility at different levels, compared with the original image. More precisely, results of CLAHE and FUSION are brighter and have larger visible regions, i.e., far away unclear structure becomes more clear. On the contrary, the luminance becomes lower when applying DCPD, and not much visibility improvement is obtained, but the structure details become finer.



(a) (b)

Figure 4.9: **Comparison of inliers number among the enhancement methods.** (a) Underwater Coral Dataset; (b) Underwater Shipwreck Dataset. The vertical axis refers to inlier number, and the horizontal axis refers to frame index.

To evaluate their performance on visual odometry, they are all applied to test the proposed VO pipeline on both underwater datasets, separately. Note that many images of Underwater Shipwreck Dataset become even worse after applying FUSION and DCPD methods, which makes the system unable to extract sufficient features to track and fail in most cases. Therefore, only the CLAHE is compared with the default in this dataset. Similar to feature techniques comparison, the inlier number obtained in each frame are used as a comparative index. The comparison of inlier number distribution is illustrated in Figure 4.9. It can be observed from (a) that, in some parts of the dataset (for instance, 1-250, 251-500, *etc.*), CLHAE and FUSION have increased the average inlier number, but in some other parts, they got even much less inliers than the original (501-750, *etc.*). Similar trend can be observed in (b), as well. This suggests that certain level of noise has been introduced when the image is enhanced using the surveyed methods.

Table 4.3: Comparsion of qualtity results towards different enhancement methods.

Underwater Coral Dataset				
Enhancement Techniques	Oiginal	CLAHE	FUSION	DCPD
Average Match Number	**242**	237	224	219
Average Inlier Number	**135**	134	129	123
Average Inlier Rate	0.5425	0.5538	**0.5613**	0.5429
Underwater Shipwreck Dataset				
Enhancement Techniques	Original		CLAHE	
Average Match Number	**257**		237	
Average Inlier Number	**135**		131	
Average Inlier Rate	0.5253		**0.5527**	

Table 4.3 demonstrates the average values of the whole sequence. In particular, the average matching number drops slightly after enhancement. This means that less extracted features are qualified to be chosen in the proposed matching procedure despite the fact that the scene visibility has been improved by enhancement. Therefore, I believe that enhancement measure

should have introduced certain level of noise to the image and brought negative effect on the completeness and uniqueness in feature descriptor. Overall, the results reveal the fact that none of the approaches has significant contribution to the visual odometry task, though they do increase the image visibility in accordance with the human's observing experience.

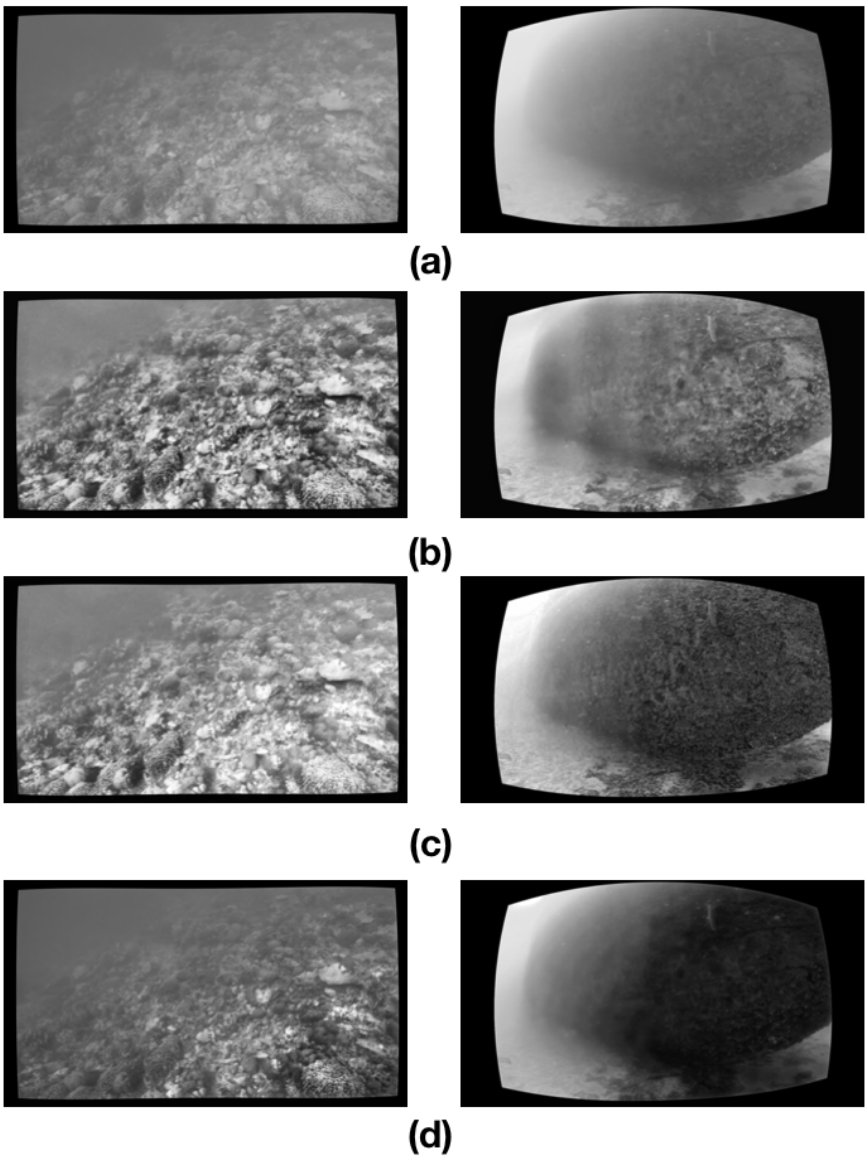


Figure 4.10: **Comparison of different image enhancement algorithms on the Coral (left) and Shipwreck (right) datasets.** (a) Original image. (b) Enhanced by CLAHE. (c) Enhanced by FUSION. (d) Enhanced by DCPD.

4.6 Conclusion

In this chapter, a robust and effective stereo underwater VO system is introduced to robustly and accurately recover the camera motion. Careful analysis for each part of the proposed visual odometry system is conducted, including image restoration, feature extraction and matching, motion estimation, to explore possibility of improvement on the system in underwater environments. Experimental results indicate that the proposed system helps to achieve excellent performance in localizing the camera in underwater and obtains satisfactory results in the KITTI benchmark dataset by comparing to the state of the arts and the ground truth.

Nevertheless, there are problems that remain to be solved. One of them is how to improve the underwater image quality for underwater visual odometry. The goal should be to increase the number of valid corners to be detected, simultaneously preserve the completeness of their features and avoid the increase of noise. Many image enhancement or dehazing methods has been proposed and some are special designed for underwater environment. However, experimental results has shown that most of them are not able to help effectively improving the task of underwater localization. I believe it is worth to look into this specific problem and explore for an effective solution.

Chapter 5

Robust Ego and Object 6-DoF Motion Estimation and Tracking

The problem of tracking self-motion as well as motion of objects in the scene using information from a camera is known as multi-body visual odometry and is a challenging task. In this chapter, a robust solution is proposed to achieve accurate estimation and consistent track-ability for dynamic multi-body visual odometry. Specifically, a compact and effective framework is constructed, which leverages recent advances in semantic instance-level segmentation and accurate optical flow estimation. A novel formulation, jointly estimating $\mathrm{SE}(3)$ motion and optic-flow is introduced to improve the quality of the tracked points and the motion estimation accuracy. The proposed approach is evaluated on the virtual KITTI Dataset Gaidon et al. [2016] and tested on the real KITTI Dataset Geiger et al. [2012], demonstrating its applicability to autonomous driving applications.

5.1 Motivation

Visual odometry (VO) has been a popular solution for robot navigation in the past decade due to its low-cost and widely applicable properties. Studies in the literature have illustrated that VO can provide accurate estimation of a camera trajectory in largely static environment, with relative position error ranging from 0.1% to 2% Scaramuzza and Fraundorfer [2011]. However, the deployment of robotic systems in our daily lives requires systems to work in significantly more complex, dynamic environments. Visual navigation in non-static environments becomes challenging because the dynamic parts in the scene violate the motion model of camera. If moving parts of a scene dominate, off-the-shelf visual odometry systems either fail completely or return poor quality trajectory estimation. Earlier solutions proposed to directly remove the dynamic information via robust estimation Tan et al. [2013]; Alcantarilla et al. [2012], however, this information is valuable if it is properly used. In most scenarios, the dynamics corresponds to a finite number of individual objects that are rigid or piecewise rigid, and their motions can be tracked and estimated in the same way as the ego-motion. Accurate object motion estimation and tracking becomes highly relevant in many applications, such as collision avoidance in autonomous driving and robotic systems, visual surveillance and augmented reality.

In this chapter, a novel multi-body visual odometry pipeline is presented to address the problem of tracking of both ego and object motion in dynamic outdoor scenes, assuming the

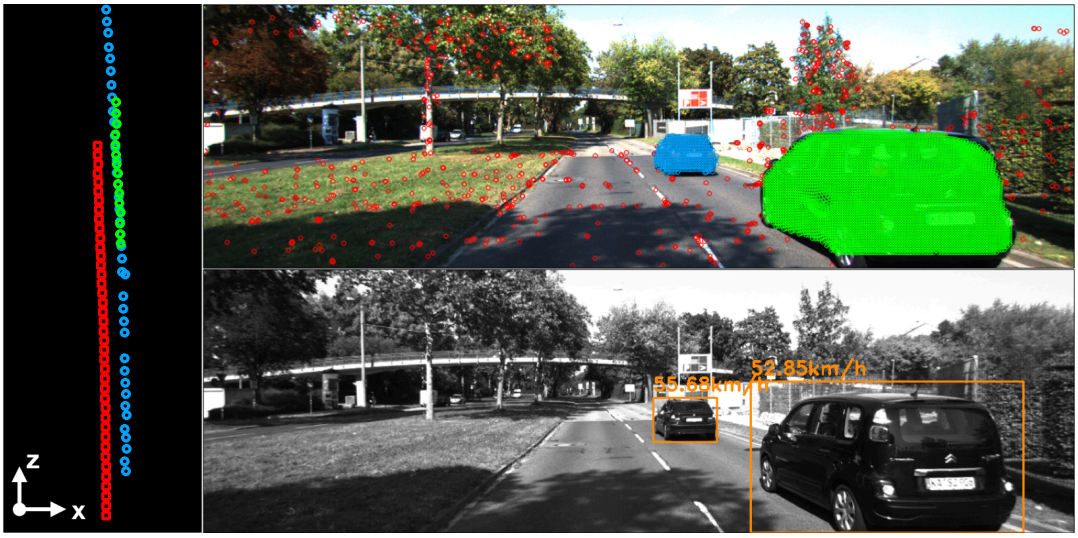


Figure 5.1: **Results of the proposed system on KITTI sequence** 03**.** Camera and object trajectory (left). Detected points on background and object body (top-right). Estimated object speed (bottom-right).

targeted objects to be rigid or near rigid. The proposed pipeline leverages instance-level object segmentation algorithms He et al. [2017] to robustly separate the scene into static background and multiple dynamic objects. Recent advances in optic-flow estimation Ilg et al. [2017]; Sun et al. [2018] provide good initial optical flow that can be used to maintain enough tracking points on each object to accurately estimate motion. With this data, a new technique is proposed to jointly refines the initial optical flow and estimate full 6-DoF motion of both the camera and objects in the scene. A fully-integrated system is constructed, which is able to robustly estimate and track self and object motions utilizing only visual sensors (stereo/RGB-D). Arguably, the proposed work is the first to conduct an extensive evaluation of accuracy and robustness of ego and object 6-DoF motion estimation and tracking, and demonstrates the feasibility on real-world outdoor datasets. Note that, as the distance from camera increases, the presence of an object in the scene becomes smaller when observed on image. Therefore, the proposed method only processes objects up to a limited distance. Details can be found in the experiment section.

5.2 Methodology

The setup of this work comprises a depth camera (stereo or RGB-D) moving in a dynamic environment. Let ${}^k\mathbf{P} = \{{}^k\mathbf{p}^i \in \mathbb{R}^3\}$ be a set of projected points into the image frame k, where ${}^k\mathbf{p}^i = [u^i, v^i, 1]^\top$ represents the point location in homogeneous coordinates. The points are either part of the static background ${}^k\mathbf{P}_s \subseteq {}^k\mathbf{P}$ or moving object ${}^k\mathbf{P}_o \subset {}^k\mathbf{P}$.

Assuming that a depth map ${}^k\mathbf{D} = \{{}^kd^i \in \mathbb{R}\}$ of frame k is provided, where ${}^kd^i$ is the corresponding depth for each point ${}^k\mathbf{p}^i \in {}^k\mathbf{P}$, the 3D point ${}^k\mathbf{m}^i \in \mathbb{R}^4$ of ${}^k\mathbf{p}^i$ can be obtained via

back-projection:

$$
{}^{k}\mathbf{m}^{i} = \begin{bmatrix} m_x^i \\ m_y^i \\ m_z^i \\ 1 \end{bmatrix} = \pi^{-1}({}^{k}\mathbf{p}^{i}) = \begin{bmatrix} (u^i - c_u) \cdot {}^{k}d^i / f_u \\ (v^i - c_v) \cdot {}^{k}d^i / f_v \\ {}^{k}d^i \\ 1 \end{bmatrix} \tag{5.1}
$$

where $\pi^{-1}(\cdot)$ is the inverse of projection function, (f_u, f_v) the focal length and (c_u, c_v) the principal point of the cameras.

The motion of the camera between frames $k-1$ and k and/or the motion of objects in the scene produce an optical flow ${}^{k}\mathbf{\Phi} = \{{}^{k}\boldsymbol{\phi}^{i} \in \mathbb{R}^2\}$, where ${}^{k}\boldsymbol{\phi}^{i}$ is the corresponding optical flow for each point ${}^{k}\mathbf{p}^{i}$ and its correspondence ${}^{k-1}\mathbf{p}^{i}$ in frame $k-1$, as demonstrated in Figure 5.2 (left) on image plane, and is given by:

$$
{}^{k}\mathbf{p}^{i} = {}^{k-1}\mathbf{p}^{i} + {}^{k}\boldsymbol{\phi}^{i} \tag{5.2}
$$

Here the notation ${}^{k-1}\mathbf{p}^{i}$ and ${}^{k}\mathbf{p}^{i}$ are overloaded and also used to represent the 2D pixel coordinates ($\mathbb{R}^2$). ${}^{k}\mathbf{\Phi}$ can be obtained using off-the-shelf classic or learning-based methods. The motions of the camera and objects in the scene are represented by pose change transformations between two temporal states. The following subsections will describe the new approach to estimate those.

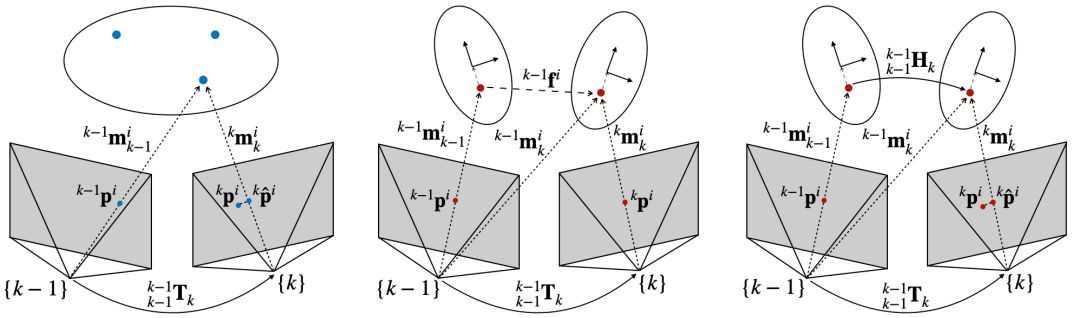


Figure 5.2: **Sketch maps of ego-motion obtained from static points (left), scene flow of points on moving objects (center) and rigid motion of points on moving object (right).** Here blue dots represent static points, and red dots dynamic points.

5.2.1 Camera Motion Estimation

The camera motion between frame $k-1$ (lower left index) and k (lower right index) represented in body-fixed frame $k-1$ (upper left index) is denoted as ${}^{k-1}_{k-1}\mathbf{T}_{k} \in \mathrm{SE}(3)$. The image plane points, associated with static 3D points ${}^{k-1}\mathbf{m}^{i}_{k-1}$, observed at time $k-1$, by the projection onto

the k image plane can now be computed by

$$ {}^{k}\hat{\mathbf{p}}^{i} := \pi({}^{k}\mathbf{m}^{i}_{k-1}) = \pi({}^{k-1}_{k-1}\mathbf{T}^{-1}_{k}\,{}^{k-1}\mathbf{m}^{i}_{k-1}). \tag{5.3} $$

Parameterize the $\mathrm{SE}(3)$ camera motion by elements $\boldsymbol{\xi}^{\wedge}_{k} \in \mathrm{se}(3)$ the Lie-algebra of $\mathrm{SE}(3)$. That is

$$ {}^{k-1}_{k-1}\mathbf{T}_{k} = \exp({}^{k-1}_{k-1}\boldsymbol{\xi}^{\wedge}_{k}) \tag{5.4} $$

where ${}^{k-1}_{k-1}\boldsymbol{\xi}_{k} \in \mathbb{R}^{6}$ and the wedge operator is the standard lift into $\mathrm{se}(3)$. Combining (5.2) and (5.3), and using the Lie-algebra parameterization of $\mathrm{SE}(3)$ the minimizing solution of the least squares cost criteria is given by

$$ {}^{k-1}_{k-1}\boldsymbol{\xi}^{*}_{k} = \underset{{}^{k-1}_{k-1}\boldsymbol{\xi}_{k}}{\operatorname{argmin}} \sum_{i=1}^{n_s} \rho_h(||{}^{k-1}\mathbf{p}^{i} + {}^{k}\boldsymbol{\phi}^{i} - {}^{k}\hat{\mathbf{p}}^{i}||^{2}_{\Sigma_1}) \tag{5.5} $$

for all the visible 3D-2D static point correspondences ($i = 1, ..., n_s$). Here ρ_h is the Huber robust cost function, and Σ_1 is covariance matrix associated to the re-projection threshold used in initialization. The estimated camera motion is given by ${}^{k-1}_{k-1}\mathbf{T}^{*}_{k} = \exp({}^{k-1}_{k-1}\boldsymbol{\xi}^{*}_{k})$ and is found using the Levenberg-Marquardt algorithm to solve for (5.5).

5.2.2 Moving Points Motion Estimation

This section aims to derive the motion model of 3D points on a rigid body in motion, explain how this motion model is estimated.
The motion of the rigid body in body-fixed frame is given by ${}^{L_{k-1}}_{k-1}\mathbf{H}_{k} \in \mathrm{SE}(3)$ as:

$$ {}^{L_{k-1}}_{k-1}\mathbf{H}_{k} = {}^{0}\mathbf{L}^{-1}_{k-1}\,{}^{0}\mathbf{L}_{k}\,. \tag{5.6} $$

where ${}^{0}\mathbf{L}_{k-1}, {}^{0}\mathbf{L}_{k} \in \mathrm{SE}(3)$ are the poses of the object in inertial frame at time $k-1$ and k, respectively. A point in its corresponding object frame is written as:

$$ {}^{L_k}\mathbf{m}^{i}_{k} = {}^{0}\mathbf{L}^{-1}_{k}\,{}^{0}\mathbf{m}^{i}_{k}\,. \tag{5.7} $$

Substituting the object pose at time k from (5.6), this becomes:

$$ {}^{0}\mathbf{m}^{i}_{k} = {}^{0}\mathbf{L}_{k}\,{}^{L_k}\mathbf{m}^{i}_{k} = {}^{0}\mathbf{L}_{k-1}\,{}^{L_{k-1}}_{k-1}\mathbf{H}_{k}\,{}^{L_k}\mathbf{m}^{i}_{k}\,. \tag{5.8} $$

Note that for rigid body objects, ${}^{L_k}\mathbf{m}^{i}_{k}$ stays constant at ${}^{L}\mathbf{m}^{i}$, and ${}^{L}\mathbf{m}^{i} = {}^{0}\mathbf{L}^{-1}_{k}\,{}^{0}\mathbf{m}^{i}_{k} = {}^{0}\mathbf{L}^{-1}_{k+n}\,{}^{0}\mathbf{m}^{i}_{k+n}$ for any integer $n \in \mathbb{Z}$. Then, for rigid objects with $n = -1$, (5.8) becomes:

$$ {}^{0}\mathbf{m}^{i}_{k} = {}^{0}\mathbf{L}_{k-1}\,{}^{L_{k-1}}_{k-1}\mathbf{H}_{k}\,{}^{0}\mathbf{L}^{-1}_{k-1}\,{}^{0}\mathbf{m}^{i}_{k-1}\,. \tag{5.9} $$

Equation (5.9) is crucially important as it relates the same 3D point on a rigid object in motion at consecutive time steps by a homogeneous transformation

$$ {}^{0}_{k-1}\mathbf{H}_{k} := {}^{0}\mathbf{L}_{k-1}\,{}^{L_{k-1}}_{k-1}\mathbf{H}_{k}\,{}^{0}\mathbf{L}^{-1}_{k-1}\,. \tag{5.10} $$

This equation represents a *frame change of a pose transformation* (Chirikjian et al. [2017]), and shows how the body-fixed frame pose change ${}^{L_{k-1}}_{k-1}\mathbf{H}_k$ relates to the global reference frame pose change ${}_{k-1}{}^{0}\mathbf{H}_k \in \mathrm{SE}(3)$. The point motion in global reference frame is then expressed as:

$$ {}^{0}\mathbf{m}^i_k = {}_{k-1}{}^{0}\mathbf{H}_k \, {}^{0}\mathbf{m}^i_{k-1} \,. \tag{5.11} $$

Equation (5.11) is at the core of the proposed motion estimation approach, as it expresses the rigid object pose change in terms of the points that reside on the object in a model-free manner without the need to include the object 3D pose as a random variable in the estimation. Note that, when formulating the motion estimation problem considering only two consecutive frames, the motion in the inertial frame in (5.10) would be expressed in the image frame $k-1$, and is denoted ${}^{k-1}_{k-1}\mathbf{H}_k$.

As shown in Figure 5.2 (right), a 3D point ${}^{k-1}\mathbf{m}^i_{k-1}$ observed on a moving object at time $k-1$, moves according to (5.11) to ${}^{k-1}\hat{\mathbf{m}}^i_k = {}^{k-1}_{k-1}\mathbf{H}_k \, {}^{k-1}\mathbf{m}^i_{k-1}$. The projection of the estimated 3D point onto the image frame at time k is given by

$$ \begin{aligned} {}^{k}\hat{\mathbf{p}}^i := \;& \pi\left({}^{k-1}_{k-1}\mathbf{T}^{-1}_{k} \, {}^{k-1}_{k-1}\mathbf{H}_k \, {}^{k-1}\mathbf{m}^i_{k-1}\right) \\ = \;& \pi\left({}^{k-1}_{k-1}\mathbf{G}_k \, {}^{k-1}\mathbf{m}^i_{k-1}\right) \end{aligned} \tag{5.12} $$

where ${}^{k-1}_{k-1}\mathbf{G}_k \in \mathrm{SE}(3)$. Similar to the camera motion estimation, parameterise ${}^{k-1}_{k-1}\mathbf{G}_k = \exp\left({}^{k-1}_{k-1}\boldsymbol{\zeta}^{\wedge}_k\right)$, with ${}^{k-1}_{k-1}\boldsymbol{\zeta}^{\wedge}_k$ the $\mathrm{se}(3)$ representation of ${}^{k-1}_{k-1}\boldsymbol{\zeta}_k \in \mathbb{R}^6$, and find the optimal solution via minimizing

$$ {}^{k-1}_{k-1}\boldsymbol{\zeta}^{*}_k = \underset{{}^{k-1}_{k-1}\boldsymbol{\zeta}_k}{\operatorname{argmin}} \sum_{i=1}^{n_o} \rho_h\left(||{}^{k-1}\mathbf{p}^i + {}^{k}\boldsymbol{\phi}^i - {}^{k}\hat{\mathbf{p}}^i||^2_{\Sigma_1}\right) \tag{5.13} $$

given all the 3D-2D point correspondences on an object ($i = 1, ..., n_o$). The motion of the object points, ${}^{k-1}_{k-1}\mathbf{H}_k = {}^{k-1}_{k-1}\mathbf{T}_k \, {}^{k-1}_{k-1}\mathbf{G}_k$ can be recovered afterwards.

5.2.3 Refining the estimation of the optical flow

Both, camera motion and object motion estimations 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 in this chapter aims at refining the estimation of the optical flow jointly with the motion estimation:

$$ \begin{aligned} \{\boldsymbol{\theta}^*, {}^{k}\boldsymbol{\Phi}^*\} = & \underset{\{\boldsymbol{\theta}, {}^{k}\boldsymbol{\Phi}\}}{\operatorname{argmin}} \sum_{i=1}^{n} \rho_h\left(||{}^{k-1}\mathbf{p}^i + {}^{k}\boldsymbol{\phi}^i - {}^{k}\hat{\mathbf{p}}^i||^2_{\Sigma_1}\right) \\ & + \rho_h\left(||{}^{k}\boldsymbol{\phi}^i - {}^{k}\hat{\boldsymbol{\phi}}^i||^2_{\Sigma_2}\right) \end{aligned} \tag{5.14} $$

where $\{\boldsymbol{\theta}, {}^{k}\boldsymbol{\Phi}\}$ can be either $\{{}^{k-1}_{k-1}\boldsymbol{\xi}_k, {}^{k}\boldsymbol{\Phi}_s\}$ for camera motion estimation, or $\{{}^{k-1}_{k-1}\boldsymbol{\zeta}_k, {}^{k}\boldsymbol{\Phi}_o\}$ for the object motion estimation, with ${}^{k}\boldsymbol{\Phi}_s \subseteq {}^{k}\boldsymbol{\Phi}$ and ${}^{k}\boldsymbol{\Phi}_o \subset {}^{k}\boldsymbol{\Phi}$. Here Σ_2 is the covariance matrix associated to initial optic-flow obtained using classic or learning-based methods.

5.3 Implementation

In this section, a novel multi-motion visual odometry system is presented to robustly estimates both camera and object motions. The proposed system takes stereo or RGB-D images as input. If the input data is stereo images, the method in Yamaguchi et al. [2014] is applied to generate the depth map **D**. The proposed pipeline is summarised in Figure 5.3 and contains three main parts: image preprocessing, ego-motion estimation and object motion tracking.

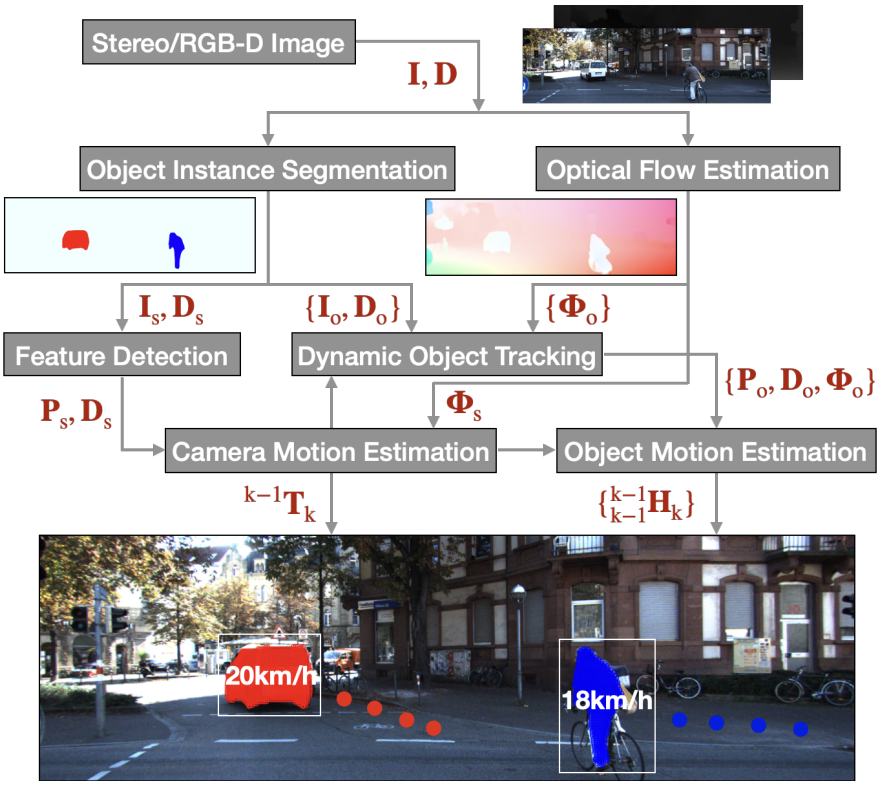


Figure 5.3: **Overview of the proposed multi-motion visual odometry system.** Letters in red colour refer to output for each blocks. $\{\cdot\}$ denotes multiple objects.

5.3.1 Image Preprocessing

There are two challenging aspects that this pipeline needs to fulfil. One is to separate static background and objects, and the other is to ensure long-term tracking of dynamic objects. For that, recent advances in computer vision technics for instance level semantic segmentation and dense optical flow calculation is utilized in order to ensure efficient object motion segmentation and good object tracking.

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 mask provides precise boundary of the object body that ensures robust tracking of points on objects. In this chapter, we apply the open-source method Mask R-CNN He et al. [2017] to obtain the instance segments of images. This method extends Faster R-CNN Ren et al. [2016] by adding a Region of Interest alignment (RoIAlign) layer to predict object mask together with existing framework for bounding box recognition. The high quality segmentation of objects in general outdoor scenes, such as humans and vehicles, makes it a perfect pre-processing tool for our task. For further details on the applied method, we refer the reader to He et al. [2017].

The dense optical flow is used to maximize the number of track points on moving objects. Most of the moving objects, they only occupy a small portion of the image. Therefore, using sparse feature matching does not guarantee a robust feature tracking. The proposed approach makes use of dense optical flow to considerably increase the number of object points. At the same time, the proposed method enhances the matching performance by refining optical flow jointly within the motion estimation process as presented in Section 5.2.3. In this chapter, a recent off-the-shelf method, PWC-Net Sun et al. [2018], is used to generate initial optic-flow map. The method learns a compact but effect CNN model by tightly integrating the principles of pyramidal processing, warping and cost volume, to achieve high performance. We refer readers to Sun et al. [2018] for more details.

The image preprocessing part of the pipeline generates the image mask $\mathbf{I}_s$, the depth $\mathbf{D}_s$ and the dense flow $\mathbf{\Phi}_s$ for the static background, and $\{\mathbf{I}_o, \mathbf{D}_o, \mathbf{\Phi}_o\}$ for dynamic objects in the scene, where $\{\cdot\}$ indicates multiple items.

5.3.2 Ego-motion Estimation

To achieve fast ego-motion estimation, a sparse feature set $\mathbf{P}_s$ is constructed in each frame. Since dense optical flow is available, it is used to match the sparse features across frames. Note that, those sparse features are only detected and matched on regions of the image other than labelled objects. To ensure robust estimation, a motion model generation method is applied for initialisation. Specifically, the method generates two motion models and compares their inlier numbers based on re-projection error. One model is generated by propagating the previous camera motion, while the other by computing a new motion transform using P3P Ke and Roumeliotis [2017] algorithm with RANSAC. The motion model that produces most inliers is then selected for initialisation. As propagating motion is not available during initialization, the model generated by P3P is used. Details of the motion model generator is demonstrated in Algorithm 4.

5.3.3 Object Motion Tracking

The process of object motion tracking consists of three steps. The first step is to classify all the objects into dynamic and static objects. Then, the dynamic objects across the two frames is associated. Finally, individual object motion is estimated.

Algorithm 4 Motion Model Generator

Require:
${}^{k-1}\mathbf{M} = \{{}^{k-1}\mathbf{m}^i_{k-1} \mid i = 1,...,N\}$: 3D points in frame $k-1$;
${}^{k}\mathbf{P} = \{{}^{k}\mathbf{p}^i \mid i = 1,...,N\}$: corresponding 2D points in frame k;
$\mathbf{X}_a \in \mathrm{SE}(3)$: prior motion model from frame $k-1$;
e_{th}: a threshold of re-projection error;
Ensure:
${}^{k-1}\mathbf{M}^* = \{{}^{k-1}\mathbf{m}^j_{k-1} \mid j = 1,...,n\}$: a subset of points in $k-1$;
${}^{k}\mathbf{P}^* = \{{}^{k}\mathbf{p}^j \mid j = 1,...,n\}$: its corresponding 2D subset in k;
$\mathbf{X}^* \in \mathrm{SE}(3)$: best motion model;

for (each ${}^{k-1}\mathbf{m}^i_{k-1} \in {}^{k-1}\mathbf{M}$ and ${}^{k}\mathbf{p}^i \in {}^{k}\mathbf{P}$) **do**
  $e_r \leftarrow$ ComputeReprojectionError$({}^{k-1}\mathbf{m}^i_{k-1}, {}^{k}\mathbf{p}^i, \mathbf{X}_a)$;
  if $(e_r < e_{th})$ **then**
    ${}^{k-1}\mathbf{M}_a \leftarrow {}^{k-1}\mathbf{m}^i_{k-1}$, ${}^{k}\mathbf{P}_a \leftarrow {}^{k}\mathbf{p}^i$;
  end if
end for
$n_a = |{}^{k-1}\mathbf{M}_a|$: inlier number that fits $\mathbf{X}_a$;
$\{{}^{k-1}\mathbf{M}_b, {}^{k}\mathbf{P}_b, \mathbf{X}_b\} \leftarrow$SolveP3PRANSAC$({}^{k-1}\mathbf{M}, {}^{k}\mathbf{P})$;
$n_b = |{}^{k-1}\mathbf{M}_b|$: inlier number that fits $\mathbf{X}_b$;
if $(n_a > n_b)$ **then**
  $\{{}^{k-1}\mathbf{M}^*, {}^{k}\mathbf{P}^*, \mathbf{X}^*\} \leftarrow \{{}^{k-1}\mathbf{M}_a, {}^{k}\mathbf{P}_a, \mathbf{X}_a\}$;
else
  $\{{}^{k-1}\mathbf{M}^*, {}^{k}\mathbf{P}^*, \mathbf{X}^*\} \leftarrow \{{}^{k-1}\mathbf{M}_b, {}^{k}\mathbf{P}_b, \mathbf{X}_b\}$;
end if
return $\{{}^{k-1}\mathbf{M}^*, {}^{k}\mathbf{P}^*, \mathbf{X}^*\}$ **;**

5.3.3.1 Classifying Dynamic Object

Instance level object segmentation allows us to separate objects from background. Despite the fact that the proposed algorithm is able to estimate the motions of all the segmented objects, dynamic object classification helps reduce computational cost of the proposed system. This is achieved using scene flow estimation as shown in Figure 5.2 (centre). More in detail, after obtaining camera motion ${}^{k-1}_{k-1}\mathbf{T}_k$, the scene flow vector ${}^{k-1}\mathbf{f}^i$ describing the motion of a 3D point ${}^{k-1}\mathbf{m}^i_{k-1}$ between frame $k-1$ and k, can be calculated as Lv et al. [2018]:

$$ {}^{k-1}\mathbf{f}^i = {}^{k-1}\mathbf{m}^i_{k-1} - ({}^{k-1}_{k-1}\mathbf{T}_k \, {}^{k}\mathbf{m}^i_k) \,. \tag{5.15} $$

Unlike optical flow, the scene flow can directly decide whether the scene structure is moving or not. Ideally, the magnitude of the scene flow vector should be zero for static 3D point. However, noise or error in depth and feature association complicates the situation in real scenarios.

To robustly tackle this, the scene flow magnitude of all the sampled points on each object is computed, and separated into two sets (static and dynamic) via threshold. An object is recognised dynamic if the proportion of 'dynamic' points is above a certain level, otherwise static. Table 5.1 demonstrates the performance of classifying dynamic and static objects using this strategy. Overall, the proposed approach achieves good accuracy among the tested sequences. Notice that, in sequence 20, there are relatively high false negative cases. That is because most

cars throughout the sequence, move slowly (nearly static) due to traffic jams.

Table 5.1: Performance of dynamic/static object classification over virtual KITTI dataset.

Sequence	01	02	06	18	20
Total Detection	1383	150	266	970	2091
Dynamic/Static	117/1266	73/77	257/9	970/0	1494/597
False Positive	3	0	9	0	3
False Negative	6	0	0	57	292

5.3.3.2 Dynamic Object Tracking

Instance-level object segmentation only provides labels frame by frame, therefore objects need to be tracked between frames and their motion models propagated over time. To manage this, optical flow is leveraged to associate point labels in across frames. For that, a finite tracking label set $\mathscr{L} \subset \mathbb{N}$ is introduced and maintained, where $l \in \mathscr{L}$ starts from $l = 1$, when the first moving object appears in the scene. The number of elements in $\mathscr{L}$ increases as more 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 previous frame $k-1$. However, in practice this is affected by the noise, image boundary and occlusions. To overcome this, all the points are assigned with the label that appears most in their correspondences from the last frame. For a dynamic object, if the most frequent label in the previous frame is 0, it means that the object starts to move, appears in the scene at the boundary, or reappears from occlusion. In this case, the object is assigned with a new tracking label.

5.3.3.3 Object Motion Estimation

As mentioned before, objects normally appear in small sizes in the scene, which makes it difficult to get sufficient sparse features to track and estimate their motions robustly. Therefore, the object point set $\mathbf{P}_o$ is densified via sampling every 3^{rd} pixel within object mask in practice. Similar to the ego-motion estimation, an initial object motion model is generated for initialisation using Algorithm 4. The model with most inliers is refined using (5.14) to get the final object motion and the best point matchings.

5.4 Experiments

In this section, experimental results on two public datasets are demonstrated. Virtual KITTI dataset Gaidon et al. [2016] is used for detailed analysis, which provides ground truth of ego/object poses, depth, optical flow and instance level object segmentation. KITTI tracking dataset Geiger et al. [2012] is used to demonstrate the applicability of the proposed algorithm in real life scenarios. A learning-based method, Mask R-CNN He et al. [2017], is adopted to

generate object segmentation in both datasets. The model of this method is trained on COCO dataset Lin et al. [2014], and it is directly used without fine-tuning. A state-of-the-art method, PWC-Net Sun et al. [2018], is used to obtain dense optical flow. The model is trained on FlyingChairs dataset Mayer et al. [2016], and then fine-tuned on Sintel Butler et al. [2012] and KITTI training datasets Geiger et al. [2012]. Feature detection is done using FAST Rosten and Drummond [2006].

Pose change error is used to evaluate the estimated $\mathrm{SE}(3)$ motion, i.e., given ground truth motion $\mathbf{X}$ and estimated $\hat{\mathbf{X}}$, where $\mathbf{X} \in \mathrm{SE}(3)$ can be either camera or object motion. The pose change error is obtained as: $\mathbf{E} = \hat{\mathbf{X}}^{-1}\mathbf{X}$. Translation error E_t is computed as the L_2 norm of translational component in $\mathbf{E}$. Rotation error E_R is measured as the angle in axis-angle representation of rotation part of $\mathbf{E}$. Object velocity error is also evaluated in this work. According to Chirikjian et al. [2017], given an object motion $\mathbf{H}$, the object velocity v can be calculated as:

$$v = ||\mathbf{t} - (\mathbf{I}_3 - \mathbf{R})\,\mathbf{c}||\,, \tag{5.16}$$

where $\mathbf{R}$ and $\mathbf{t}$ are the rotation and translation part of the motion of points in inertial frame. $\mathbf{I}_3$ is identity matrix and $\mathbf{c}$ is centroid of object. Then error of velocity E_v between estimated $\hat{v}$ and ground truth v can be represented as: $E_v = |\hat{v} - v|$. The optical flow is evaluated using end-point error (EPE) Sun et al. [2014], that is defined as the Euclidean distance between estimated and ground truth optical flow vectors,

$$E_{ep} = ||\boldsymbol{\phi}_{est} - \boldsymbol{\phi}_{gt}||\,. \tag{5.17}$$

Table 5.2: Average optical flow end-point error (EPE) of static background and objects in S18-F124-134.

	Static	Obj1	Obj2	Obj3
Object Distance (m)	–	7.52	16.52	24.67
Object Area (%)	–	6.29	0.73	0.29
EPE X-axis (pix)	1.34	0.35	0.34	0.15
EPE Y-axis (pix)	0.27	0.24	0.22	0.18

5.4.1 Virtual KITTI Dataset

This dataset is used to analyse the influence of the optical flow and depth accuracy on the estimation of the ego and object motion. Moving objects appears scatteredly within a sequence, which makes it hard to perform in-depth tests using the whole sequence. Therefore, a representative set that contains multiple moving objects is selected for analysis. The set is part of the Sequence 18 and the frame IDs are between 124 and 134 (denoted as S18-F124-134, with 'Sxx' for sequence index and 'Fxxx-xxx' for frame indices). It contains 10 frames of the

agent car with camera moving forward, and three observed vehicles. Two of them are moving alongside in the left lane, with one closer to the camera and the other farther. The third car is moving upfront and it is furthest from the camera.

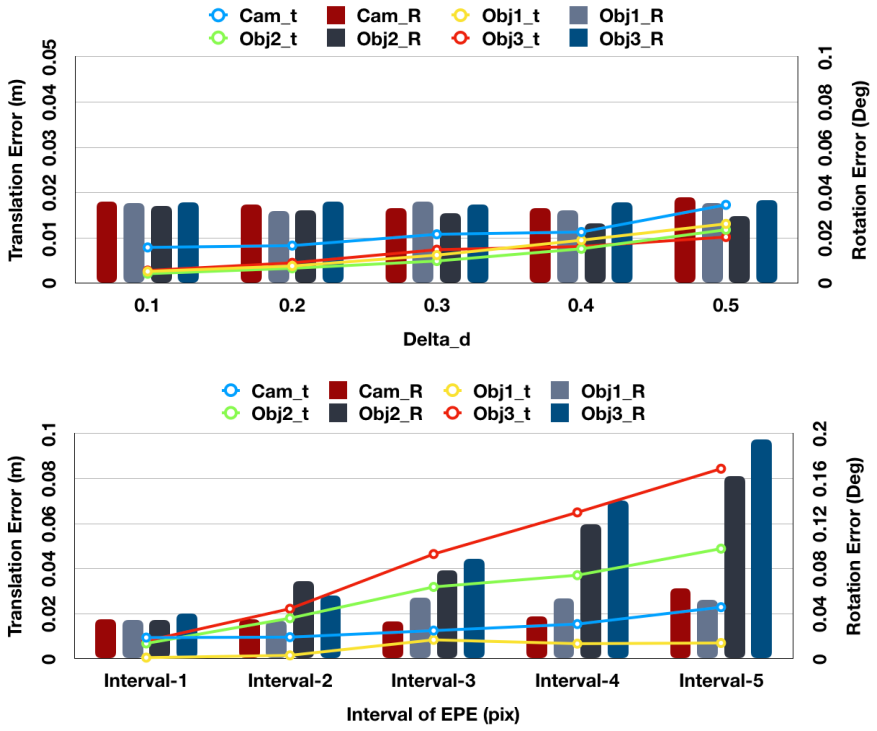


Figure 5.4: **Average error of rigid motion with regard to noise level of depth (top), and to End-point Error of optical flow (bottom).** Curves represent translation error that are corresponding to left-Y axis, and bars represent rotation error that are corresponding to right-Y axis.

Depth: Ground truth depth is corrupted with zero mean Gaussian noise, with σ following standard depth accuracy of a stereo camera system expressed as Chang and Chatterjee [1992]:

$$\sigma = \frac{z^2}{f \cdot b} \cdot \triangle d \,, \tag{5.18}$$

where z is depth, f focal length, b baseline and Δd the disparity accuracy. The baseline is set to $b = 0.5$ m and control Δd to get the noise level of depth. Normally, Δd varies from 0.1 to 0.2 for a standard industrial stereo camera. Figure 5.4 (top) demonstrates the average error of rigid motion over all selected frames. The results indicate that the proposed algorithm is robust to depth noise within reasonable range. The translation error grows gradually with the depth error for both camera and objects, but stays in low range ($E_t < 0.02$m). Rotation error fluctuates slightly but still in low range ($E_R < 0.04$deg). Note that, the camera errors are slightly worse than the object errors for most depth noise levels. As the motions of camera are mainly in translation, we believe this is due to most static points of the background are

distributed more distantly from the camera than the observed objects, and hence higher depth noise is introduced to the static points according to (5.18).

Optical Flow: The ground truth optical flow is corrupted with zero mean Gaussian noise with σ decided by the end-point error (EPE). Table 5.2 demonstrates the average EPE of PWC-Net results for the static and object points among this sequence. Since the errors among static background and objects are different, five intervals are set in increasing order and these average errors are used as the middle value. As an example, for static points, σ_y is set to [0.09 0.18 0.27 0.36 0.45].

As illustrated in Figure 5.4 (bottom) and Table 5.2, the camera and object 1 motion errors are relatively low and stable for different EPEs. However, the motion errors of object 2 and 3 increase reaching nearly 0.09 meter in translation and 0.2 degree in rotation and this is mainly because they are far away from the camera and occupy quite small area of the image ($< 1\%$). Consequently, the object motion estimation is sensitive to optical flow error if the objects are not well distributed in the scene. To avoid unreliable estimation, the proposed system addresses only objects within 25m, and 0.5% image presence.

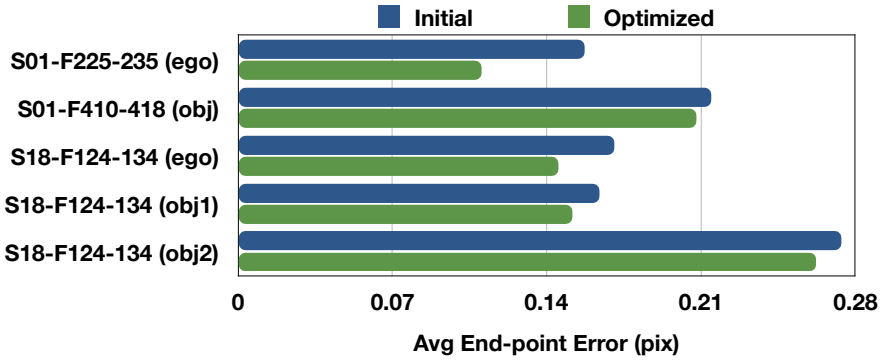


Figure 5.5: **Average end-point error between initial and optimized optical flow, among different tested sets.**

Overall Results: Now overall results without ground truth are shown. Because vKITTI does not provide stereo images, generating depth map from stereo is not available. Instead, the ground truth depth map is used with noise added using (5.18) and $\Delta d = 0.2$.

As the objects in S18-F124-134 are mainly translating, two more sets with obvious rotation is introduced. One of them (S01-F225-235) contains the agent car (camera) turning left into the main street. The other (S01-F410-418) contains static camera observing one car turning left at the crossroads. To prove the effectiveness of jointly optimising motion and optical flow, two methods are set and compared: a baseline method that only optimises for motion (Motion Only) using (5.5) for camera or (5.13) for object, and the improved method that optimises for both motion and optical flow with prior constraint (Joint) using (5.14).

As demonstrated in Table 5.3, optimising for the optical flow jointly with the $\mathrm{SE}(3)$ motions improve the results, with approximately 300% for the camera motion, and 10-20% on object motion. Moreover, the corresponding optical flow error after joint optimisation is also reduced, see Figure 5.5.

Table 5.3: Average error of object motions of different sets.

		Motion only		Joint	
		E_t (m)	E_R (deg)	E_t (m)	E_R (deg)
S01-F225-235	Ego	0.0117	0.0354	**0.0043**	**0.0310**
S01-F410-418	Obj	0.0647	0.2811	**0.0470**	**0.2286**
	Ego	0.0367	0.1012	**0.0052**	**0.0315**
S18-F124-134	Obj1	0.0169	0.1016	**0.0132**	**0.0804**
	Obj2	0.1121	0.2720	**0.1008**	**0.1907**

5.4.2 Real KITTI Dataset

In KITTI tracking dataset, there are 21 sequences with ground truth camera and object poses. For camera motion, the ego-motion error is calculated over all the sequences (12 in total) except the ones that the camera is not moving at all. The results of a state-of-the-art method ORB-SLAM2 Mur-Artal and Tardós [2017] is also generated for comparison. Figure 5.6 illustrates

Table 5.4: Average ego-motion error over 12 tested sequences.

	PROPOSED	ORB-SLAM2
E_t (m)	**0.0642**	0.0730
E_R (deg)	**0.0573**	0.0622

the camera trajectory results on four representative sequences. Compared with ORB-SLAM2, the proposed method is able to produce smooth trajectories that are more consistent with the ground truth, given the fact that the proposed method conducts only frame-by-frame tracking, while ORB-SLAM2 integrates more complex modules, such as local map tracking and local bundle adjustment. In particular, the result of Seq. 20 in Figure 5.6 (right) shows that ORB-SLAM2 obtains bad estimates in first half of the sequence, mainly because this part contains dynamic scenes of traffic on the highway. Nevertheless, the proposed algorithm is robust against this case. Table 5.4 shows average motion error over all the 12 tested sequences. The results prove the improved performance over ORB-SLAM2.

For object motion, the results of object velocity error among 9 sequences containing considerable number of moving objects is demonstrated, since vehicle velocity is important information for autonomous and safety driving applications. As demonstrated in Table 5.5, the number of tracks refers to how many frames those objects are being tracked. This indicates the proposed pipeline is able to simultaneously and robustly track multiple moving objects for long distances. The average velocity error E_v is computed over all the tracks among one or all objects (see the second row in Table 5.5). Overall, the proposed method gets around 2-5km/h error, which is considerably accurate for the velocity ranging from 11-55km/h.

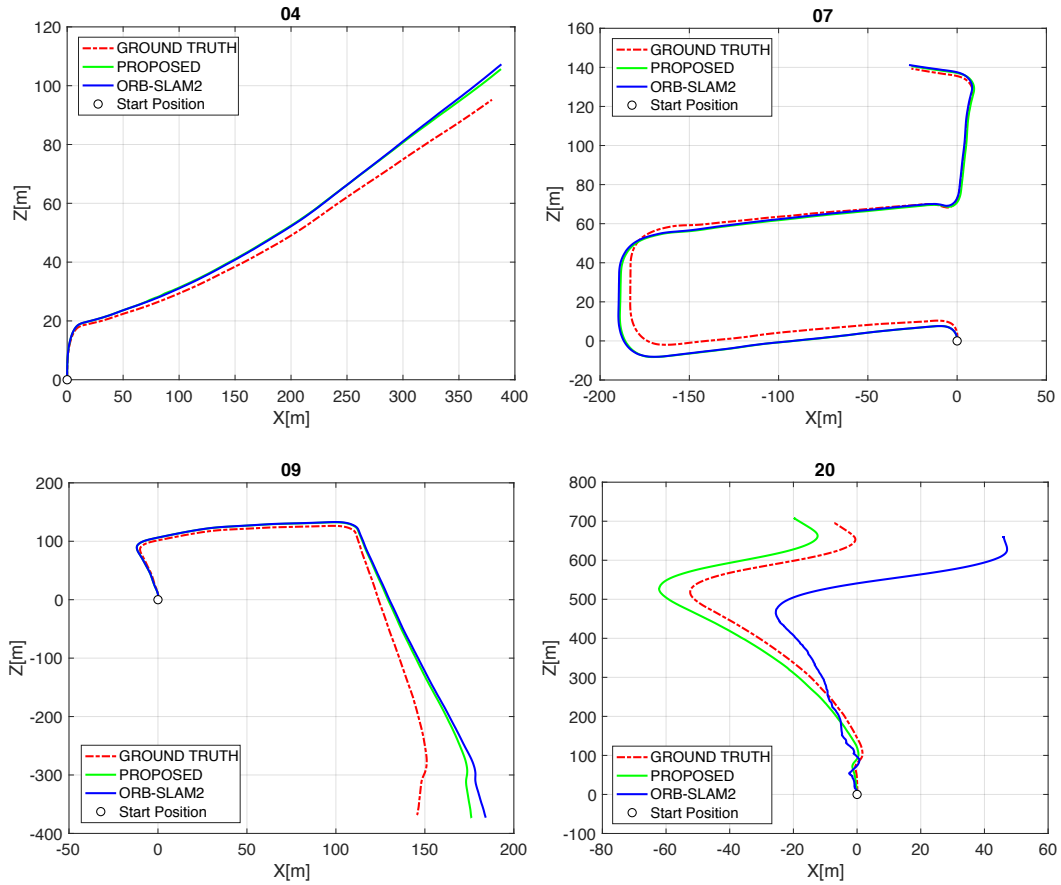


Figure 5.6: **Top view of camera trajectories of four tested KITTI sequences, compared to ORB-SLAM2 and ground truth.**

Figure 5.7 demonstrates qualitative results of four of the KITTI sequences. The results indicate that the proposed system is able to output the camera poses and dynamic tracks of every detected moving object in the scene. Specifically, results of sequence 00 and 03 on the left and the top right, both show the camera tracking two objects for long distance, which corresponds to the results in Table 5.5. In sequence 04 (bottom), it shows the agent car turning right, with the camera observing multiple cars upfront and on the other side of the street. A similar case can be observed in the results of sequence 06 (bottom right).

The computational cost of the proposed algorithm is around 6fps when run on an i7 2.6Ghz laptop. The main cost lies in denser points tracking on multiple objects. This can be improved by employing parallel implementation to achieve real-time performance.

5.5 Conclusion

In this chapter, a novel framework to simultaneously track camera and multiple object motions is presented. The proposed framework detects moving objects via combining instance-level object segmentation and scene flow, and tracks them over frames using optical flow. The $\mathrm{SE}(3)$

Table 5.5: Average velocity error of sequences with multiple moving objects.

Sequence	00		01	02	03		04	05	06	18	20
Detected Objects	van	cyclist	5 cars	6 cars	wagon	suv	20 cars	12 cars	10 cars	18 cars	46 cars
Num. of Tracks	44	90	76	39	44	49	109	57	137	431	489
Avg. Velocity (km/h)	18.92	16.06	14.07	34.29	54.44	52.23	30.12	45.22	32.82	20.95	11.73
Avg. Error E_v (km/h)	3.04	2.01	2.02	5.22	2.70	2.63	5.13	5.52	4.26	1.96	2.18

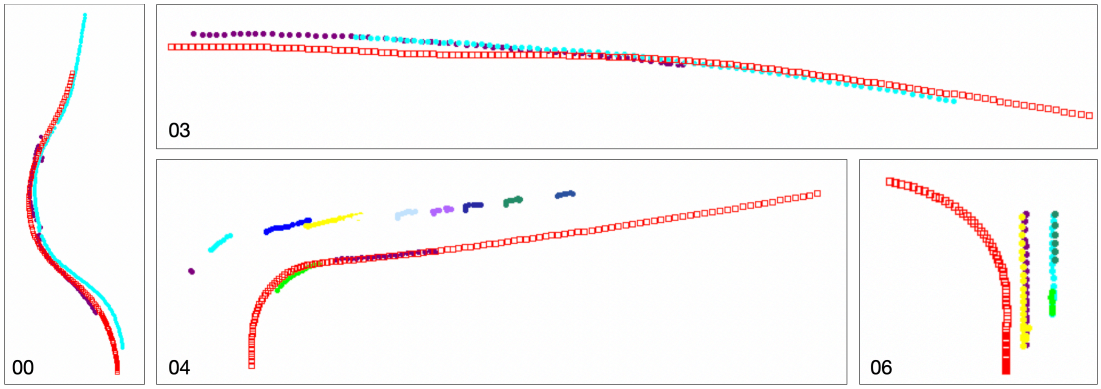


Figure 5.7: **Qualitative results of camera and objects trajectories of four tested KITTI sequences.** Red square refers to camera and colour dots refer to objects.

motions of the objects, as well as the camera are optimised jointly with the optical flow in a unified formulation. The proposed approach is carefully evaluated on virtual KITTI dataset, and demonstrate its effectiveness. Furthermore, extensive testing is performed on the real KITTI dataset. The results show that the proposed method is able to obtain robust and accurate camera trajectories in dynamic scene, and track the velocity of objects with high accuracy. Further work will integrate the proposed motion estimation within a SLAM framework.

The proposed method, however, is limited by relying on learning-based techniques to recognize and segment potential moving objects, which can only handle pre-trained object categories. This can be a problem when objects are out of the object categories in training set or the conditions are too far from the training setup. As the proportion of the non-segmented moving objects increases, the possibility of system's failure to track camera becomes higher. A possible solution is to integrate IMU for robust camera pose recovery, which would be an interesting direction for future work.

Chapter 6

VDO-SLAM: A Visual Dynamic Object-aware SLAM System

Combining Simultaneous Localisation and Mapping (SLAM) estimation and dynamic scene modelling can highly benefit robot autonomy in dynamic environments. Robot path planning and obstacle avoidance tasks rely on accurate estimations of the motion of dynamic objects in the scene.

This chapter presents VDO-SLAM, a visual object-aware dynamic SLAM system that exploits semantic information to enable motion estimation of rigid objects in the scene without any prior knowledge of the objects shape or motion models. The proposed system is developed based on the multi-body visual odometry introduced in Chapter 5 as an extension, aiming to integrate dynamic and static structures in the environment into a unified estimation framework and ensure more accurate robot pose and spatio-temporal map estimation. A simple but effective way to extract velocity estimates from object pose change of moving objects in the scene is also provided as an important functionality for navigation in complex dynamic environments. The performance of the proposed system is demonstrated on a number of real indoor and outdoor datasets. Results show consistent and substantial improvements over state-of-the-art algorithms.

6.1 Motivation

The ability of a robot to build a model of the environment, often called map, and to localise itself within this map is a key factor in enabling autonomous robots to operate in real world environments. Creating these maps is achieved by fusing multiple sensor measurements of the environment into a consistent representation using estimation techniques such as Simultaneous Localisation And Mapping (SLAM). SLAM is a mature research topic and have already revolutionised a wide range of applications from mobile robotics, inspection, entertainment and film production to exploration and monitoring of natural environments, amongst many others. However, most of the existing solutions to SLAM rely heavily on the assumption that the environment is predominantly static.

The conventional techniques 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 (Hahnel et al. [2002, 2003]; Wolf and Sukhatme [2005]; Zhao et al. [2008]; Bescos and Neira

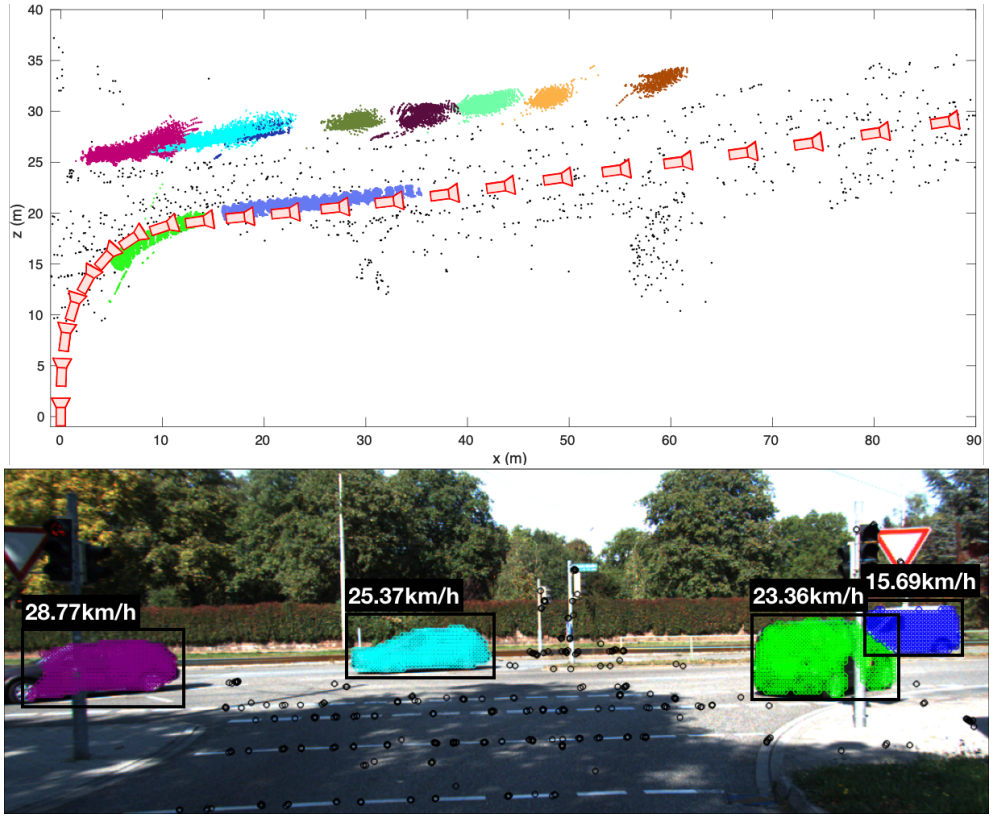


Figure 6.1: **Results of the proposed VDO-SLAM system.** (Top) A full map including camera trajectory, static background and moving object structure. (Bottom) Detected points on static background and object body, and estimated object speed. Black circles represents static points, and each object is assigned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

[2018]), or detect moving objects and track them separately using traditional multi-target tracking approaches (Wang et al. [2003]; Miller and Campbell [2007]; Rogers et al. [2010]; Kundu et al. [2011]). 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 the community to challenge the scene rigidity assumption, also known as the static world assumption, that underpins most existing open-source SLAM algorithms. In this chapter, we redefine the term "mapping" in SLAM to be concerned with a *spatiotemporal representation of the world*, as opposed to the concept of a static map that has long been the emphasis of classical SLAM algorithms. Our approach focuses on accurate motion estimation of dynamic entities in the environment including the robot and other moving objects in the scene, this information being of high relevance in the context of robot path planning and navigation in dynamic environments.

Literature of object motion estimation is mainly represented as image-based optical flow (Yamaguchi et al. [2014]; Sun et al. [2010, 2018]; Ilg et al. [2017]) or 3D based scene flow es-

timation (Vogel et al. [2013]; Menze and Geiger [2015]; Liu et al. [2019]; Jiang et al. [2019]). Optical flow records the scene motion by estimating the velocities associated with the movement of brightness patterns in an image. As it is characterized on image plane, optical flow involves both camera and scene motion, which may fail in degenerate cases where the scene motion is close to the camera motion. Scene flow, on the other hand, describes the 3D motion field of a scene that is completely independent of the camera motion. As both methods are only concerned with the motion of 2D image pixels or 3D points in the scene, they only describe the linear translation of a pixel/point, lack the concept of collective behaviour of points motion on a rigid object and fail to describe the full $\mathrm{SE}(3)$ motion of objects in the scene. Describing the $\mathrm{SE}(3)$ motion of dynamic objects in the scene has not only proven to be more accurate and robust as it incorporates the concept of rigidity of the moving objects, but also more computationally efficient as it eliminates the need to estimate for a different motion for each individual point as is the case in optical and scene flow estimation. These unique features make $\mathrm{SE}(3)$ object motion a perfect choice for solving a dynamic SLAM problem. In particular, $\mathrm{SE}(3)$ object motion can be easily integrated into a SLAM optimisation problem, as an additive term to the cost function, which in turn helps to robustly track objects and accurately estimate their motions.

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 information (raw and 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.

Feature tracking has been more reliable and robust with the advances in deep learning to provide algorithms that can reliably estimate the 2D optical flow associated with the apparent motion of every pixel on an image in a dense manner. A task that is particularly important for data association and that has been otherwise challenging in dynamic environments using classical feature tracking methods.
Other primitives such as lines and planes (de la Puente and Rodríguez-Losada [2014]; Kaess [2015]; Henein et al. [2017]; Hsiao et al. [2017]) or even objects (Mu et al. [2016]; Salas-Moreno et al. [2013]; Yang and Scherer [2019]) have been considered in order to provide richer map representations. Semantic information can provide important prior information in identifying and tracking dynamic objects in the scene (Wang et al. [2007]; Gálvez-López et al. [2016]). Advances in deep learning have provided algorithms that can reliably detect and segment classes of objects at almost real time (Girshick et al. [2018]; He et al. [2018]). Despite recent developments in vision-based object detection and segmentation, the visual SLAM community has not yet fully exploited such information Nicholson et al. [2018a]. To incorporate such information in existing geometric SLAM algorithms, either a dataset of 3D-models of every object in the scene must be available a priori (Salas-Moreno et al. [2013]; Gálvez-López et al. [2016]) or the front end must explicitly provide object pose information in addition to detection and segmentation (Milan et al. [2016]; Byravan and Fox [2017]; Wohlhart and Lepetit [2015]) 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 algorithms without relying on additional pose estimation or object model priors, an algorithm that operates at feature-level with the awareness of an object concept.

In this chapter, a novel feature-based stereo/RGB-D dynamic SLAM system, VDO-SLAM, is proposed. The system is developed upon the previous multi-body VO framework as the tracking front-end, by improving the robustness of feature and object tracking, and extending it with a novel graph-based back-end to simultaneously localise the robot, map the static and dynamic structure, and track motions of rigid objects in the scene. In summary, the contributions of this work are:

- a novel formulation to model dynamic scenes in a unified estimation framework over robot poses, static and dynamic 3D points, and object motions.
- accurate estimation for $\mathrm{SE}(3)$ motion of dynamic objects that outperforms state-of-the-art algorithms, as well as a way to extract objects' velocity in the scene,
- 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.

Arguably, 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 and linear velocity extraction of every rigid object in the scene, and be demonstrable in real-world outdoor scenarios (see Figure 6.1). The performance of the proposed algorithm is demonstrated on real datasets, showing the capability 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.

6.2 Methodology

In this section, a novel dynamic SLAM algorithm of jointly estimating the camera pose and object motion, as well as the environment structure in a unified framework is explained. The framework is represented as a factor graph, and formulated as a non-linear least squares optimization problem.

In the tracking component of the proposed system, shown in Figure 6.4, the cost function chosen to estimate the camera pose and object motion (described in Section 5.2.1 and Section 5.2.2) is associated with the 3D-2D re-projection error and is defined on the image plane. Since the noise is better characterised on image plane, this yields more accurate results for camera localisation (Nistér et al. [2004]). Moreover, based on this error term, a novel formulation to jointly optimise the optical flow along with the camera pose and the object motion is introduced to ensure a robust tracking of points (described in Section 5.2.3). In the mapping module, a 3D point error cost function is utilised to ensure best results of 3D structure and object motions as described in Section 6.2.2.

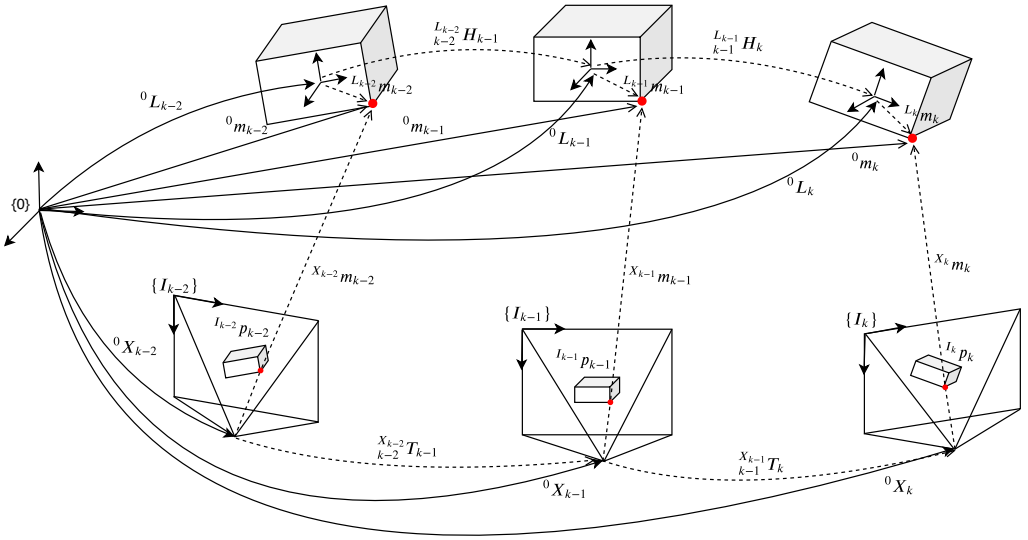


Figure 6.2: **Notation and coordinate frames.** Solid curves represent camera and object poses in inertial frame; ${}^0\mathbf{X}$ and ${}^0\mathbf{L}$ respectively, and dashed curves their respective motions in body-fixed frame. Solid lines represent 3D points in inertial frame, and dashed lines their projections into image and camera frames.

6.2.1 Background and Notation

Coordinate Frames: Let ${}^0\mathbf{X}_k, {}^0\mathbf{L}_k \in \mathrm{SE}(3)$ be the robot/camera and the 3D object pose respectively, at time k in a global reference frame 0, with $k \in \mathscr{T}$ the set of time steps. Note that calligraphic capital letters are used in the notation to represent sets of indices. Figure 6.2 illustrates these pose transformations as solid curves.

Points: 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 $i \in \mathscr{M}$ the set of points. A point in robot/camera frame is denoted as:

$$ {}^{X_k}\mathbf{m}_k^i = {}^0\mathbf{X}_k^{-1}\, {}^0\mathbf{m}_k^i \,. \tag{6.1} $$

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 ${}^{\mathbf{I}_k}\mathbf{p}_k^i = \left[u^i, v^i, 1\right] \in \mathbb{E}^2$ be the pixel location on image $\mathbf{I}_k$ corresponding to the homogeneous 3D point ${}^{\mathbf{X}_k}\mathbf{m}_k^i$, which is obtained via the projection function $\pi(\cdot)$ as follows:

$$ {}^{I_k}\mathbf{p}_k^i = \pi({}^{X_k}\mathbf{m}_k^i) = \mathbf{K}\, {}^{X_k}\mathbf{m}_k^i \,, \tag{6.2} $$

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

by:

$$ {}^{\mathbf{I}_k}\boldsymbol{\phi}^i = {}^{\mathbf{I}_k}\tilde{\mathbf{p}}^i_k - {}^{\mathbf{I}_{k-1}}\mathbf{p}^i_{k-1} \,. \tag{6.3} $$

Here ${}^{\mathbf{I}_k}\tilde{\mathbf{p}}^i_k$ is the correspondence of ${}^{\mathbf{I}_{k-1}}\mathbf{p}^i_{k-1}$ in $\mathbf{I}_k$. Note that, the same notation is overloaded to represent the 2D pixel coordinates $\in \mathbb{R}^2$. In this work, optical flow is used to find correspondences between consecutive frames.

3D Point Motion on Object Body: The motion of a 3D point on a rigid body in inertial frame is given by ${}^{0}_{k-1}\mathbf{H}_k \in \mathrm{SE}(3)$, which is derived in Section 5.2.2 to express the rigid object pose change in terms of the points that reside on the object without the need to know the object 3D pose. Here ${}^{0}_{k-1}\mathbf{H}_k$ represents the object point motion in global reference frame; for the remainder of this document, this quantity is referred as the object pose change or the object motion for ease of reading.

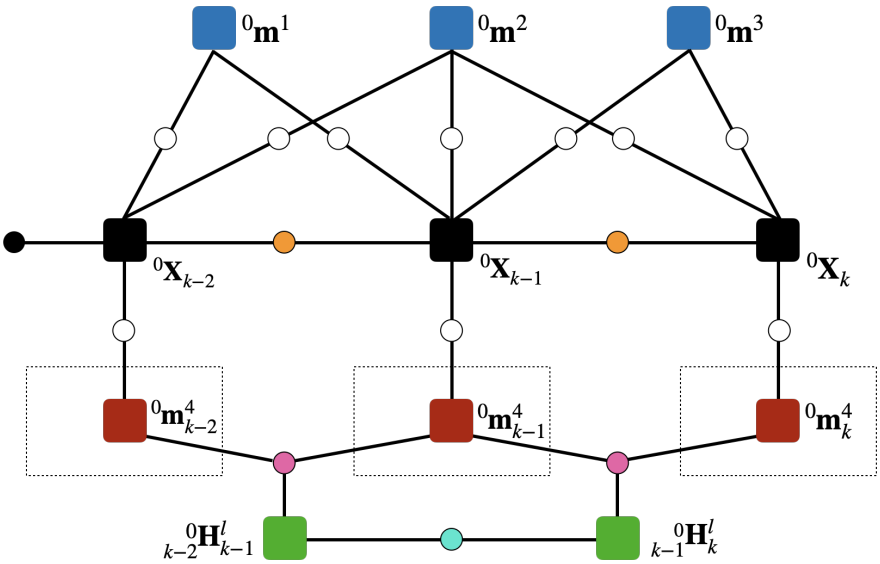


Figure 6.3: **Factor graph representation of an object-aware SLAM with a moving object.** Black squares represent the camera poses at different time steps, blue squares represent five static points, red squares represent the same dynamic point on an object (dashed box) at different time steps and green squares 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 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.

6.2.2 Graph Optimisation

The proposed approach formulates the dynamic SLAM as a graph optimisation problem, to estimate the camera poses and object motions, and build a global consistent map including

static and dynamic structure. The dynamic SLAM problem is modelled as a factor graph as the one demonstrated in Figure 6.3. The factor graph formulation is highly intuitive and has the advantage that allows for efficient implementations of batch (Dellaert and Kaess [2006]; Agarwal et al. [2012]; Kümmerle et al. [2011]) and incremental (Kaess et al. [2011]; Polok et al. [2013]; Ila et al. [2017]) solvers.

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 and the object smooth motion observations. The 3D point measurement model error $\mathbf{e}_{i,k}({}^0\mathbf{X}_k, {}^0\mathbf{m}_k^i)$ is defined as:

$$\mathbf{e}_{i,k}({}^0\mathbf{X}_k, {}^0\mathbf{m}_k^i) = {}^0\mathbf{X}_k^{-1}\,{}^0\mathbf{m}_k^i - \mathbf{z}_k^i\,. \tag{6.4}$$

Here $\mathbf{z} = \{\mathbf{z}_k^i \mid i \in \mathscr{M}, k \in \mathscr{T}\}$ is the set of all 3D point measurements at all time steps, with cardinality n_z and $\mathbf{z}_k^i \in \mathbb{R}^3$. The 3D point measurement factors are shown as white circles in Figure 6.3.
The tracking component of the system provides a high-quality ego-motion via 3D-2D error minimization, which can be used as an odometry measurement to constrain camera poses in the graph. The visual odometry model error $\mathbf{e}_k({}^0\mathbf{X}_{k-1}, {}^0\mathbf{X}_k)$ is defined as:

$$\mathbf{e}_k({}^0\mathbf{X}_{k-1}, {}^0\mathbf{X}_k) = ({}^0\mathbf{X}_{k-1}^{-1}\,{}^0\mathbf{X}_k)^{-1}\,{}^{X_{k-1}}_{k-1}\mathbf{T}_k\,, \tag{6.5}$$

where $\mathbf{T} = \{{}^{X_{k-1}}_{k-1}\mathbf{T}_k \mid k \in \mathscr{T}\}$ 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 Figure 6.3.
The motion model error of points on dynamic objects $\mathbf{e}_{i,l,k}({}^0\mathbf{m}_k^i, {}^0_{k-1}\mathbf{H}_k^l, {}^0\mathbf{m}_{k-1}^i)$ is defined as:

$$\mathbf{e}_{i,l,k}({}^0\mathbf{m}_k^i, {}^0_{k-1}\mathbf{H}_k^l, {}^0\mathbf{m}_{k-1}^i) = {}^0\mathbf{m}_k^i - {}^0_{k-1}\mathbf{H}_k^l\,{}^0\mathbf{m}_{k-1}^i\,. \tag{6.6}$$

The motion of all points on a detected rigid object l are characterised by the same pose transformation ${}^0_{k-1}\mathbf{H}_k^l \in \mathrm{SE}(3)$ given by (5.11) and the corresponding factor, shown as magenta circles in Figure 6.3, is a ternary factor which is called the *motion model of a point on a rigid body*.
It has been proven that incorporating prior knowledge about the motion of objects in the scene is highly valuable in dynamic SLAM (Wang et al. [2007]; Henein et al. [2020]). 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, the proposed work introduces smooth motion factors to minimise the change in consecutive object motions, with the error term defined as:

$$\mathbf{e}_{l,k}({}^0_{k-2}\mathbf{H}_{k-1}^l, {}^0_{k-1}\mathbf{H}_k^l) = {}^0_{k-2}\mathbf{H}_{k-1}^l{}^{-1}\,{}^0_{k-1}\mathbf{H}_k^l. \tag{6.7}$$

The object smooth motion factor $\mathbf{e}_{l,k}({}^0_{k-2}\mathbf{H}_{k-1}^l, {}^0_{k-1}\mathbf{H}_k^l)$ is used to minimise the change between the object motion at consecutive time steps and is shown as cyan circles in Figure 6.3.

Let $\boldsymbol{\theta}_M = \{{}^0\mathbf{m}_k^i \mid i \in \mathscr{M}, k \in \mathscr{T}\}$ be the set of all 3D points. Parameterise the $\mathrm{SE}(3)$ camera pose ${}^0\mathbf{X}_k$ by elements ${}^0\mathbf{x}_k \in \mathrm{se}(3)$ the Lie-algebra of $\mathrm{SE}(3)$:

$${}^0\mathbf{X}_k = \exp({}^0\mathbf{x}_k)\,, \tag{6.8}$$

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

$$ {}^0_{k-1}\mathbf{H}^l_k = \exp({}^0_{k-1}\mathbf{h}^l_k)\,, \tag{6.9} $$

and define $\boldsymbol{\theta}_H = \{{}^0_{k-1}\mathbf{h}^{l\,\vee}_k \mid k \in \mathscr{T}, l \in \mathscr{L}\}$ as the set of all object motions, with ${}^0_{k-1}\mathbf{h}^{l\,\vee}_k \in \mathbb{R}^6$ and $\mathscr{L}$ 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 $\mathbf{X}$ and $\mathbf{H}$, *i.e.*, substituting (6.8) in (6.4) and (6.5), and substituting (6.9) in (6.6) and (6.7), the solution of minimizing all the least squares costs is given by:

$$ \begin{aligned} \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \Big\{ & \sum_{i,k}^{n_z} \rho_h\big(\mathbf{e}^\top_{i,k}({}^0\mathbf{x}_k, {}^0\mathbf{m}^i_k)\,\Sigma_z^{-1}\,\mathbf{e}_{i,k}({}^0\mathbf{x}_k, {}^0\mathbf{m}^i_k)\big) \\ & + \sum_{k}^{n_o} \rho_h\big(\log(\mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k))^\top\,\Sigma_o^{-1}\,\log(\mathbf{e}_k({}^0\mathbf{x}_{k-1}, {}^0\mathbf{x}_k))\big) \\ & + \sum_{i,l,k}^{n_g} \rho_h\big(\mathbf{e}^\top_{i,l,k}({}^0\mathbf{m}^i_k, {}^0_{k-1}\mathbf{h}^l_k, {}^0\mathbf{m}^i_{k-1})\,\Sigma_g^{-1}\,\mathbf{e}_{i,l,k}({}^0\mathbf{m}^i_k, {}^0_{k-1}\mathbf{h}^l_k, {}^0\mathbf{m}^i_{k-1})\big) \\ & + \sum_{l,k}^{n_s} \rho_h\big(\log(\mathbf{e}_{l,k}({}^0_{k-2}\mathbf{h}^l_{k-1}, {}^0_{k-1}\mathbf{h}^l_k))^\top\,\Sigma_s^{-1}\,\log(\mathbf{e}_{l,k}({}^0_{k-2}\mathbf{h}^l_{k-1}, {}^0_{k-1}\mathbf{h}^l_k))\big)\Big\}\,, \end{aligned} \tag{6.10} $$

where Σ_z is the 3D point measurement noise covariance matrix, Σ_o is the odometry noise covariance matrix, Σ_g is the motion noise covariance matrix with n_g the total number of ternary object motion factors, and Σ_s the smooth motion covariance matrix, with n_s the total number of smooth motion factors. The non-linear least squares problem in (6.10) is solved using Levenberg-Marquardt method.

6.3 System

In this section, a novel object-aware dynamic SLAM system is presented, which is able to robustly estimates both camera and object motions, along with the static and dynamic structure of the environment. The full system overview is shown in Figure 6.4, and is composed of three main modules: image pre-processing, tracking and mapping.

The input to the system is stereo or RGB-D images. For stereo images, the depth information is extracted by applying the stereo depth estimation method described in Yamaguchi et al. [2014] 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 exploit image-based semantic information, a single image depth estimation is applied to achieve depth information from monocular camera. The proposed 'learning-based monocular' system is monocular in the sense that only RGB images are used as input to the system, while the estimation problem is still formulated using RGB-D data, where the depth is obtained using single image depth estimation.

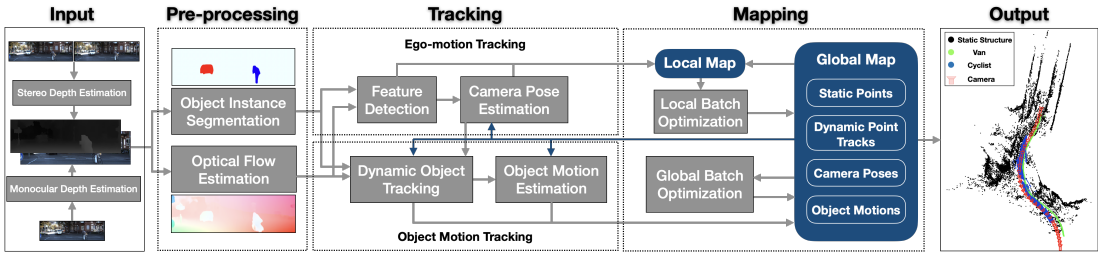


Figure 6.4: **Overview of the proposed VDO-SLAM system.** Input images are first pre-processed to generate instance-level object segmentation and dense optical flow. These are then used to track features on both static background structure and dynamic objects. Camera poses and object motions initially 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.

6.3.1 Pre-processing

As the proposed system in this chapter is an extension of the multi-motion visual odometry pipeline in Chapter 5, it shares the same image pre-processing component as described in Section 5.3.1.

6.3.2 Tracking

The tracking component includes two modules; the camera ego-motion tracking and the object motion tracking.

6.3.2.1 Feature Detection

To achieve fast camera pose estimation, a *sparse* set of corner features is detected and tracked with optical flow. Different from the multi-motion VO pipeline, where features are only matched between two consecutive frames, in the new proposed tracking module, inlier feature points that fit the estimated camera motion in each frame are saved into the map, and continuously used to track correspondences in the coming frames. New features are detected and added, if the number of inlier tracks falls below a certain level. These sparse features are detected and tracked on static background, *i.e.*, image regions excluding the segmented objects.

6.3.2.2 Camera Pose Estimation

The camera pose can be computed using (5.5) for all detected 3D-2D static point correspondences (Section 5.2.1). 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 (Ke and Roumeliotis [2017]) algorithm with RANSAC. The motion model that generates most inliers

is then selected for initialisation. As propagating motion is not available during initialization, the model generated by P3P is used.

6.3.2.3 Dynamic Object Tracking

The process of dynamic object tracking consists of two steps. In the first step, segmented objects are classified into static and dynamic. Then the dynamic objects are associated across pairs of consecutive frames. This process stays the same as described in the multi-motion tracking front-end in Chapter 5. More details can be found in Section 5.3.3.

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. To overcome this, denser object points are used by sampling every third point within an object mask, and tracked across frames. Similar to the camera pose estimation, only inlier points that fit the estimated object motion 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. The same method that is discussed in Section 6.3.2.2 is used 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

A local map is maintained and updated in the system. 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. Only the camera poses and static structure is optimized 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, 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 6.2.2. 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 illustrated in Figure 6.4 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. See Figure 6.9.

6.4 Experiments

The proposed method is evaluated in terms of camera motion, object motion and velocity, as well as object tracking performance. The evaluation is done on the Oxford Multi-motion Dataset Judd and Gammell [2019] for indoor, and KITTI Tracking dataset Geiger et al. [2013] for outdoor scenarios. Results of VDO-SLAM is also compared to the ones generated by two state-of-the-art methods, MVO Judd et al. [2018] and CubeSLAM Yang and Scherer [2019], to prove the better performance of the proposed system.

6.4.1 System Setup

A learning-based instance-level object segmentation method, Mask R-CNN He et al. [2018], is adopted to generate object segmentation masks. The model of this method is trained on COCO dataset Lin et al. [2014], and is directly used in this work without any fine-tuning. For dense optical flow, a state-of-the-art method called PWC-Net Sun et al. [2018] is used. The model is trained on FlyingChairs dataset Mayer et al. [2016], and then fine-tuned on Sintel Butler et al. [2012] and KITTI training datasets Geiger et al. [2012]. To generate depth maps for a 'monocular' version of the proposed system, a learning-based monocular depth estimation method (MonoDepth2 Godard et al. [2019]) is applied. The model is trained on Depth Eigen split Eigen et al. [2014] excluding the tested data in this chapter. Feature detection is done using FAST Rosten and Drummond [2006] implemented in Rublee et al. [2011]. All the above methods are applied using the default parameters.

6.4.2 Error Metrics

A pose change error metric is used to evaluate the estimated $\mathrm{SE}(3)$ motion, *i.e.*, given a ground truth motion transform $\mathbf{T}$ and a corresponding estimated motion $\hat{\mathbf{T}}$, where $\mathbf{T} \in \mathrm{SE}(3)$ could be either a camera relative pose or an object motion. The pose change error is computed as: $\mathbf{E} = \hat{\mathbf{T}}^{-1}\mathbf{T}$. The translational error E_t is computed as the L_2 norm of the translational component of $\mathbf{E}$. The rotational error E_r is calculated as the angle of rotation in an axis-angle representation of the rotational component of $\mathbf{E}$. For different camera time steps and different objects in a sequence, the root mean squared error (RMSE) is computed for camera poses and object motions, respectively. The object pose change in body-fixed frame is obtained by transforming the pose change ${}_{k-1}^{0}\mathbf{H}_k$ in the inertial frame into the body frame using the object pose ground-truth

$$ {}_{k-1}^{L_{k-1}}\mathbf{H}_k = {}^{0}\mathbf{L}_{k-1}^{-1}\ {}_{k-1}^{0}\mathbf{H}_k\ {}^{0}\mathbf{L}_{k-1}. \tag{6.11} $$

The object speed error is also evaluated in this work. The linear velocity of a point on the object, expressed in the inertial frame, can be estimated by applying the pose change ${}_{k-1}^{0}\mathbf{H}_k$ and taking the difference

$$ \begin{aligned} \mathbf{v} \approx {}^{0}\mathbf{m}_k^i - {}^{0}\mathbf{m}_{k-1}^i &= \left({}_{k-1}^{0}\mathbf{H}_k - \mathbf{I}_4\right){}^{0}\mathbf{m}_{k-1}^i \\ &= {}_{k-1}^{0}\mathbf{t}_k - \left(\mathbf{I}_3 - {}_{k-1}^{0}\mathbf{R}_k\right){}^{0}\mathbf{m}_{k-1}^i. \end{aligned} \tag{6.12} $$

To get a more reliable measurement, all points are used to calculate and take the average. Define $\mathbf{c}_{k-1} := \frac{1}{n}\sum \mathbf{m}_{k-1}^i$ for all points on an object at time $k-1$. Then

$$ \begin{aligned} \mathbf{v} &\approx \frac{1}{n}\sum_{i=1}^{n}\left({}_{k-1}^{0}\mathbf{t}_k - \left(\mathbf{I}_3 - {}_{k-1}^{0}\mathbf{R}_k\right){}^{0}\mathbf{m}_{k-1}^i\right) \\ &= {}_{k-1}^{0}\mathbf{t}_k - \left(\mathbf{I}_3 - {}_{k-1}^{0}\mathbf{R}_k\right)\mathbf{c}_{k-1}. \end{aligned} \tag{6.13} $$

Then the speed error E_s between the estimated $\hat{\mathbf{v}}$ and the ground truth $\mathbf{v}$ velocities can be calculated as: $E_s = |\hat{\mathbf{v}}| - |\mathbf{v}|$. For different objects tracked over temporal frames, the RMSE is calculated as an error metric.

6.4.3 Oxford Multi-motion Dataset

The recent Oxford Multi-motion Dataset Judd and Gammell [2019] 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. Since results of real driving scenarios are evaluated on KITTI dataset, only the swinging boxes sequences are chosen for evaluation. Table 6.1 shows results compared to the state-of-the-art MVO Judd et al. [2018]. As MVO is a visual odometry system without global refinement, the batch optimisation module is switched off in proposed system for fair comparison. The pose change error metrics described in Section 6.4.2 are used for evaluation.

Overall, the proposed method achieves better accuracy in 7 out of 10 error indexes for cam-

Table 6.1: Comparison versus MVO Judd et al. [2018] 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	**0.7709**	**0.0112**	1.1948	0.0314
Top-left Swinging Box	**1.1889**	**0.0207**	1.4553	0.0288
Top-right Swinging and rotating Box	**0.7631**	0.0132	0.8992	**0.0130**
Bottom-left Swinging Box	**0.9153**	**0.0149**	1.4949	0.0261
Bottom-right Rotating Box	0.8469	0.0192	**0.7815**	**0.0115**

era pose estimation and motion estimation of the 4 moving boxes. In particular, our proposed method achieves better accuracy in the estimation of camera pose (35%) and motion of the swinging boxes, top-left (15%) and bottom-left (40%). We obtain slightly higher errors when there is spinning rotational motion of the object observed, in particular the top-right swinging and rotating box (in translation only), and the bottom-right rotating box. We believe that this is due to using an optical flow algorithm that is not well optimised for self-rotating objects. 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.

An illustrative result of the map output of the proposed algorithm on Oxford Multi-motion Dataset is shown in Figure 6.5. 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 in purple in Figure 6.5.

6.4.4 KITTI Tracking Dataset

The KITTI Tracking Dataset (Geiger et al. [2013]) contains 21 sequences in total with ground truth information about the 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. Note that, as only rotation around Y-axis is provided in the ground truth object poses, we assign zeros to the other two axes for the convenience of full motion evaluation.

6.4.4.1 Camera and Object Motion

Table 6.2 demonstrates results of both camera and object motion estimation in 9 sequences, compared to CubeSLAM Yang and Scherer [2019], with results of 5 sequences provided by the authors. An initial attempt to evaluate CubeSLAM with the default provided parameters was conducted, however errors were much higher, and hence only results of the sequences provided

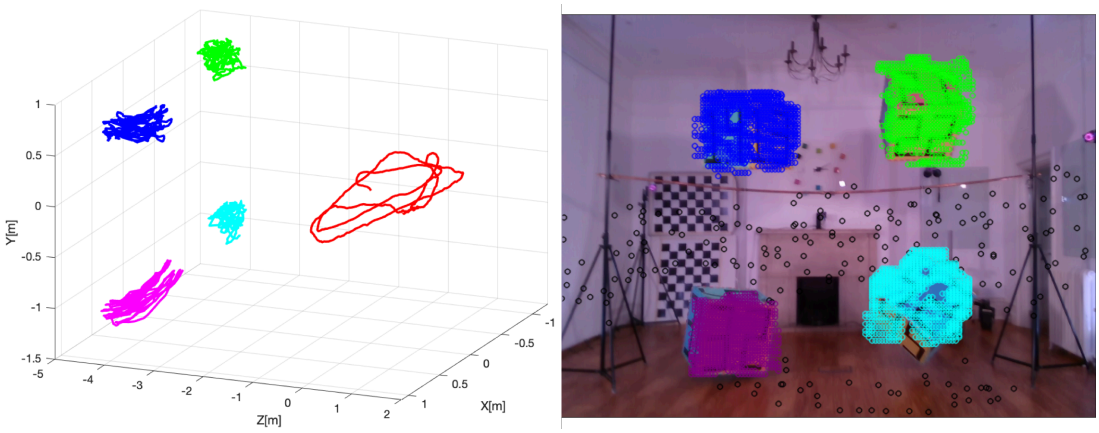


Figure 6.5: **Qualitative results of the proposed method on Oxford Multi-motion Dataset.** (Left) The 3D map including camera trajectory, static structure and tracks of dynamic points. (Right) Detected points on static background and object body. Black colour corresponds to static points and features on each object are shown in a different colour.

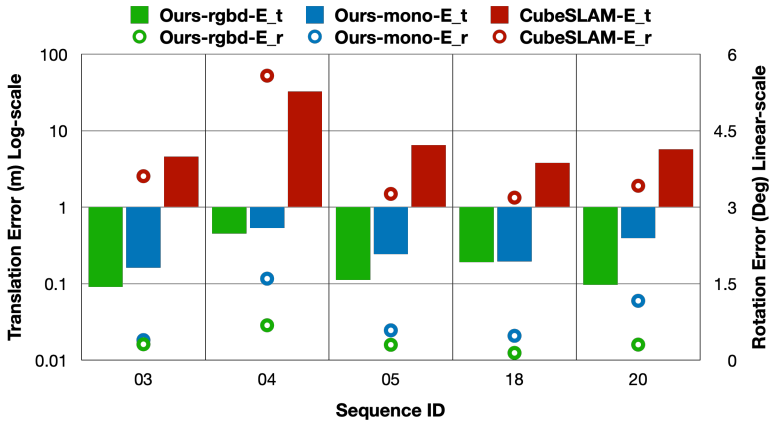


Figure 6.6: **Accuracy of object motion estimation of the proposed method compared to CubeSLAM (Yang and Scherer [2019]).** The color bars refer to translation error that is corresponding to the left Y-axis in log-scale. The circles refer to rotation error, which corresponds to the right Y-axis in linear-scale.

by the authors of CubeSLAM are reported after some correspondences. As CubeSLAM is for monocular camera, results of a learning-based monocular version of the proposed method are also computed (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, the RGB-D version gets lower errors in camera motion, while the learning-based monocular version slightly higher. We believe the weak performance of monocular version is because the model does not capture the scale of depth accurately with only monocular input. Nevertheless, both versions obtain consistently lower errors in object motion estimation. In particular, as demon-

Table 6.2: Comparison versus CubeSLAM Yang and Scherer [2019] 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	**0.0533**	**0.0547**	**1.0648**	**0.0935**	0.0738	0.0652	1.0707	0.1552	-	-	-	-
01	**0.0382**	**0.1230**	**0.6843**	**0.1423**	0.1336	0.4165	1.2301	0.3514	-	-	-	-
02	**0.0182**	**0.0457**	**1.0197**	**0.2985**	0.0352	0.0672	1.6849	0.4692	-	-	-	-
03	**0.0345**	**0.0811**	**0.3113**	**0.0912**	0.0739	0.1197	0.3936	0.1622	0.0498	0.0929	3.6085	4.5947
04	**0.0482**	**0.1114**	**0.6836**	**0.4517**	0.1065	0.2767	1.5981	0.5334	0.0708	0.1159	5.5803	32.5379
05	**0.0215**	0.0933	**0.3011**	**0.1124**	0.0576	0.0414	0.1047	0.5844	0.2438	**0.0696**	3.2610	6.4851
06	**0.0481**	**0.0186**	**0.7932**	**0.1158**	0.0502	0.0261	2.1720	0.2403	-	-	-	-
18	**0.0214**	0.0756	**0.1422**	**0.1908**	0.0861	0.0669	0.1866	0.4775	0.1944	**0.0510**	3.1876	3.7948
20	**0.0271**	**0.1662**	**0.3047**	**0.0971**	0.0688	0.3678	1.1639	0.3942	0.1348	0.1888	3.4206	5.6986

strated in Figure 6.6, 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 work 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 the object motion estimation of VDO-SLAM achieves an order of magnitude improvements. In general, the results suggest that point-based object motion/pose estimation methods is more robust and accurate than those using high-level geometric models, probably due to the fact that geometric model extraction could lead to losing information and introducing more uncertainty.

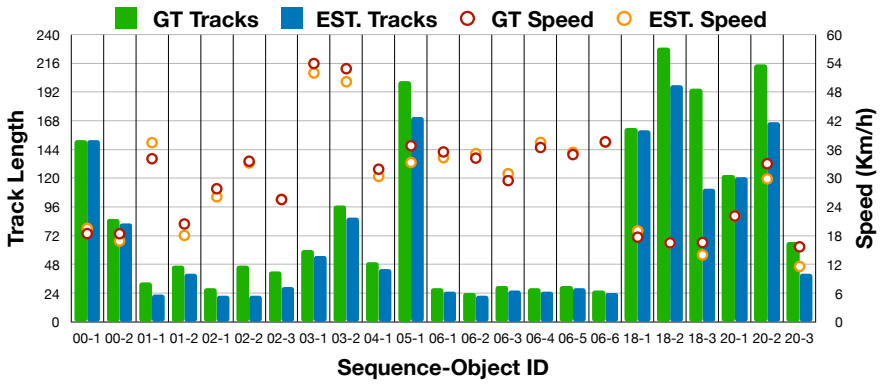


Figure 6.7: **Tracking performance and speed estimation.** Results of object tracking length and object speed for some selected objects (tracked for over 20 frames), due to limited space. The colour bars represent the length of object tracks, which is corresponding to the left Y-axis. The circles represent object speeds, which is corresponding to the right Y-axis. GT refers to ground truth, and EST. refers to estimated values.

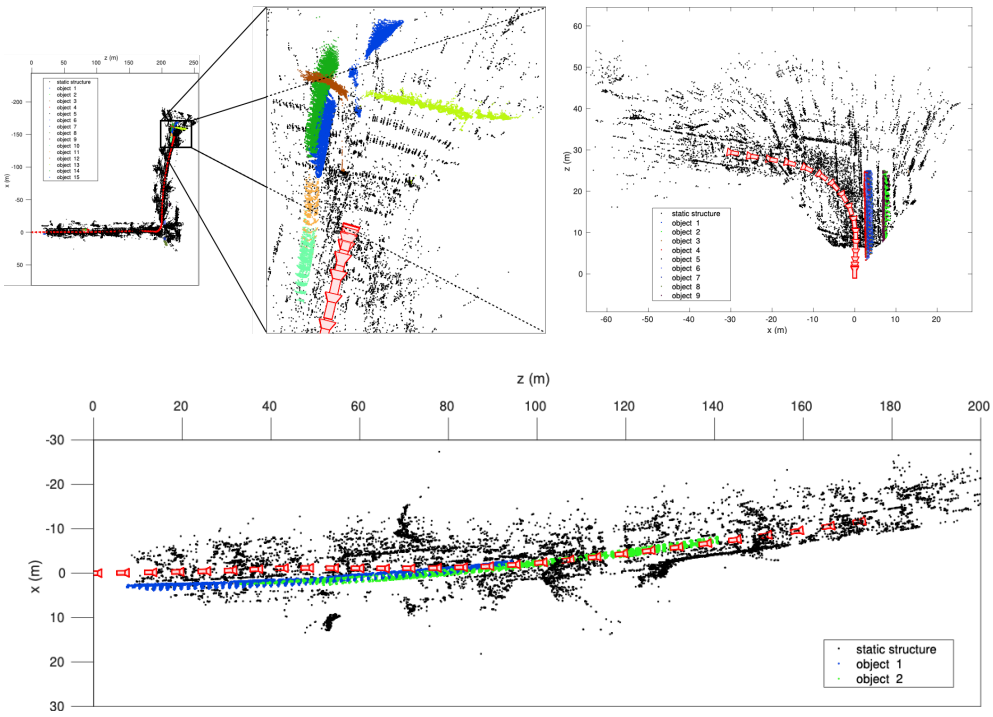


Figure 6.8: **Illustration of system output; a dynamic map with camera poses, static background structure, and tracks of dynamic objects.** Sample results of VDO-SLAM on KITTI sequences. Black represents static background, and each detected object is shown in a different colour. Top left figure presents Sequence 01 with a zoom-in on the intersection at the end of the sequence; top right figure presents Sequence 06 and bottom figure presents Sequence 03.

6.4.4.2 Object Tracking and Velocity

The performance of tracking dynamic objects is also evaluated, with results of object speed estimation demonstrated, which is an important information for autonomous driving applications. Figure 6.7 illustrates results of object tracking length and object speed for some selected objects (tracked for over 20 frames) in all the tested sequences. The proposed 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.

6.4.4.3 Qualitative Results

Figure 6.8 illustrates output of VDO-SLAM 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.

6.4.5 Discussion

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

Table 6.3: 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**

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 5.2.3. The effectiveness of joint optimisation is shown by comparing a baseline method that only optimises for the motion (Motion Only) using (5.5) for camera motion or (5.13) for object motion, and the improved method that optimises for both the motion and the optical flow (Joint) using (5.14). Table 6.3 demonstrates that the joint method obtains considerably more points that are tracked for long periods.

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 6.4.

Table 6.4: 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 Indirect Occlusion

The mask segmentation may fail in some cases, due to direct or indirect occlusions (illumination change, *etc.*). Thanks to the mask propagating method described in Section 6.3.3.3, the proposed system is able to handle mask failure cases caused by indirect occlusions. Figure 6.9 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 proposed system is still able to continuously track the van, and estimate its speed with an average error of 2.64 km/h across the whole sequence. Notice that, speed errors in the second half of the sequence are higher due to partial occlusions, and increased distance to the object getting farther away from the camera.

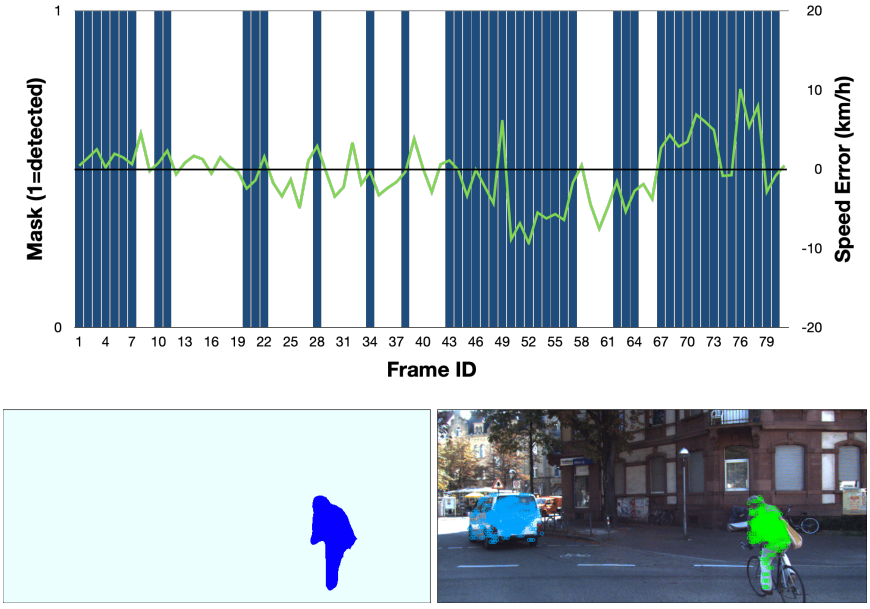


Figure 6.9: **Robustness in tracking performance and speed estimation in case of semantic segmentation failure.** An example of tracking performance and speed estimation for a white van (ground-truth average speed 20km/h) in Sequence 00. (Top) Blue bars represent a successful object segmentation, and 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 by the proposed system.

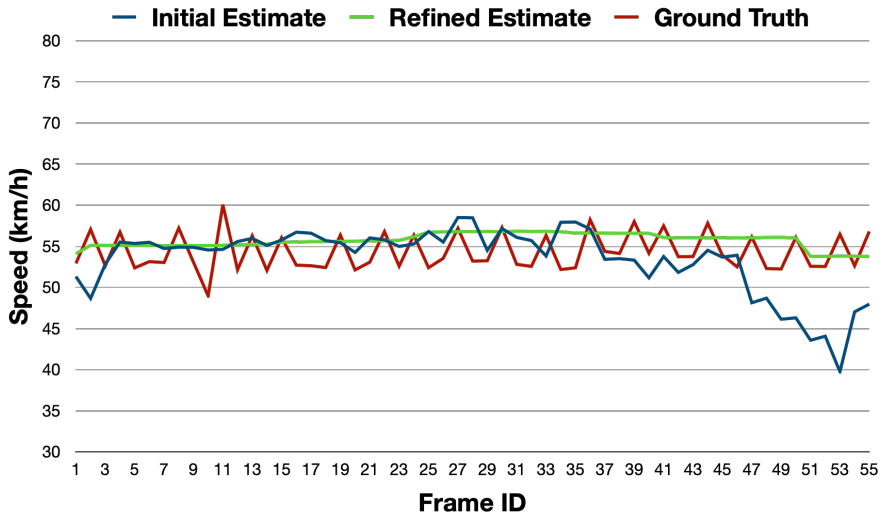


Figure 6.10: **Global refinement effect on object motion estimation.** The initial (blue) and refined (green) estimated speeds of a wagon in Sequence 03, travelling along a straight road, compared to the ground truth speed (red).

6.4.5.3 Global Refinement on 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 Figure 6.10, 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, a novel factor graph is formulated to refine the motions together with the static and dynamic structure as discussed in Section 6.2.2. The green curve in Figure 6.10 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 6.11 demonstrates the average improvement for all objects in each sequence of KITTI dataset. With graph optimization, the errors can be reduced up to 39% in translation and 55% in rotation. Interestingly, the translation errors in Seq.18 and Seq.20 increase slightly. We believe it is because the vehicles keep alternating between acceleration and deceleration due to the heavy traffic jams in both sequences, which strongly violates the smooth motion constraint that is set for general cases.

6.4.5.4 Computational Analysis

Finally, the computational analysis of the proposed system is provided. 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

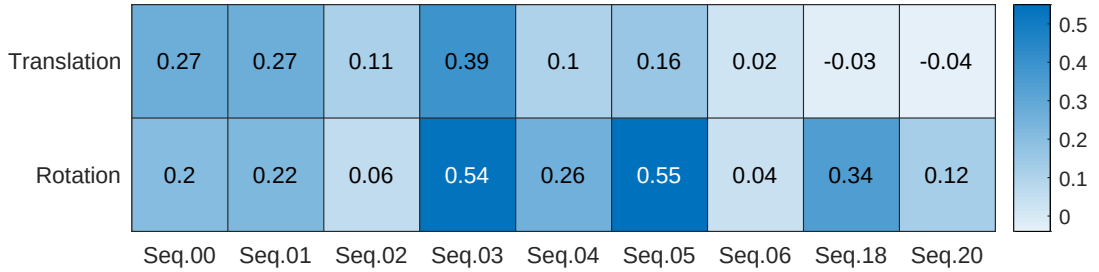


Figure 6.11: **Improvement on object motion after graph optimization.** The numbers in the heatmap show the ratio of decrease in error on the nine sequences of the KITTI dataset.

CNN model complexity. Many current state-of-the-art algorithms can run in real time Bolya et al. [2019]; Hui et al. [2020]. In this chapter, 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. The computational time is shown in Table 6.5 for both datasets. Note that, for both dataset, the maximum number of points to track on background and each object are 1200 and 800, respectively. 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, except for the dynamic object tracking and object motion estimation that are averaged over the number of objects. 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.

6.5 Conclusion

In this chapter, a novel dynamic feature-based SLAM system, VDO-SLAM, is presented, which 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. I believe the high performance accuracy achieved in object motion estimation is due to the fact that the proposed 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.

An important issue to be addressed 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 Strasdat et al. [2011]; Ila et al. [2010]. In fact, history summarisation/deletion of map points pertaining to dynamic objects observed far in the past seems to be a natural

Table 6.5: Runtime of different system components for both datasets. The time cost of every component is averaged over all frames and sequences, except for the dynamic object tracking and object motion estimation that are averaged over the number of objects.

Dataset	Tasks	Runtime (mSec)
KITTI	Feature Detection	16.2550
	Camera Pose Estimation	52.6542
	Dynamic Object Tracking (avg/object)	8.2980
	Object Motion Estimation (avg/object)	22.9081
	Map and Mask Updating	22.1830
	Local Batch Optimisation	18.2828
OMD	Feature Detection	7.5220
	Camera Pose Estimation	32.0909
	Dynamic Object Tracking (avg/object)	7.0134
	Object Motion Estimation (avg/object)	19.5280
	Map and Mask Updating	30.3153
	Local Batch Optimisation	15.3414

step towards a long-term SLAM system in highly dynamic environments.

Chapter 7

Conclusion

Visual odometry and visual SLAM have been fundamental research topics in the robotics and computer vision communities for more than two decades. As their application domains become larger, we witness new challenging scenarios where application-specific conditions occur. This drives us to develop more accurate, efficient and robust systems on the basis of new, tailor-made theories and effective implementations.

7.1 Summary and Contributions

This thesis developed, tested and evaluated new visual odometry algorithms that are able to robustly localize the robot in two challenging cases, underwater and dynamic environments. The demonstrated results for both cases push the limits of current state-of-the-art in terms of applicability, accuracy and robustness. Beyond that, to explore the valuable information in dynamic environments, the thesis proposed to extract the multiple moving objects in the scene, and track their motions separately. Furthermore, a unified framework was formulated to exploit the spatial and temporal relations between the camera and object motions, as well as the scene structures (static and dynamic), to achieve more consistent and accurate estimations. A summary of the contributions is given in the following.

- A robust and effective stereo underwater visual odometry system is developed to be able to accurately recover the camera motion and localize the AUV. Intensive analysis and evaluation of components in the VO system, including image restoration, feature extraction and matching, as well as motion estimation, are performed to tackle the problems of poor imaging quality and inconsistent motion in underwater scenes, and explore possibility of improvement on the system in underwater environments. Experimental results indicate the proposed system helps to achieve excellent performance in localizing the AUV underwater. Benchmarking on the KITTI dataset demonstrates satisfactory quantitative results compared to the state-of-the-art.

- A novel way of modelling and estimating object motion is proposed, along with a brand new dynamic multi-body visual odometry framework to simultaneously track camera and multiple object motions. The proposed framework detects moving objects via combining instance-level object segmentation and scene flow, and tracks them over frames using optical flow. A novel unified formulation that jointly optimizes the $\mathrm{SE}(3)$ motions

(camera or object) and the optical flow, is introduced to improve the quality of the tracked points and the motion estimation accuracy. The proposed approach is carefully evaluated on the virtual dataset and extensively tested on the real dataset. The experimental results show that the proposed method is able to obtain robust and accurate camera trajectories in dynamic scene, and track the velocity of objects with high accuracy.

- A full dynamic visual SLAM system is proposed as an extension of the multi-body VO front-end, which integrates the static and dynamic structures of the environment into an unified estimation framework. The full SLAM system is able to achieve robust moving object tracking, accurate estimation of dynamic objects full $\mathrm{SE}(3)$ pose change, and extract velocity information of moving objects in the scene, as well as an accurate robot localisation and mapping of static environment structure. The proposed system consistently shows robust and accurate results on indoor and challenging outdoor datasets, and establishes state-of-the-art performance in object motion estimation.

7.2 Future Work

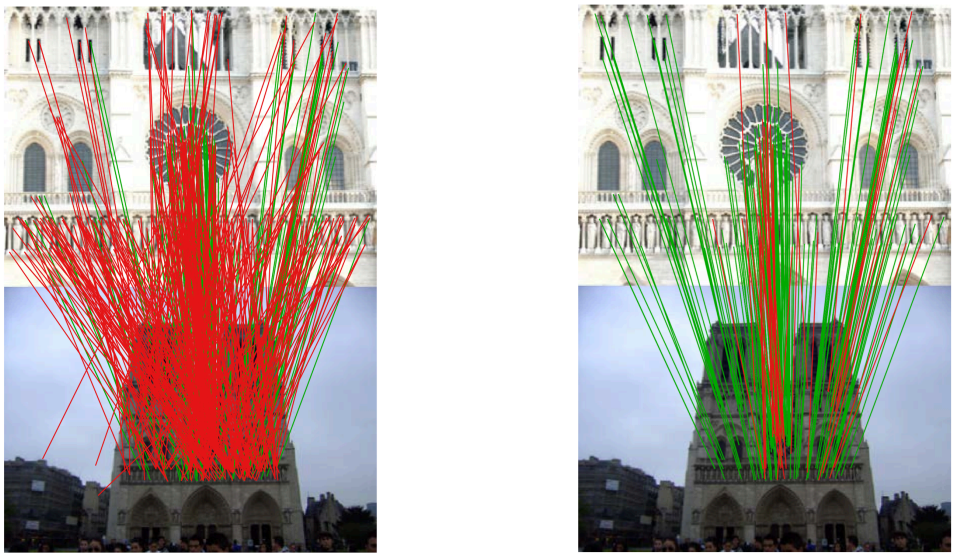
The proposed algorithms in this thesis are certainly possible to be further refined. Discussions on the potential improvements have already been provided in the conclusion part of each chapter. In this section, the author would like to discuss about several potential projects that are closely related to the proposed methods, and are meaningful and valuable, from a personal perspective, for the research direction studied in this thesis, and the research community of visual odometry and SLAM.

7.2.1 Features to Track on Moving Objects

Local features extraction and matching has been the subject of vast amounts of research in computer vision and robotics areas. Several traditional hand-crafted feature techniques have been discussed in Section 4.5.1. Over the past few years, new approaches based on Deep Learning have become the research focus and started to outperform the traditional methods.

One of the representative work is LIFT Yi et al. [2016], a deep network framework implementing the full feature extraction pipeline of detection, orientation estimation and feature description. It has been proved to outperform the state-of-the-art methods due to its unified optimization of each individual steps. To deal with feature association across images, Moo Yi et al. [2018] introduces a deep architecture to learn to find good correspondences, given a set of putative sparse matches and the camera intrinsics. Meanwhile, the relative pose between images is also recovered using the selected good correspondences. The authors proved the effectiveness of the pipeline against the cases of wide baselines, repetitive structure and illumination changes, compared with typical methods such as RANSAC Fischler and Bolles [1981b], as demonstrated in Figure 7.1.

The above solutions only consider finding feature correspondences between two images, and achieve better performance with static world assumption. Nevertheless, I believe the problem of consistently tracking good features in the scene, particularly on the moving objects for a period of time steps, is important for VO/SLAM for high dynamic scenarios, but is also ex-



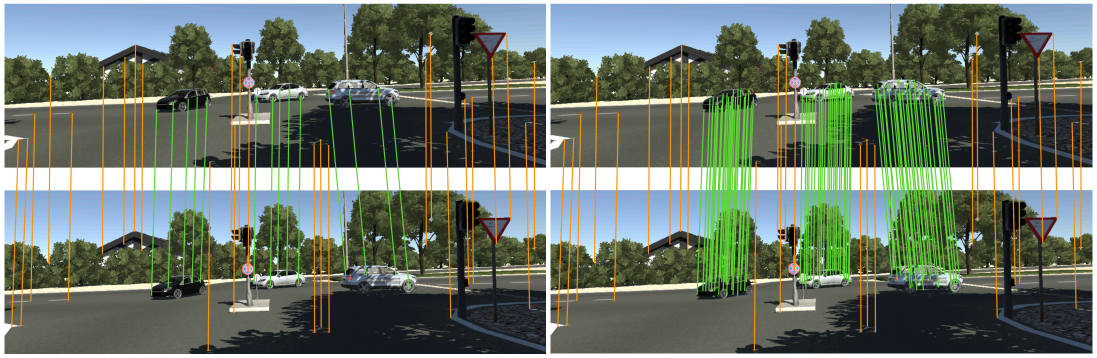
(a) RANSAC (b) Method in Moo Yi et al. [2018]

Figure 7.1: **A demo of matching 2k SIFT keypoints from a challenging image pair, with inlier matches found with RANSAC (a) and the proposed approach in Moo Yi et al. [2018] (b).** The matches are drawn in green if they conform to the ground-truth Epipolar geometry, and in red otherwise.

tremely challenging. Specifically, as illustrated in Figure 7.2 (a), only limited number of valid correspondences can be found using traditional detecting and matching algorithms. Fewer features are on the moving objects, leading to easily losing tracking of them within a few frames. Considering this and the proposed solution in Moo Yi et al. [2018], an end-to-end learning framework can be developed to find rich and good quality feature tracks (see Figure 7.2 (b)), and simultaneously the motions of both the camera and objects. What's more, an in-depth analysis of the quantity and quality of the feature tracks that decides the optimal solution of camera and object motion can be performed.

7.2.2 Pixel-wise Rigid Motions: Segment, Estimate and Track

As discussed in Section 5.3.3.1, scene flow is a strong constraint to decide whether the scene is dynamic or static, and it is relied on the estimate of relative camera motion. Based on these consideration, current state-of-the-art algorithms made their efforts to jointly solve the problem of scene flow, relative pose, and motion segmentation, and make them benefit from each others. For instance, in Taniai et al. [2017], the authors introduce an algorithm of multi-frame scene flow estimation with motion segmentation. The primary contribution is that they present a unified framework to compute dense disparity, optical flow and 6-DoF camera motion, as well as segment the foreground from background on consecutive frames. Similarly, Lv et al. [2018] proposes a Rigid Transform Network to estimate relative camera transform and rigid(background)/ non-rigid(foreground) regions. Then the relative camera pose is refined with dense optical flow over the background region, and finally scene flow is calculated with



(a) (b)

Figure 7.2: **A sketch map of (a) typical matching results using traditional methods and (b) possible matching results with the proposed idea.** Note that lines of orange color refer to correspondences on the background, while those of green color refer to correspondences on the moving objects.

the refined pose.

The outlined methods above all focus on estimating scene flow with foreground and background separation. What can be done differently, is to estimate multiple rigid motions with rigid objects and background separation. Combining with appearance cue (*e.g.*, semantic information), rigid motion can act as important geometry cue for segmenting and tracking rigid objects over multiple frames. Overall, the ideas can be summarized as a problem to be tackled: given prior camera and object motions, optical flow and instance-level semantic segmentation mask, is it possible to effectively fuse these cues to get pixel level segmentation of rigid moving objects, together with estimation of camera and object motions in new coming frame? This problem is essentially an extension of the proposed framework in Chapter 5 towards dense, pixel-wise level, with exploration of the inherent relationship for mutual improvement of the sub-tasks. Note that the proposal of dense framework is considered for the reason that the whole image region is fully exploited to offer richer information for dynamic scene modelling. For instance, moving objects in the scene can be tracked consistently even they are distant from the camera or partial-occluded, while this is quite hard to achieve using sparse features.

Formally, given a reasonable initialization, as shown in Figure 7.3, the camera pose, an estimated mask (orange) of an object and its motion ${}^{k-2}_{k-2}\mathbf{H}_{k-1}$ from frame $k-2$ to $k-1$, assume the object with a nearly constant motion, *i.e.*,

$$ {}^{k-2}_{k-2}\mathbf{H}_{k-1} \approx {}^{k-1}_{k-1}\mathbf{H}_{k} = \mathbf{H}\,. \tag{7.1} $$

Given the ego-motion ${}^{k-1}_{k-1}\mathbf{T}_k$ computed at k frame, the 3D structure of estimated mask is moved via $\mathbf{H}$ and projected into the k frame image plane, resulting a projected mask (green). Besides, a flow mask (blue) can be obtained with provided optical flow at frame k, as well as a semantic mask (red) estimated using semantic segmentation method. With all the provided prior information, the goal is to get an optimal segmentation mask and motion estimation of the rigid object ${}^{k-1}_{k-1}\mathbf{H}_k$.

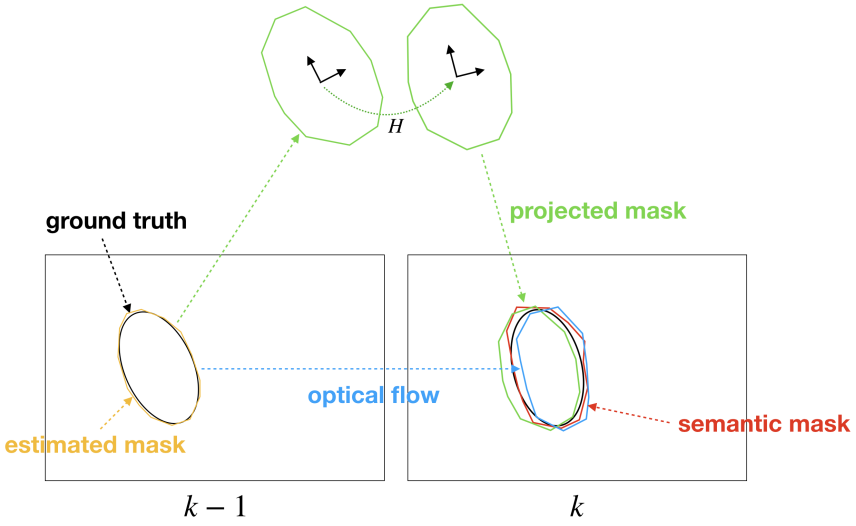


Figure 7.3: **A sketch map of analysing the motion and segmentation of one moving object between two consecutive frames.** Different mask color in the image region indicates the mask is generated by a specific method.

To sum up, though the problem of moving object tracking (MOT) has been a popular topic that is well studied, tracking of object in feature level still remains a challenging task due to illumination change and occlusion, and it is a vita part for high-level scene understanding, *e.g.*, static and dynamic object modelling. In addition, scene flow is a basic but essential information for dynamic scene understanding. A lot of efforts have been made to improve accuracy, but few has exploited its potential applications in real scenarios. Hence I believe it is of significant value to go deeper into high-level dynamic scene modelling with rigid motion flow, building and maintaining a map of static structure, as well as tracking dynamic objects.

7.3 Discussions

Finally, the author would like to discuss a number of interesting questions, which are highly relevant to the topic of this thesis and the research community of VO and SLAM.

- **Is it important to model dynamic information (motion, shape, *etc.*) in VO/SLAM problems?** The author considers that the answer is definitely yes. Regarding potential benefits and applications, the problem of accurately identifying, tracking and estimating the motion of objects in the scene is relevant in many scenarios. One important application is in autonomous and safety driving systems that are required to robustly work in high dynamic environments such as urban and highways. Estimating and modelling the motion of the objects allows for accurate predictions, which provides essential information in collision avoidance and planning in dynamic environments to ensure the safety of drivers and pedestrians. Moreover, modelling dynamic information can also be used to assist driver to monitor the surroundings and release driving fatigue. For example,

self-adaptive cruise control, which is commonly equipped in modern cars, can adaptively adjust self-speed on highways and reduce operation from the driver, by accurately tracking and estimating the distance and speed of a targeted vehicle. Other similar applications include visual assistance system for the blind and visual surveillance system for security, *etc.* The benefits of modelling dynamic information can also be gained from augmented reality, where real time modelling of the dynamic scene can provide opportunities in gaming and other AR applications. In current AR solutions, the simulated characters can only interact with static structure, while mapping the dynamic scenes in real time allows AR applications to show interactions with dynamic objects and enhance the sense of reality.

- **What is the best way to model objects in VO/SLAM problems?** This is an interesting open question that attracts the attention of many researchers in the SLAM community. In the literature, many methods have been proposed using sparse Reddy et al. [2016]; Zou and Tan [2013] or dense points Runz et al. [2018]; Rünz and Agapito [2017]; Xu et al. [2019] to represent objects, which can be easily obtained from sensors such as cameras and Lidars, containing rich geometry information with relatively lower uncertainty. However, sparse points does not guarantee robust or long-term tracking of objects, while dense representation requires heavy computational and storage resources. Thanks to the fast development of deep learning based semantic scene understanding and object tracking in the computer vision community, new object representations for visual SLAM start to gain more popularity, such as 3D bounding box Yang and Scherer [2019]; Huang et al. [2020] and quadric Nicholson et al. [2018b]; Hosseinzadeh et al. [2019]. The new high-level entities has the advantage of compact formulation with lower memory cost, hence ensuring high efficiency of mapping in SLAM. The obvious disadvantage, nevertheless, lies in the lower accuracy in modelling the objects (pose/motion estimation) compared to point-based methods, since less parameters imply higher uncertainty for model estimation. To summarize, the author believes there is a clear need for a better representation of objects in VO/SLAM problems, with careful consideration of the trade-off between accuracy and computational cost.

- **Can event camera sensors be used to solve the dynamic VO/SLAM problems?** This question is raised on the basis of the new emerging technology of event cameras for their nature of capturing the dynamics of the scenes. Event cameras do not capture images using a shutter as conventional cameras do. Instead, each pixel inside an event camera operates independently and asynchronously, reporting changes in brightness as they occur, and staying silent otherwise. Because of the special mechanism, event cameras have the advantages of high dynamic range and temporal resolution, thereby are free from motion blur. This makes event cameras better sensors for highly dynamic environments and opens new research areas in dynamic 'events' tracking and event-based dynamic scene modelling. Although a considerable amount of event-based VO/SLAM methods Censi and Scaramuzza [2014]; Kueng et al. [2016]; Rebecq et al. [2017]; Vidal et al. [2018]; Mueggler et al. [2018]; Weikersdorfer et al. [2014]; Milford et al. [2015] and very few approaches concerning dynamic scene understanding using event cameras Stoffregen et al. [2019]; Falanga et al. [2020] have been proposed in the literature, no proposed

solution has been found to tackle the dynamic VO/SLAM problem using event sensors. However, the author believes, by fully exploiting the advantages of event cameras to solve dynamic VO/SLAM problems, the potential benefits brought by the new sensors will push the limits of the state-of-the-art towards a higher level in performance and a wider domain of application.

Appendix

A.1 Derivation on Jacobian Matrix of Joint Optic-flow and Object Motion Cost

If a good initial guess is given, the numerical solution of Equation (5.14) can be obtained by using Gauss-Newton or Levenberg-Marquart algorithms as discussed in Section 3.4.3. The idea is to approximate the error function

$$\mathbf{e}_i(\boldsymbol{\theta}, \boldsymbol{\phi}^i) := {}^{k-1}\mathbf{p}^i + {}^{k}\boldsymbol{\phi}^i - {}^{k}\hat{\mathbf{p}}^i, \tag{A.1}$$

by its first order Taylor expansion around the current initial guess:

$$\mathbf{e}_i(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}, \boldsymbol{\phi}^i + \Delta\boldsymbol{\phi}^i) \simeq \mathbf{e}_i(\boldsymbol{\theta}, \boldsymbol{\phi}^i) + \mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\theta})\Delta\boldsymbol{\theta} + \mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\phi}^i)\Delta\boldsymbol{\phi}^i, \tag{A.2}$$

where $\mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\theta})$ and $\mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\phi}^i)$ are the Jacobians of $\mathbf{e}_i(\boldsymbol{\theta}, \boldsymbol{\phi}^i)$ computed in $\boldsymbol{\theta}$ and $\boldsymbol{\phi}^i$, respectively.

Assume $\mathbf{G}$ is the $\mathrm{SE}(3)$ element of $\boldsymbol{\theta}$, which can be either the camera motion ${}^{k-1}_{k-1}\mathbf{T}_k$ in (5.3) or the variant of object motion ${}^{k-1}_{k-1}\mathbf{G}_k$ in (5.12). the Jacobian matrix $\mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\theta})$ could be obtained using chains rule as

$$\mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\theta}) = \mathbf{J}_{\mathbf{e}}\mathbf{J}_{\pi}\mathbf{J}_{\mathbf{G}}. \tag{A.3}$$

Since

$${}^{k}\hat{\mathbf{p}}^i \simeq \pi(\boldsymbol{\theta}) \tag{A.4}$$

is a projection function of $\pi(\boldsymbol{\theta})$, we can easily know that,

$$\mathbf{J}_{\mathbf{e}} = \frac{\partial \mathbf{e}_i}{\partial \pi}|_{\pi={}^{k}\hat{\mathbf{p}}^i} = -1. \tag{A.5}$$

The partial derivative of π regarding to the 3D point ${}^{k}\mathbf{m}_{k-1}^{i} = \mathbf{G}^{k-1}\mathbf{m}_{k-1}^{i}$ is:

$$\mathbf{J}_{\pi} = \frac{\partial \pi}{\partial \mathbf{m}}|_{\mathbf{m}={}^{k}\mathbf{m}_{k-1}^{i}=\mathbf{G}^{k-1}\mathbf{m}_{k-1}^{i}} = \begin{bmatrix} \frac{f_u}{m_z^i} & 0 & -\frac{f_u \cdot m_x^i}{(m_z^i)^2} \\ 0 & \frac{f_v}{m_z^i} & -\frac{f_v \cdot m_y^i}{(m_z^i)^2} \end{bmatrix}. \tag{A.6}$$

The third part of the Jacobian is the derivative of the 3D point ${}^{k}\mathbf{m}_{k-1}^{i}$ with respect to the transformation vector $\boldsymbol{\theta} \in \mathrm{se}(3)$, which can be found by Eade [2013]:

$$\mathbf{J}_{\pi} = \frac{\partial \mathbf{G}}{\partial \boldsymbol{\theta}}|_{\mathbf{G}=\mathbf{G}(\boldsymbol{\theta}),\mathbf{m}={}^{k}\mathbf{m}_{k-1}^{i}} = \left(\mathbf{I}_3 \,|\, -\mathbf{m}_{\times}\right), \tag{A.7}$$

where $\mathbf{m}_{\times}$ is the skew-symmetric matrix obtained using the Lie algebra generators for $\mathrm{so}(3)$ (Equation (3.37)), and is given as,

$$\mathbf{m}_{\times} = \begin{bmatrix} 0 & -m_z^i & m_y^i \\ m_z^i & 0 & -m_x^i \\ -m_y^i & m_x^i & 0 \end{bmatrix}. \tag{A.8}$$

Finally, multiplying all the sub Jacobian matrices, we have

$$\mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\theta}) = \begin{bmatrix} -\frac{f_u}{m_z^i} & 0 & \frac{f_u \cdot m_x^i}{(m_z^i)^2} & \frac{f_u \cdot m_x^i \cdot m_y^i}{(m_z^i)^2} & -f_u\left(1+\frac{(m_x^i)^2}{(m_z^i)^2}\right) & \frac{f_u \cdot m_y^i}{m_z^i} \\ 0 & -\frac{f_v}{m_z^i} & \frac{f_v \cdot m_y^i}{(m_z^i)^2} & f_v\left(1+\frac{(m_y^i)^2}{(m_z^i)^2}\right) & -\frac{f_v \cdot m_x^i \cdot m_y^i}{(m_z^i)^2} & -\frac{f_v \cdot m_y^i}{m_z^i} \end{bmatrix}. \tag{A.9}$$

The calculation of Jacobian matrix $\mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\phi}^i)$ is straightforward, since the variable $\boldsymbol{\phi}^i$ is a linear vector. Given the error cost in (A.1), we have

$$\mathbf{J}_{\mathbf{e}_i}(\boldsymbol{\phi}^i) = \frac{\partial \mathbf{e}}{\partial \boldsymbol{\phi}}|_{\boldsymbol{\phi}={}^{k}\boldsymbol{\phi}^i} = \mathbf{I}_2 \,. \tag{A.10}$$

A.2 Derivation on Jacobian Matrix of Object Motion Model Ternary Factor

Parameterise the object motion ${}_{k-1}^{0}\mathbf{H}_k^l \in \mathrm{SE}(3)$ by elements of the Lie-algebra ${}_{k-1}^{0}\mathbf{h}_k^l \in \mathrm{se}(3)$, the linearisation of the motion model error in (6.6) around the initial guess is,

$$\begin{aligned} \mathbf{e}_{i,l,k}({}^{0}\mathbf{m}_k^i + \Delta\mathbf{m}_k, {}_{k-1}^{0}\mathbf{h}_k^l + \Delta\mathbf{h}, {}^{0}\mathbf{m}_{k-1}^i + \Delta\mathbf{m}_{k-1}) \simeq\ & \mathbf{e}_{i,l,k}({}^{0}\mathbf{m}_k^i, {}_{k-1}^{0}\mathbf{h}_k^l, {}^{0}\mathbf{m}_{k-1}^i) \\ & + \mathbf{J}_{\mathbf{e}_{i,l,k}}({}^{0}\mathbf{m}_k^i)\Delta\mathbf{m}_k \\ & + \mathbf{J}_{\mathbf{e}_{i,l,k}}({}_{k-1}^{0}\mathbf{h}_k^l)\Delta\mathbf{h} \\ & + \mathbf{J}_{\mathbf{e}_{i,l,k}}({}^{0}\mathbf{m}_{k-1}^i)\Delta\mathbf{m}_{k-1} \,. \end{aligned} \tag{A.11}$$

Each sub Jacobian matrix are derived as in the following.
The partial derivative of $\mathbf{e}_{i,l,k}$ with respect to ${}^0\mathbf{m}_k^i$ is

$$\mathbf{J}_{\mathbf{e}_{i,l,k}}({}^0\mathbf{m}_k^i) = \frac{\partial \mathbf{e}}{\partial \mathbf{m}}|_{\mathbf{m}={}^0\mathbf{m}_k^i} = \mathbf{I}_3\,. \tag{A.12}$$

Similar to (A.7), the derivative of the 3D point ${}^0\mathbf{m}_{k-1}^i$ with respect to the transformation vector ${}_{k-1}^0\mathbf{h}_k^l \in \mathrm{se}(3)$, which can be found as,

$$\mathbf{J}_{\mathbf{e}_{i,l,k}}({}_{k-1}^0\mathbf{h}_k^l) = \frac{\partial \mathbf{e}}{\partial \mathbf{h}}|_{\mathbf{h}={}_{k-1}^0\mathbf{h}_k^l,\mathbf{m}={}^0\mathbf{m}_{k-1}^i} = \left(\mathbf{I}_3 \,|\, -\mathbf{m}_\times\right). \tag{A.13}$$

Finally, the partial derivative of $\mathbf{e}_{i,l,k}$ with respect to ${}^0\mathbf{m}_{k-1}^i$ is,

$$\mathbf{J}_{\mathbf{e}_{i,l,k}}({}^0\mathbf{m}_{k-1}^i) = \frac{\partial \mathbf{e}}{\partial \mathbf{m}}|_{\mathbf{m}={}^0\mathbf{m}_{k-1}^i,\mathbf{h}={}_{k-1}^0\mathbf{h}_k^l} = -\mathbf{R}_\mathbf{H}\,, \tag{A.14}$$

where $\mathbf{R}_\mathbf{H} \in \mathrm{SO}(3)$ is the rotation part of ${}_{k-1}^0\mathbf{H}_k^l$.

Bibliography

AGARWAL, S.; MIERLE, K.; AND OTHERS, 2012. Ceres solver. http://ceres-solver.org. (cited on pages 29 and 69)

ALCANTARILLA, P. F.; YEBES, J. J.; ALMAZÁN, J.; AND BERGASA, L. M., 2012. 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*, 1290–1297. IEEE. (cited on pages 11 and 47)

ANCUTI, C.; ANCUTI, C. O.; HABER, T.; AND BEKAERT, P., 2012. Enhancing underwater images and videos by fusion. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 81–88. IEEE. (cited on page 42)

ARUN, K. S.; HUANG, T. S.; AND BLOSTEIN, S. D., 1987. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, , 5 (1987), 698–700. (cited on page 18)

BÂRSAN, I. A.; LIU, P.; POLLEFEYS, M.; AND GEIGER, A., 2018. Robust Dense Mapping for Large-Scale Dynamic Environments. In *International Conference on Robotics and Automation (ICRA)*. (cited on page 12)

BAY, H.; TUYTELAARS, T.; AND VAN GOOL, L., 2006. SURF: Speeded up robust features. In *European conference on computer vision*, 404–417. Springer. (cited on page 40)

BEALL, C.; DELLAERT, F.; MAHON, I.; AND WILLIAMS, S. B., 2011. Bundle adjustment in large-scale 3d reconstructions based on underwater robotic surveys. In *OCEANS 2011 IEEE-Spain*, 1–6. IEEE. (cited on pages 10 and 11)

BELLAVIA, F.; FANFANI, M.; AND COLOMBO, C., 2017. Selective visual odometry for accurate auv localization. *Autonomous Robots*, 41, 1 (2017), 133–143. (cited on page 11)

BESCOS, B.; CAMPOS, C.; TARDÓS, J. D.; AND NEIRA, J., 2020. Dynaslam ii: Tightly-coupled multi-object tracking and slam. *arXiv preprint arXiv:2010.07820*, (2020). (cited on page 13)

BESCOS, F. J. C. J., BERTA AND NEIRA, J., 2018. DynaSLAM: Tracking, mapping and inpainting in dynamic environments. *IEEE RA-L*, (2018). (cited on pages 11, 12, and 63)

BESL, P. J. AND MCKAY, N. D., 1992. Method for registration of 3-d shapes. In *Robotics-DL tentative*, 586–606. International Society for Optics and Photonics. (cited on page 10)

BLANCO, J.-L., 2010. A tutorial on SE(3) transformation parameterizations and on-manifold optimization. *University of Malaga, Tech. Rep*, 3 (2010). (cited on page 27)

BOLYA, D.; ZHOU, C.; XIAO, F.; AND LEE, Y. J., 2019. Yolact: real-time instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 9157–9166. (cited on page 82)

BOUGUET, J.-Y., 1996. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html. (cited on page 16)

BRADSKI, G., 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, (2000). (cited on page 16)

BURGUERA, A.; BONIN-FONT, F.; AND OLIVER, G., 2014. Towards robust image registration for underwater visual slam. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, vol. 3, 539–544. IEEE. (cited on page 2)

BURGUERA, A.; BONIN-FONT, F.; AND OLIVER, G., 2015. Trajectory-based visual localization in underwater surveying missions. *Sensors*, 15, 1 (2015), 1708–1735. (cited on page 10)

BUTLER, D. J.; WULFF, J.; STANLEY, G. B.; AND BLACK, M. J., 2012. A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, 611–625. Springer-Verlag. (cited on pages 56 and 73)

BYRAVAN, A. AND FOX, D., 2017. Se3-nets: Learning rigid body motion using deep neural networks. In *IEEE International Conference on Robotics and Automation (ICRA), 2017*, 173–180. IEEE. (cited on page 65)

CENSI, A. AND SCARAMUZZA, D., 2014. Low-latency event-based visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 703–710. IEEE. (cited on page 90)

CHANG, C. AND CHATTERJEE, S., 1992. Quantization error analysis in stereo vision. In *[1992] Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems & Computers*, 1037–1041. IEEE. (cited on page 57)

CHEEIN, F. A.; STEINER, G.; PAINA, G. P.; AND CARELLI, R., 2011. Optimized eif-slam algorithm for precision agriculture mapping based on stems detection. *Computers and electronics in agriculture*, 78, 2 (2011), 195–207. (cited on page 3)

CHIRIKJIAN, G. S.; MAHONY, R.; RUAN, S.; AND TRUMPF, J., 2017. Pose changes from a different point of view. In *Proceedings of the ASME International Design Engineering Technical Conferences (IDETC) 2017*. ASME. (cited on pages 51 and 56)

COMPORT, A. I.; MALIS, E.; AND RIVES, P., 2007. Accurate quadrifocal tracking for robust 3d visual odometry. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 40–45. IEEE. (cited on page 9)

COMPORT, A. I.; MALIS, E.; AND RIVES, P., 2010. Real-time quadrifocal visual odometry. *The International Journal of Robotics Research*, 29, 2-3 (2010), 245–266. (cited on page 9)

COMPORT, A. I.; MARCHAND, E.; PRESSIGOUT, M.; AND CHAUMETTE, F., 2006. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on visualization and computer graphics*, 12, 4 (2006), 615–628. (cited on page 2)

CREUZE, V., 2017. Monocular odometry for underwater vehicles with online estimation of the scale factor. In *IFAC World Congress*. (cited on page 10)

DAVISON, A. J., 2003. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE Internation Conference on*, 1403–1410. IEEE. (cited on page 7)

DAVISON, A. J. AND MURRAY, D. W., 2002. Simultaneous localization and map-building using active vision. *IEEE transactions on pattern analysis and machine intelligence*, 24, 7 (2002), 865–880. (cited on page 7)

DAVISON, A. J.; REID, I. D.; MOLTON, N. D.; AND STASSE, O., 2007. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29, 6 (2007), 1052–1067. (cited on page 7)

DE LA PUENTE, P. AND RODRÍGUEZ-LOSADA, D., 2014. Feature based graph-SLAM in structured environments. *Autonomous Robots*, 37, 3 (2014), 243–260. (cited on page 65)

DELLAERT, F., 2012. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology. (cited on page 29)

DELLAERT, F. AND KAESS, M., 2006. Square Root Sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25, 12 (2006), 1181–1203. (cited on page 69)

DEWAN, A.; CASELITZ, T.; TIPALDI, G. D.; AND BURGARD, W., 2016. Motion-based detection and tracking in 3d lidar scans. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 4508–4513. IEEE. (cited on page 14)

DHARMASIRI, T.; LUI, V.; AND DRUMMOND, T., 2016. Mo-slam: Multi object slam with run-time object discovery through duplicates. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1214–1221. IEEE. (cited on page 13)

DISSANAYAKE, M. G.; NEWMAN, P.; CLARK, S.; DURRANT-WHYTE, H. F.; AND CSORBA, M., 2001. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17, 3 (2001), 229–241. (cited on pages 7 and 20)

DONG, J.; BURNHAM, J. G.; BOOTS, B.; RAINS, G.; AND DELLAERT, F., 2017. 4d crop monitoring: Spatio-temporal reconstruction for agriculture. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 3878–3885. IEEE. (cited on page 3)

DURRANT-WHYTE, H. AND BAILEY, T., 2006. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13, 2 (2006), 99–110. (cited on page 7)

EADE, E., 2013. Lie groups for 2d and 3d transformations. *URL http://ethaneade. com/lie. pdf, revised Dec*, (2013). (cited on pages 27 and 94)

EIGEN, D.; PUHRSCH, C.; AND FERGUS, R., 2014. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, 2366–2374. (cited on page 73)

ENGEL, J.; KOLTUN, V.; AND CREMERS, D., 2017. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 4 (2017). (cited on page 9)

ENGEL, J.; SCHÖPS, T.; AND CREMERS, D., 2014. LSD-SLAM: Large-scale direct monocular slam. In *European Conference on Computer Vision*, 834–849. Springer. (cited on pages 9 and 31)

EUSTICE, R. M.; PIZARRO, O.; AND SINGH, H., 2008. Visually augmented navigation for autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, 33, 2 (2008), 103–122. (cited on page 10)

FALANGA, D.; KLEBER, K.; AND SCARAMUZZA, D., 2020. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5, 40 (2020). (cited on page 90)

FISCHLER, M. A. AND BOLLES, R. C., 1981a. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24, 6 (1981), 381–395. (cited on page 14)

FISCHLER, M. A. AND BOLLES, R. C., 1981b. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24, 6 (1981), 381–395. (cited on pages 19 and 86)

FORSTER, C.; PIZZOLI, M.; AND SCARAMUZZA, D., 2014. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*. (cited on pages 2 and 31)

GAIDON, A.; WANG, Q.; CABON, Y.; AND VIG, E., 2016. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*. (cited on pages 47 and 55)

GÁLVEZ-LÓPEZ, D.; SALAS, M.; TARDÓS, J. D.; AND MONTIEL, J., 2016. Real-time monocular object slam. *Robotics and Autonomous Systems*, 75 (2016), 435–449. (cited on page 65)

GÁLVEZ-LÓPEZ, D. AND TARDOS, J. D., 2012. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28, 5 (2012), 1188–1197. (cited on page 35)

GEIGER, A.; LENZ, P.; STILLER, C.; AND URTASUN, R., 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32, 11 (2013), 1231–1237. (cited on pages 73 and 75)

GEIGER, A.; LENZ, P.; AND URTASUN, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 31, 38, 47, 55, 56, and 73)

GEIGER, A.; ZIEGLER, J.; AND STILLER, C., 2011. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*. (cited on pages 9 and 36)

GIRSHICK, R.; RADOSAVOVIC, I.; GKIOXARI, G.; DOLLÁR, P.; AND HE, K., 2018. Detectron. https://github.com/facebookresearch/detectron. (cited on page 65)

GODARD, C.; MAC AODHA, O.; FIRMAN, M.; AND BROSTOW, G. J., 2019. Digging into self-supervised monocular depth prediction. (October 2019). (cited on page 73)

GOLDSTEIN, H., 1980. *Classical Mechanics*. Addison-Wesley. (cited on page 18)

GUTMANN, J.-S. AND KONOLIGE, K., 1999. Incremental mapping of large cyclic environments. In *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No. 99EX375)*, 318–325. IEEE. (cited on page 7)

HAHNEL, D.; SCHULZ, D.; AND BURGARD, W., 2002. Map building with mobile robots in populated environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.*, vol. 1, 496–501. IEEE. (cited on page 63)

HAHNEL, D.; TRIEBEL, R.; BURGARD, W.; AND THRUN, S., 2003. Map building with mobile robots in dynamic environments. In *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA'03*, vol. 2, 1557–1563. IEEE. (cited on pages 11 and 63)

HARTLEY, R. AND ZISSERMAN, A., 2003. *Multiple view geometry in computer vision*. Cambridge university press. (cited on page 16)

HE, K.; GKIOXARI, G.; DOLLÁR, P.; AND GIRSHICK, R., 2017. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969. (cited on pages 48, 53, and 55)

HE, K.; GKIOXARI, G.; DOLLÁR, P.; AND GIRSHICK, R., 2018. Mask r-cnn. *IEEE transactions on pattern analysis and machine intelligence*, (2018). (cited on pages 65 and 73)

HE, K.; SUN, J.; AND TANG, X., 2011. Single image haze removal using dark channel prior. *IEEE transactions on pattern analysis and machine intelligence*, 33, 12 (2011), 2341–2353. (cited on page 42)

HENEIN, M.; ABELLO, M.; ILA, V.; AND MAHONY, R., 2017. Exploring the effect of meta-structural information on the global consistency of slam. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1616–1623. IEEE. (cited on pages 9 and 65)

HENEIN, M.; ZHANG, J.; MAHONY, R.; AND ILA, V., 2020. Dynamic slam: The need for speed. *2020 IEEE International Conference on Robotics and Automation (ICRA). To appear.*, (2020). (cited on pages 13 and 69)

HOSSEINZADEH, M.; LI, K.; LATIF, Y.; AND REID, I., 2019. Real-time monocular object-model aware sparse slam. In *2019 International Conference on Robotics and Automation (ICRA)*, 7123–7129. IEEE. (cited on pages 13 and 90)

HSIAO, M.; WESTMAN, E.; ZHANG, G.; AND KAESS, M., 2017. Keyframe-based dense planar SLAM. In *IEEE International Conference on Robotics and Automation (ICRA), 2017*, 5110–5117. IEEE. (cited on pages 9 and 65)

HUANG, A. S.; BACHRACH, A.; HENRY, P.; KRAININ, M.; MATURANA, D.; FOX, D.; AND ROY, N., 2017. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Robotics Research*, 235–252. Springer. (cited on page 9)

HUANG, J.; YANG, S.; MU, T.-J.; AND HU, S.-M., 2020. Clustervo: Clustering moving instances and estimating visual odometry for self and surroundings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2168–2177. (cited on pages 14 and 90)

HUI, T.-W.; TANG, X.; AND LOY, C. C., 2020. A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization. http://mmlab.ie.cuhk.edu.hk/projects/LiteFlowNet/. (cited on page 82)

ILA, V.; POLOK, L.; ŠOLONY, M.; AND SVOBODA, P., 2017. SLAM++-A highly efficient and temporally scalable incremental SLAM framework. *International Journal of Robotics Research*, Online First, 0 (2017), 1–21. doi:10.1177/0278364917691110. (cited on pages 29 and 69)

ILA, V.; PORTA, J. M.; AND ANDRADE-CETTO, J., 2010. Information-based compact pose SLAM. *IEEE Transactions on Robotics*, 26, 1 (2010), 78–93. (cited on page 82)

ILG, E.; MAYER, N.; SAIKIA, T.; KEUPER, M.; DOSOVITSKIY, A.; AND BROX, T., 2017. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. http://lmb.informatik.uni-freiburg.de//Publications/2017/IMKDB17. (cited on pages 48 and 64)

IRETA MUÑOZ, F. I. AND COMPORT, A. I., 2018. Point-to-hyperplane icp: fusing different metric measurements for pose estimation. *Advanced Robotics*, 32, 4 (2018), 161–175. (cited on page 10)

JIANG, H.; SUN, D.; JAMPANI, V.; LV, Z.; LEARNED-MILLER, E.; AND KAUTZ, J., 2019. Sense: A shared encoder network for scene-flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, 3195–3204. (cited on page 65)

JOHANNSSON, H.; KAESS, M.; ENGLOT, B.; HOVER, F.; AND LEONARD, J., 2010. Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In *2010 IEEE/RSJ*

International Conference on Intelligent Robots and Systems, 4396–4403. IEEE. (cited on page 10)

JUDD, K. M. AND GAMMELL, J. D., 2019. The oxford multimotion dataset: Multiple se (3) motions with ground truth. *IEEE Robotics and Automation Letters*, 4, 2 (2019), 800–807. (cited on pages 73 and 74)

JUDD, K. M.; GAMMELL, J. D.; AND NEWMAN, P., 2018. Multimotion visual odometry (mvo): Simultaneous estimation of camera and third-party motions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3949–3956. IEEE. (cited on pages xix, 12, 14, 73, 74, and 75)

KAESS, M., 2015. Simultaneous Localization and Mapping with Infinite Planes. In *IEEE International Conference on Robotics and Automation (ICRA), 2015*, 4605–4611. IEEE. (cited on page 65)

KAESS, M.; JOHANNSSON, H.; ROBERTS, R.; ILA, V.; LEONARD, J. J.; AND DELLAERT, F., 2011. iSAM2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, (2011), 0278364911430419. (cited on page 69)

KAVETI, P. AND SINGH, H., 2020. A Light Field Front-end for Robust SLAM in Dynamic Environments. *arXiv preprint arXiv:2012.10714*, (2020). (cited on page 12)

KE, T. AND ROUMELIOTIS, S. I., 2017. An efficient algebraic solution to the perspective-three-point problem. In *CVPR*. (cited on pages 53 and 71)

KIM, A. AND EUSTICE, R. M., 2013. Real-time visual slam for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics*, 29, 3 (2013), 719–733. (cited on page 10)

KLEIN, G. AND MURRAY, D., 2007. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 225–234. IEEE. (cited on page 7)

KNEIP, L.; FURGALE, P.; AND SIEGWART, R., 2013. Using multi-camera systems in robotics: Efficient solutions to the npnp problem. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 3770–3776. IEEE. (cited on page 33)

KNEIP, L.; SCARAMUZZA, D.; AND SIEGWART, R., 2011. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2969–2976. IEEE. (cited on page 19)

KSCHISCHANG, F. R.; FREY, B. J.; AND LOELIGER, H.-A., 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47, 2 (2001), 498–519. (cited on page 21)

KUENG, B.; MUEGGLER, E.; GALLEGO, G.; AND SCARAMUZZA, D., 2016. Low-latency visual odometry using event-based feature tracks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 16–23. IEEE. (cited on page 90)

KÜMMERLE, R.; GRISETTI, G.; STRASDAT, H.; KONOLIGE, K.; AND BURGARD, W., 2011. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 3607–3613. IEEE. (cited on pages 8, 29, 35, and 69)

KUNDU, A.; KRISHNA, K. M.; AND JAWAHAR, C., 2011. Realtime multibody visual slam with a smoothly moving monocular camera. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2080–2087. IEEE. (cited on pages 12 and 64)

LACROIX, S.; MALLET, A.; CHATILA, R.; AND GALLO, L., 1999. Rover self localization in planetary-like environments. In *Artificial Intelligence, Robotics and Automation in Space*, vol. 440, 433. (cited on page 7)

LATEGAHN, H.; GEIGER, A.; AND KITT, B., 2011. Visual slam for autonomous ground vehicles. In *2011 IEEE International Conference on Robotics and Automation*, 1732–1737. IEEE. (cited on page 3)

LEONARD, J. J. AND DURRANT-WHYTE, H. F., 1991a. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, 7, 3 (1991), 376–382. (cited on page 1)

LEONARD, J. J. AND DURRANT-WHYTE, H. F., 1991b. Simultaneous map building and localization for an autonomous mobile robot. In *IROS*, vol. 3, 1442–1447. (cited on page 7)

LEONARD, J. J. AND DURRANT-WHYTE, H. F., 2012. *Directed sonar sensing for mobile robot navigation*, vol. 175. Springer Science & Business Media. (cited on page 10)

LEONARD, J. J.; DURRANT-WHYTE, H. F.; AND COX, I. J., 1992. Dynamic map building for an autonomous mobile robot. *The International Journal of Robotics Research*, 11, 4 (1992), 286–298. (cited on page 20)

LEPETIT, V.; MORENO-NOGUER, F.; AND FUA, P., 2009. Epnp: An accurate O(n) solution to the pnp problem. *International journal of computer vision*, 81, 2 (2009), 155–166. (cited on page 19)

LEUTENEGGER, S.; LYNEN, S.; BOSSE, M.; SIEGWART, R.; AND FURGALE, P., 2015. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34, 3 (2015), 314–334. (cited on page 9)

LI, A. Q.; COSKUN, A.; DOHERTY, S. M.; GHASEMLOU, S.; JAGTAP, A. S.; MODASSHIR, M.; RAHMAN, S.; SINGH, A.; XANTHIDIS, M.; O'KANE, J. M.; AND REKLEITIS, I., 2016. Experimental comparison of open source vision based state estimation algorithms. In *Proc. International Symposium on Experimental Robotics*. (cited on page 32)

LI, P.; QIN, T.; ET AL., 2018. Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 646–661. (cited on page 13)

LI, P.; SHI, J.; AND SHEN, S., 2020. Joint spatial-temporal optimization for stereo 3d object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6877–6886. (cited on page 13)

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; AND ZITNICK, C. L., 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer. (cited on pages 56 and 73)

LIU, X.; QI, C. R.; AND GUIBAS, L. J., 2019. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 529–537. (cited on page 65)

LONGUET-HIGGINS, H. C., 1981. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293, 5828 (1981), 133–135. (cited on page 18)

LOWE, D. G., 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60, 2 (2004), 91–110. (cited on pages 10 and 40)

LV, Z.; KIM, K.; TROCCOLI, A.; REHG, J.; AND KAUTZ, J., 2018. Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation. In *ECCV*. (cited on pages 54 and 87)

MADSEN, K.; NIELSEN, H. B.; AND TINGLEFF, O., 2004. Methods for non-linear least squares problems. (2004). (cited on page 27)

MATTHIES, L. AND SHAFER, S., 1987. Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3, 3 (1987), 239–248. (cited on page 7)

MATTHIES, L. H., 1989. Dynamic stereo vision. (1989). (cited on page 7)

MAYBECK, P. S., 1982. *Stochastic models, estimation, and control*. Academic press. (cited on page 20)

MAYER, N.; ILG, E.; HAUSSER, P.; FISCHER, P.; CREMERS, D.; DOSOVITSKIY, A.; AND BROX, T., 2016. 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*, 4040–4048. (cited on pages 56 and 73)

MENZE, M. AND GEIGER, A., 2015. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3061–3070. (cited on page 65)

MILAN, A.; LEAL-TAIXÉ, L.; REID, I.; ROTH, S.; AND SCHINDLER, K., 2016. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, (Mar. 2016). http://arxiv.org/abs/1603.00831. ArXiv: 1603.00831. (cited on page 65)

MILFORD, M.; KIM, H.; LEUTENEGGER, S.; AND DAVISON, A., 2015. Towards visual slam with event-based cameras. In *The problem of mobile sensors workshop in conjunction with RSS*. (cited on page 90)

MILLER, I. AND CAMPBELL, M., 2007. Rao-blackwellized particle filtering for mapping dynamic environments. In *IEEE International Conference on Robotics and Automation, 2007*, 3862–3869. IEEE. (cited on page 64)

MOO YI, K.; TRULLS, E.; ONO, Y.; LEPETIT, V.; SALZMANN, M.; AND FUA, P., 2018. Learning to find good correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2666–2674. (cited on pages xviii, 86, and 87)

MORAVEC, H. P., 1980a. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, DTIC Document. (cited on page 2)

MORAVEC, H. P., 1980b. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, Stanford Univ Ca Dept of Computer Science. (cited on page 7)

MORENO-NOGUER, F.; LEPETIT, V.; AND FUA, P., 2007. Accurate non-iterative O(n) solution to the pnp problem. In *2007 IEEE 11th International Conference on Computer Vision*, 1–8. IEEE. (cited on page 19)

MU, B.; LIU, S.-Y.; PAULL, L.; LEONARD, J.; AND HOW, J. P., 2016. SLAM with objects using a nonparametric pose graph. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016*, 4602–4609. IEEE. (cited on page 65)

MUEGGLER, E.; GALLEGO, G.; REBECQ, H.; AND SCARAMUZZA, D., 2018. Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34, 6 (2018), 1425–1440. (cited on page 90)

MUR-ARTAL, R.; MONTIEL, J. M. M.; AND TARDOS, J. D., 2015. ORB-SLAM: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31, 5 (2015), 1147–1163. (cited on page 9)

MUR-ARTAL, R. AND TARDÓS, J. D., 2017. ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33, 5 (2017), 1255–1262. (cited on pages 31, 32, 33, 36, and 59)

NEWCOMBE, R. A.; IZADI, S.; HILLIGES, O.; MOLYNEAUX, D.; KIM, D.; DAVISON, A.; KOHI, P.; SHOTTON, J.; HODGES, S.; AND FITZGIBBON, A., 2011a. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, 127–136. IEEE. (cited on page 10)

NEWCOMBE, R. A.; LOVEGROVE, S. J.; AND DAVISON, A. J., 2011b. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, 2320–2327. IEEE. (cited on page 9)

NICHOLSON, L.; MILFORD, M.; AND SÜNDERHAUF, N., 2018a. Quadricslam: Dual quadrics as slam landmarks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 313–314. (cited on page 65)

NICHOLSON, L.; MILFORD, M.; AND SÜNDERHAUF, N., 2018b. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4, 1 (2018), 1–8. (cited on pages 13 and 90)

NISTÉR, D., 2004. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26, 6 (2004), 756–770. (cited on page 18)

NISTÉR, D.; NARODITSKY, O.; AND BERGEN, J., 2004. Visual odometry. In *null*, 652–659. IEEE. (cited on pages 1, 7, and 66)

NOCEDAL, J. AND WRIGHT, S., 2006. *Numerical optimization*. Springer Science & Business Media. (cited on page 24)

OLSON, C. F.; MATTHIES, L. H.; SCHOPPERS, H.; AND MAIMONE, M. W., 2000. Robust stereo ego-motion for long distance navigation. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, 453–458. IEEE. (cited on page 7)

POLOK, L.; ILA, V.; SOLONY, M.; SMRZ, P.; AND ZEMCIK, P., 2013. Incremental Block Cholesky Factorization for Nonlinear Least Squares in Robotics. In *Proceedings of Robotics: Science and Systems*. Berlin, Germany. doi:10.15607/RSS.2013.IX.042. (cited on page 69)

POMERLEAU, F.; COLAS, F.; SIEGWART, R.; AND MAGNENAT, S., 2013. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34, 3 (2013), 133–148. (cited on page 10)

POMERLEAU, F.; MAGNENAT, S.; COLAS, F.; LIU, M.; AND SIEGWART, R., 2011. Tracking a depth camera: Parameter exploration for fast icp. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3824–3829. IEEE. (cited on page 10)

PUMAROLA, A.; VAKHITOV, A.; AGUDO, A.; SANFELIU, A.; AND MORENO-NOGUER, F., 2017. Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE international conference on robotics and automation (ICRA)*, 4503–4508. IEEE. (cited on page 9)

QIN, T.; LI, P.; AND SHEN, S., 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34, 4 (2018), 1004–1020. (cited on page 2)

REBECQ, H.; HORSTSCHAEFER, T.; AND SCARAMUZZA, D., 2017. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In *British Machine Vision Conference (BMVC)*, CONF. (cited on page 90)

REDDY, N. D.; ABBASNEJAD, I.; REDDY, S.; MONDAL, A. K.; AND DEVALLA, V., 2016. Incremental real-time multibody vslam with trajectory optimization using stereo camera. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 4505–4510. IEEE. (cited on page 90)

REDDY, N. D.; SINGHAL, P.; CHARI, V.; AND KRISHNA, K. M., 2015. Dynamic body VSLAM with semantic constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015*, 1897–1904. IEEE. (cited on page 12)

REN, S.; HE, K.; GIRSHICK, R.; AND SUN, J., 2016. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39, 6 (2016), 1137–1149. (cited on page 53)

RIGBY, P.; PIZARRO, O.; AND WILLIAMS, S. B., 2006. Towards geo-referenced auv navigation through fusion of usbl and dvl measurements. In *OCEANS 2006*, 1–6. IEEE. (cited on page 10)

ROGERS, J. G.; TREVOR, A. J.; NIETO-GRANDA, C.; AND CHRISTENSEN, H. I., 2010. SLAM with expectation maximization for moveable object tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010*, 2077–2082. IEEE. (cited on page 64)

ROS, G.; SAPPA, A.; PONSA, D.; AND LOPEZ, A. M., 2012. Visual slam for driverless cars: A brief survey. In *Intelligent Vehicles Symposium (IV) Workshops*, vol. 2. (cited on page 3)

ROSTEN, E. AND DRUMMOND, T., 2006. Machine learning for high-speed corner detection. In *European conference on computer vision*, 430–443. Springer. (cited on pages 40, 56, and 73)

RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; AND BRADSKI, G. R., 2011. ORB: An efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, (2011), 2564–2571. (cited on pages 40 and 73)

RÜNZ, M. AND AGAPITO, L., 2017. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 4471–4478. IEEE. (cited on page 90)

RUNZ, M.; BUFFIER, M.; AND AGAPITO, L., 2018. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 10–20. IEEE. (cited on pages 2, 13, and 90)

SALAS-MORENO, R. F.; NEWCOMBE, R. A.; STRASDAT, H.; KELLY, P. H.; AND DAVISON, A. J., 2013. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013*, 1352–1359. IEEE. (cited on pages 13 and 65)

SALVI, J.; PETILLO, Y.; THOMAS, S.; AND AULINAS, J., 2008. Visual slam for underwater vehicles using video velocity log and natural landmarks. In *OCEANS 2008*, 1–6. IEEE. (cited on page 10)

SCARAMUZZA, D. AND FRAUNDORFER, F., 2011. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18, 4 (2011), 80–92. (cited on page 47)

SHKURTI, F.; REKLEITIS, I.; SCACCIA, M.; AND DUDEK, G., 2011. State estimation of an underwater robot using visual and inertial information. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5054–5060. IEEE. (cited on page 10)

SMITH, R.; SELF, M.; AND CHEESEMAN, P., 1990. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, 167–193. Springer. (cited on page 20)

SNYDER, J., 2010. Doppler velocity log (dvl) navigation for observation-class rovs. In *OCEANS 2010 MTS/IEEE SEATTLE*, 1–9. IEEE. (cited on page 10)

SOLA, J.; VIDAL-CALLEJA, T.; CIVERA, J.; AND MONTIEL, J. M. M., 2012. Impact of landmark parametrization on monocular ekf-slam with points and lines. *International journal of computer vision*, 97, 3 (2012), 339–368. (cited on page 9)

SRINIVASAN, M.; ZHANG, S.; AND BIDWELL, N., 1997. Visually mediated odometry in honeybees. *Journal of Experimental Biology*, 200, 19 (1997), 2513–2522. (cited on page 7)

STOFFREGEN, T.; GALLEGO, G.; DRUMMOND, T.; KLEEMAN, L.; AND SCARAMUZZA, D., 2019. Event-based motion segmentation by motion compensation. In *Proceedings of the IEEE International Conference on Computer Vision*, 7244–7253. (cited on page 90)

STRASDAT, H.; DAVISON, A. J.; MONTIEL, J. M.; AND KONOLIGE, K., 2011. Double window optimisation for constant time visual SLAM. In *IEEE International Conference on Computer Vision (ICCV), 2011*, 2352–2359. IEEE. (cited on page 82)

STURM, J.; ENGELHARD, N.; ENDRES, F.; BURGARD, W.; AND CREMERS, D., 2012. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 573–580. IEEE. (cited on page 38)

SUCAR, E.; WADA, K.; AND DAVISON, A., 2020. Nodeslam: Neural object descriptors for multi-view shape reconstruction. *arXiv preprint arXiv:2004.04485*, (2020). (cited on page 13)

SUN, D.; ROTH, S.; AND BLACK, M. J., 2010. Secrets of optical flow estimation and their principles. In *2010 IEEE computer society conference on computer vision and pattern recognition*, 2432–2439. IEEE. (cited on page 64)

SUN, D.; ROTH, S.; AND BLACK, M. J., 2014. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106, 2 (2014), 115–137. (cited on page 56)

SUN, D.; YANG, X.; LIU, M.-Y.; AND KAUTZ, J., 2018. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. (cited on pages 48, 53, 56, 64, and 73)

TAN, W.; LIU, H.; DONG, Z.; ZHANG, G.; AND BAO, H., 2013. Robust monocular slam in dynamic environments. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, 209–218. IEEE. (cited on pages 11 and 47)

TANIAI, T.; SINHA, S. N.; AND SATO, Y., 2017. Fast Multi-frame Stereo Scene Flow with Motion Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6891–6900. (cited on page 87)

TATENO, K.; TOMBARI, F.; AND NAVAB, N., 2016. When 2.5D is not enough: Simultaneous reconstruction, segmentation and recognition on dense SLAM. In *2016 IEEE international conference on robotics and automation (ICRA)*, 2295–2302. IEEE. (cited on page 13)

THRUN, S.; BURGARD, W.; AND FOX, D., 1998. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5, 3-4 (1998), 253–271. (cited on page 7)

VIDAL, A. R.; REBECQ, H.; HORSTSCHAEFER, T.; AND SCARAMUZZA, D., 2018. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3, 2 (2018), 994–1001. (cited on page 90)

VOGEL, C.; SCHINDLER, K.; AND ROTH, S., 2013. Piecewise rigid scene flow. In *Proceedings of the IEEE International Conference on Computer Vision*, 1377–1384. (cited on page 65)

WANG, C.-C.; THORPE, C.; AND THRUN, S., 2003. 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, 842–849. IEEE. (cited on page 64)

WANG, C.-C.; THORPE, C.; THRUN, S.; HEBERT, M.; AND DURRANT-WHYTE, H., 2007. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26, 9 (2007), 889–916. (cited on pages 12, 65, and 69)

WEIKERSDORFER, D.; ADRIAN, D. B.; CREMERS, D.; AND CONRADT, J., 2014. Event-based 3d slam with a depth-augmented dynamic vision sensor. In *2014 IEEE international conference on robotics and automation (ICRA)*, 359–364. IEEE. (cited on page 90)

WHELAN, T.; KAESS, M.; FALLON, M.; JOHANNSSON, H.; LEONARD, J.; AND MCDONALD, J., 2012. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*. Sydney, Australia. (cited on page 10)

WHELAN, T.; LEUTENEGGER, S.; SALAS-MORENO, R.; GLOCKER, B.; AND DAVISON, A., 2015. ElasticFusion: Dense SLAM without a pose graph. (cited on page 10)

WIRTH, S.; CARRASCO, P. L. N.; AND CODINA, G. O., 2013. Visual odometry for autonomous underwater vehicles. In *OCEANS-Bergen, 2013 MTS/IEEE*, 1–6. IEEE. (cited on page 2)

WOHLHART, P. AND LEPETIT, V., 2015. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3109–3118. (cited on page 65)

WOLF, D. F. AND SUKHATME, G. S., 2005. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19, 1 (2005), 53–65. (cited on page 63)

WRIGHT, S. AND NOCEDAL, J., 1999. Numerical optimization. *Springer Science*, 35, 67-68 (1999), 7. (cited on page 26)

WU, X.; STUCK, R. E.; REKLEITIS, I.; AND BEER, J. M., 2015. Towards a framework for human factors in underwater robotics. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 59, 1115–1119. SAGE Publications Sage CA: Los Angeles, CA. (cited on page 10)

XU, B.; LI, W.; TZOUMANIKAS, D.; BLOESCH, M.; DAVISON, A.; AND LEUTENEGGER, S., 2019. Mid-fusion: Octree-based object-level multi-instance dynamic slam. In *2019 International Conference on Robotics and Automation (ICRA)*, 5231–5237. IEEE. (cited on pages 13 and 90)

YAMAGUCHI, K.; MCALLESTER, D.; AND URTASUN, R., 2014. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision*, 756–771. Springer. (cited on pages 52, 64, and 70)

YANG, S. AND SCHERER, S., 2019. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, (2019). (cited on pages xvii, xix, 13, 65, 73, 75, 76, 77, and 90)

YI, K. M.; TRULLS, E.; LEPETIT, V.; AND FUA, P., 2016. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, 467–483. Springer. (cited on page 86)

ZHANG, Z., 1998. Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision*, 27, 2 (1998), 161–195. (cited on page 18)

ZHAO, H.; CHIBA, M.; SHIBASAKI, R.; SHAO, X.; CUI, J.; AND ZHA, H., 2008. SLAM in a dynamic large outdoor environment using a laser scanner. In *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.*, 1455–1462. IEEE. (cited on page 63)

ZHOU, Y.; LI, H.; AND KNEIP, L., 2018. Canny-vo: Visual odometry with rgb-d cameras based on geometric 3-d–2-d edge alignment. *IEEE Transactions on Robotics*, 35, 1 (2018), 184–199. (cited on page 9)

ZOU, D. AND TAN, P., 2013. Coslam: Collaborative visual slam in dynamic environments. *IEEE transactions on pattern analysis and machine intelligence*, 35, 2 (2013), 354–366. (cited on page 90)

ZUIDERVELD, K., 1994. Contrast limited adaptive histogram equalization. In *Graphics gems IV*, 474–485. Academic Press Professional, Inc. (cited on page 42)