# Dense Visual Simultaneous Localisation and Mapping in Collaborative and Outdoor Scenarios

**Louis Patrick Gallagher**

A dissertation submitted for the degree of
*Doctor of Philosophy*


Department of Computer Science

Faculty of Science and Engineering

Maynooth University

August, 2023

Head of Department: Dr Joseph Timoney
Supervisor: Prof. John B. McDonald

# Contents

# List of Tables

# List of Figures

# List of Algorithms

# Abstract

Dense visual simultaneous localisation and mapping (SLAM) systems can produce $3D$ reconstructions that are digital facsimiles of the physical space they describe. Systems that can produce dense maps with this level of fidelity in real time provide foundational spatial reasoning capabilities for many downstream tasks in autonomous robotics. Over the past 15 years, mapping small scale, indoor environments, such as desks and buildings, with a single slow moving, hand-held sensor has been one of the central focuses of dense visual SLAM research.

However, most dense visual SLAM systems exhibit a number of limitations which mean they cannot be directly applied in collaborative or outdoors settings. The contribution of this thesis is to address these limitations with the development of new systems and algorithms for collaborative dense mapping, efficient dense alternation and outdoors operation with fast camera motion and wide field of view (FOV) cameras. We use ElasticFusion, a state-of-the-art dense SLAM system, as our starting point where each of these contributions is implemented as a novel extension to the system.

We first present a collaborative dense SLAM system that allows a number of cameras starting with unknown initial relative positions to maintain local maps with the original ElasticFusion algorithm. Visual place recognition across local maps results in constraints that allow maps to be aligned into a common global reference frame, facilitating collaborative mapping and tracking of multiple cameras within a shared map.

Within dense alternation based SLAM systems, the standard approach is to fuse every frame into the dense model without considering whether the information contained within the frame is already captured by the dense map and therefore redundant. As the number of cameras or the scale of the map increases, this approach becomes inefficient. In our second contribution, we address this inefficiency by introducing a novel information theoretic approach to keyframe selection that allows the system to avoid processing redundant information. We implement the procedure within ElasticFusion, demonstrating a marked reduction in the number of frames required by the system to estimate an accurate, denoised surface reconstruction.

Before dense SLAM techniques can be applied in outdoor scenarios we must first address their reliance on active depth cameras, and their lack of suitability to fast camera motion. In our third contribution we present an outdoor dense SLAM system. The

system overcomes the need for an active sensor by employing neural network-based depth inference to predict the geometry of the scene as it appears in each image. To address the issue of camera tracking during fast motion we employ a hybrid architecture, combining elements of both dense and sparse SLAM systems to perform camera tracking and to achieve globally consistent dense mapping.

Automotive applications present a particularly important setting for dense visual SLAM systems. Such applications are characterised by their use of wide FOV cameras and are therefore not accurately modelled by the standard pinhole camera model. The fourth contribution of this thesis is to extend the above hybrid sparse-dense monocular SLAM system to cater for large FOV fisheye imagery. This is achieved by reformulating the mapping pipeline in terms of the Kannala-Brandt fisheye camera model. To estimate depth, we introduce a new version of the PackNet depth estimation neural network (Guizilini et al., 2020) adapted for fisheye inputs.

To demonstrate the effectiveness of our contributions, we present experimental results, computed by processing the synthetic ICL-NUIM dataset of Handa et al. (2014) as well as the real-world TUM-RGBD dataset of Sturm et al. (2012). For outdoor SLAM we show the results of our system processing the autonomous driving KITTI and KITTI-360 datasets of Geiger et al. (2012a) and Liao et al. (2021) respectively.

# Acknowledgements

I would like to express my gratitude to my adviser, John McDonald. At the start of my PhD, after a few months of scratching my head I went to his office and asked "how does one actually do research?" Though I never quite figured out the answer to that question, his deep knowledge of robotics and his ability to bring ideas to life gave me something to aim for. Without his steadfast guidance and enthusiasm this thesis would not have been possible. The many tangential, blue-sky conversations we found ourselves having has given me much to think about for years to come.

I was lucky to be surrounded by a number of people who added a layer of *texture* to my time at Maynooth, and without whom the PhD would have been a much blander experience. I would like to express my gratitude to my lab mates and colleagues Will, Toby, Rob, Sarav, Niamh, James, Liam, Conor, Marie, and Keith. I am very grateful for your friendship and support and that I got to share my time in Maynooth with you all. Postgraduate seminars, research rubber-ducking and coffee breaks that went on for far too long, without these things we would have just been people doing an unhealthy amount of staring at computer screens. I would also like to express my gratitude to my housemate Jake. Going to Judo, chatting over cups of tea in the evening and, during the pandemic, our garden workouts, reminded me that life outside the lab continues.

This research was funded by the Irish Research Council under the IRC GOIPG Scholarship scheme 2016 (GOIPG/2016/1320). At a critical point, this research also received funding from the HEA COVID extension fund. I am grateful to both these bodies for the funding they provided.

Finally I would like to acknowledge the silent co-authors of my work. I would like to thank Mira for her patience, benign neglect and support during what, at the time, seemed like an endless, long-tailed end to my PhD. I am grateful to you for listening to my SLAM monologues, you almost certainly know more about SLAM than you would otherwise care to, but, it helped unknot the tangled SLAM mess in my head. Finally, I would like to thank my Mum and Dad, my two sisters, Amy and Daisy, and my brother, Frank. Heading back home for the occasional loop closure kept me moving forward during the long years of this PhD.

# CHAPTER 1

# Introduction

To build autonomous robots - robots that can think and act on their own - requires that we also build robots with the capability for perception. In the context of robotics, perception is the automatic analysis and structuring of sensor data into a model of the world. A world model makes explicit certain useful information implicitly encoded within the sensor data, for example the location and geometry of surfaces in the scene (Marr, 1982), that, once acquired, makes advanced forms of autonomy, like independent path planning in unknown environments, possible. The foundation of advanced perception, and, indeed, advanced autonomy, is the ability to build a geometric map of an environment and to simultaneously localise relative to that map. This problem is known as *simultaneous localisation and mapping* (SLAM).

SLAM derives its name from the problem's circular nature. When performing SLAM, a robot attempts to integrate local, relative measurements from onboard sensors into a global map of the environment. However, in order to integrate each local measurement, the precise location of the robot, relative to the map, is required. In SLAM, the map is a representation of the geometry of the scene in a global coordinate system, though some works represent the scene as a series of relative locations without a single privileged global coordinate frame (e.g., Mei et al., 2010). Other perceptual information can be layered on top of the map's geometric base layer such as colour, semantics, object boundaries and so forth.

The interdependence between mapping and localisation is one of SLAM's features

that makes it a challenging problem to solve. Given perfect prior knowledge of a robot's trajectory as it explores an environment, an optimal map that explains the robot's sensor readings can be computed. Similarly, a precise map of an environment's geometry, known prior to exploration, can be used to track the robot's motion. However, when a robot is tasked with exploring an a priori unknown environment, neither map nor camera trajectory are, by definition, known in advance, and the full SLAM problem must be solved. In this case, simply alternating between tracking the robot's motion and updating the map does not account for correlations between the estimated robot trajectory and environment map caused by estimation errors. When a sensor measurement is integrated into the map using an estimate of the robot's position that contains a quantity of error, then the robot position estimate and the portion of the map related to that update become correlated. This has led to the development of joint estimation frameworks based on probabilistic filtering (e.g., Leonard and Durrant-Whyte, 1991), and smoothing (e.g., Dellaert, 2005), that take into account correlations between the estimated robot pose and map.

When the robot's sensor is a camera, SLAM is referred to as *visual SLAM* (VSLAM). Initial developments within VSLAM research focused on extracting a small number of robust, salient features in each image for camera tracking and map building. This approach allowed systems to estimate sparse maps consisting of collections of geometric primitives such as points and lines (e.g., Davison, 2003; Klein and Murray, 2007, 2008). Although the sparse maps estimated by these systems provided a sufficient level of detail to facilitate accurate camera tracking, they were ultimately limited in their ability to capture the full extent of the geometry of the scene being explored.

Driven by the limited representational power of sparse maps, as well as advances in general purpose GPU (GPGPU) technology and the availability of commodity depth sensors such as the Microsoft Kinect, dense methods have become a central focus of VSLAM research. In contrast to feature-based systems, dense systems attempt to use as many of the pixels in each new camera frame as possible for camera tracking and map building. A prominent feature of dense systems is a direct approach to camera tracking, where tracking is performed directly on image intensities rather than on extracted features.

However, what really distinguishes dense systems from their sparse counterparts, is that they attempt to model the geometry of the underlying continuous surface in a dense fashion. In some dense systems this is achieved by estimating a collection of geometric primitives that, while densely populated and high resolution, ultimately represent a discrete sampling of the scene's surface (e.g., Henry et al., 2012; Keller et al., 2013; Stühmer et al., 2010). In contrast, other dense systems attempt to estimate the topological structure of the scene's surface (e.g., Dai et al., 2017; Newcombe and Davison, 2010; Newcombe et al., 2011a; Schöps et al., 2020).

## 1.1 Dense SLAM in Constrained Environments

SLAM has become a mature research field in recent years. So much so that the question of whether SLAM is solved is a legitimate question that requires considerable knowledge of the field to answer (Cadena et al., 2016; Frese, 2010). Naturally, the act of trying to answer this question has brought into sharp focus what is, and what is not, solved in SLAM. In their answer to the "is SLAM solved?" question, Cadena et al. (2016) argue that SLAM is only solved for specific combinations of robots, environments and performance requirements, e.g., $2D$ SLAM in a static indoor environment with a laser scanner. When considered in the context of what is required for long-term autonomous function of a robot, it is evident that for many other combinations of robot/environment/performance, such as mapping highly dynamic environments with a fast moving camera, SLAM remains largely unsolved (Cadena et al., 2016).

A particular case in point is dense visual SLAM, where indoor mapping with a single RGBD sensor undergoing slow hand-held camera motion has been the focus of research. This relatively constrained scenario has played an important, if not necessary, role in the development of initial solutions to dense VSLAM by reducing the complexity of the problem being faced. Consider each component of the scenario in turn:

**Indoor mapping:** Environmental factors that create challenges for SLAM systems, such as variable lighting and scene dynamics, are easier to control or eliminate in indoor

environments. The scale of the space being mapped is also an important factor. Mapping large-scale outdoor spaces, such as towns and suburbs (e.g., Geiger et al., 2013; Liao et al., 2021), presents challenges, in terms of system scalability and efficiency, beyond what is involved in mapping room and building sized interior spaces (e.g., Handa et al., 2014; Sturm et al., 2012).

**Single sensor:** Although the framing of the SLAM problem given in the previous section considers only a single robot operating in isolation, the problem definition can be broadened to include multiple robots localising against, and contributing to the construction of, a shared map. As will become evident through Chapters 1 to 3 of this thesis, solving single-sensor SLAM is, to a large degree, a prerequisite to solving the multi-sensor SLAM problem. Therefore, it is reasonable that early developments in dense visual SLAM have focused on solving the single-sensor SLAM problem.

**RGBD sensor:** Prior to the widespread availability of active depth sensors, sparse visual SLAM systems relied on multi-view stereo techniques to estimate scene geometry themselves by integrating information from the camera as it moved (e.g., Davison et al., 2007; Klein and Murray, 2007). Estimating dense scene geometry at frame-rate with this approach proved to be challenging (e.g., Newcombe and Davison, 2010; Newcombe et al., 2011b; Stühmer et al., 2010). In addition, the overall scale of the scene cannot be estimated by a monocular camera which limits the usefulness of the resultant map in real-world applications that involve physical interactions with the environment. The introduction of active depth sensors, such as the Microsoft Kinect, simplified the SLAM process by providing accurate, dense, metric-scale estimates of the scene's geometry at frame-rate.

**Slow, hand-held motion:** The relatively slow motion of a hand-held camera results in small amounts of inter-frame displacement between subsequent frames which is favourable to SLAM performance. The reasons for this are manyfold and will be discussed at length in Section 5.1. However, for now, it suffices to say that, by moving slowly, sensor data is less affected by processes that hinder SLAM performance, such as rolling-shutter and motion-blurring (see for example Tourani et al., 2016), and estimating important quantities,

such as camera motion and scene geometry, is rendered much simpler.

Indoor mapping with a slow moving, hand-held RGBD camera has been doubly important in developing dense SLAM solutions for real-world applications. Firstly, this scenario is itself an important domain of application, with many potential use cases across robotics, augmented reality and $3D$ reconstruction involving such conditions. Secondly, the dense SLAM solutions developed for this domain have provided the basis for further developments under less constrained conditions. Of this second point, we will have much more to say throughout the remainder of the thesis.

## 1.2 Motivation

Although important in the development of initial solutions to dense visual SLAM, indoor mapping with a slow moving hand-held RGBD camera represents only a small fraction of the possible conditions that a robot capable of long-term autonomy will be expected to operate in. However, as a result of focusing on this scenario, some or all of the conditions listed in Section 1.1 have been built into the particular algorithms and techniques employed by many state-of-the-art dense visual SLAM systems, inhibiting their application under different conditions. The motivation for this thesis is to address the limited scope of dense visual SLAM, so that its capabilities may be brought to bear on a number of new domains including collaborative and outdoor scenarios. We propose to do this in four ways:

**From Single-Sensor to Collaborative:** Much of the developments within dense SLAM have focused on scenarios where a single sensor must map the environment in isolation. However, in reality it is often the case that multiple mobile agents must work together in the same space. In order to collaborate on a task, each robot requires a map of the world around it and knowledge of how it, and its collaborators, are positioned relative to that map. While it would be possible for each robot to solve this problem independently, the question that naturally arises is *how can mobile agents collaborate in the construction of a shared map?* This is the problem of *multiagent* or *collaborative* SLAM. An early motivation for the work in this thesis was to begin to address the limited scope of dense visual SLAM

through the design and implementation of a collaborative dense visual SLAM system, i.e. to design a system that overturns the single-sensor limitation outlined above. Cooperative robotics, multi participant augmented reality and human-robot interaction are all examples of situations where a collaborative mapping system can be leveraged for greater agent autonomy.

**From Every-Frame to Informative Keyframes:** Scalability is a further motivation for the work in this thesis. Scalability is the problem of ensuring that the SLAM system continues to perform accurately and in real time as the scale of the space being mapped increases, or as the level of detail with which the space is represented increases. More broadly, scalability overlaps with the concepts of resource-awareness, i.e. the idea that the SLAM system can modulate the complexity of the map representation based on available resources, and task-driven perception, i.e. the idea that the SLAM system can filter out perceptual information that is irrelevant to the task at hand (Cadena et al., 2016). Our research in this direction is motivated by the following observation: many dense visual SLAM systems operate on the assumption that every pixel in every frame should be used to produce as accurate and as dense a reconstruction as possible. While such a data hungry approach may not present a problem when mapping a room or building size space, when mapping in large-scale environments, or within a collaborative setting with many cameras, it may not be tenable, or even desirable, to do so. We investigate an alternative, information-based approach that allows a surfel-based dense visual SLAM system to first quantify the amount of novel information in each incoming frame with respect to the map and then discard uninformative frames. This capability allows for map accuracy and completeness to be traded-off, in a principled manner, against the computational cost incurred by updating the map.

**From Indoor to Outdoor:** The use of active depth sensors and direct camera tracking methods limits, for the most part, the application of dense visual SLAM systems to indoor environments and relatively slow camera movement. However, dense mapping systems, and their resultant high fidelity $3D$ reconstructions, could potentially confer many benefits to autonomous driving and other outdoors robotics applications. Autonomous driving

in particular acts as a motivating context for the work contained in Chapters 5 and 6 of this thesis. The conditions present in autonomous driving are somewhat antithetical to those found while mapping an indoor space with a hand-held RGBD camera, i.e. we can expect fast camera motion, global illumination changes in the scene and large-scale, outdoor spaces. Before dense visual SLAM systems can be used effectively in autonomous driving, the limitations induced by the sensors and algorithms employed in dense mapping need to be addressed. A further motivation for the work in this thesis is to overcome these limitations through the design of dense SLAM systems that can operate in outdoor autonomous driving scenarios.

**From Pinhole to Fisheye:** In automotive applications, a single monocular camera with a wide viewing angle offers many potential benefits. Fisheye cameras have become increasingly popular given their wide field of view allowing a much larger portion of the scene to remain in view of the camera at any one time. Surround view sensing is essential in applications where sensory blind spots are a safety hazard (Hughes et al., 2009). It is important to note that increasing the FOV of an imaging device while holding all other characteristics fixed decreases the angular resolution of a pixel and increases the level of distortion in the image. When it comes to localisation and mapping, the loss of angular resolution and degree of distortion can have an effect on performance (Zhang et al., 2016). However, when taking a holistic view of a robot platform, such as an autonomous car, the benefits of the increased FOV to downstream safety critical modules may outweigh the negative impact on SLAM performance. Thus, there is a strong need for a robust dense SLAM system that can cater for fisheye camera geometries. The fourth and final motivation for the work in this thesis is the design and implementation of a dense visual SLAM system that supports operation with wide FOV fisheye cameras.

## 1.3 VSLAM and Its Applications

There are increasing examples of technologies that can operate autonomously and with spatial-awareness due to the perceptual capabilities built into them. In this section we give

a brief overview of some key advancements in VSLAM research and the applications they have enabled.

## Robotic Space Exploration

One of the earliest, and most well known, efforts to incorporate spatial awareness into an autonomous robotic platform began in the 1960s and 1970s when NASA's Jet Propulsion Laboratory began conducting exploratory research into remote controlled rover designs for future unmanned missions to Mars. The inherent low bandwidth and high latency nature of inter-planetary communications meant that remote manual control of rover functions would be neither efficient nor precise, as real-time decision making based on up-to-date information would not be possible. At best, the rover would be commanded intermittently, once per Martian sol. At the end of the day the rover would upload sensor and telemetry data for planning the next sol's activities. Any rover sent to Mars would require semi-autonomous navigation and perception capabilities. The rover would be required to operate autonomously between remote commands in order to avoid obstacles, correct for wheel slippage and interact with rock samples. Ultimately, for the rover to be able to complete its mission and carry out science experiments, it would have to be able to automatically understand the shape and $3D$ position of rock samples and other objects, and its motion relative to these parts of the environment, such that it could manipulate them (O'Handley, 1973).

This work culminated with the Mars Exploration Rover (MER) mission which saw stereo visual odometry (VO) deployed on an interplanetary rover for the first time. Added as an 'extra credit' capability, VO was not initially relied on for precision control of the rover. However, VO soon proved invaluable for correcting for wheel odometry errors caused by the wheels slipping on certain types of terrain. This effectively increased the rover's speed and precision, allowing the operations team to command longer traverses, knowing the rover would arrive close to the target (Maimone et al., 2007). Figure 1.1 shows the rovers deployed to Mars during the MER mission. Figure 1.2 shows outputs from the MER stereo vision pipeline.

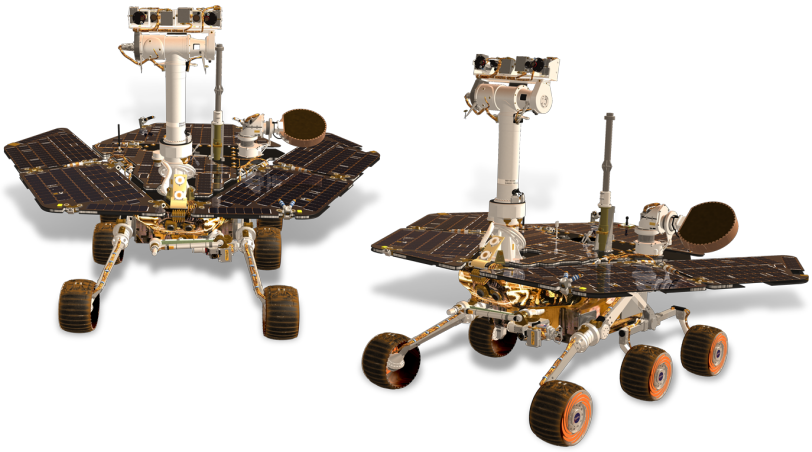
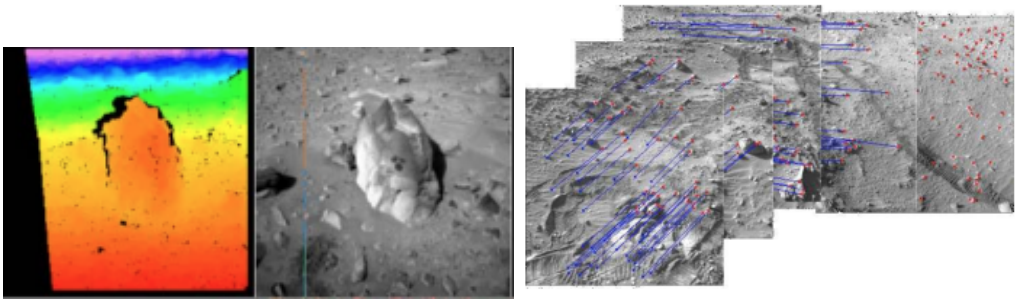


Figure 1.1: Twin rovers, Spirit and Opportunity, deployed to Mars during the MER mission. The two cameras at the top of each rover's central mast are set up in a stereo configuration and used for calculating visual odometry NASA (2022).

The *stop-process-move-repeat* approach to exploratory robotics was permissible for the MER mission. The mission specification was not very demanding in terms of minimum driving distance per sol and the barren surface of Mars is, for the most part, not a particularly dynamic environment. This was convenient as these early stereo VO techniques were not real-time and not very robust. Each *stop-process-move* sequence required $2-3$ minutes of VO computation time (Maimone et al., 2007). For a more expansive discussion of the early history of computer vision research for Mars rovers see Matthies et al. (2007).

Contemporaneously to early Mars rover research, mobile robotics with more terrestrial aims such as self-driving vehicles and deployment in challenging conditions were being pursued (e.g., Bares and Wettergreen, 1999; Pomerleau and Jochem, 1996; Thorpe et al., 1991a,b; Wettergreen et al., 1993). During this period, it became apparent that for a robot to be deployed in unknown environments, it would need to be capable of performing SLAM, i.e. it would need to be capable of building a persistent, globally consistent map of its surroundings in real time (Smith et al., 1990; Smith and Cheeseman, 1986). Although early work on SLAM focused on active ranging sensors such as sonar and LiDAR, SLAM methods based on visual sensors have become a central focus of SLAM research.



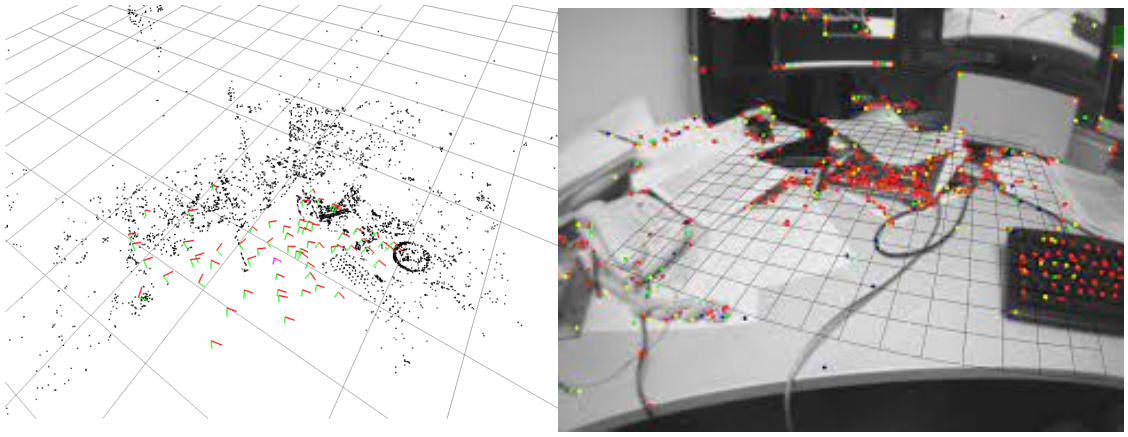
(a) (b)

Figure 1.2: Examples of outputs from the MER stereo system. Figure (a) adapted from Matthies et al. (2007) showing a dense stereo disparity map from which range information can be calculated. Figure (b) adapted from Maimone et al. (2007) showing stereo feature tracking between pairs of frames. The blue lines and red dots show features matched between successive frames.

## Augmented Reality

In an offline setting, VSLAM from a set of images can be formulated as a *bundle adjustment* (Triggs et al., 2000). The goal of bundle adjustment (called *Structure from Motion* in computer vision) is to estimate a set of $3D$ world points and a set of camera matrices that minimises the distance (i.e. error) between world points projected back onto the image plane of each camera as described by their respective camera matrices, and their corresponding measured $2D$ image locations. Estimates of the $3D$ points and cameras can be iteratively refined within a non-linear optimisation framework until the reprojection error is considered sufficiently low. Under the assumption that the measured locations of the $3D$ points in the images are perturbed by noise with a *Gaussian distribution*, bundle adjustment can lead to a maximum likelihood reconstruction with respect to the world points and camera matrices. Hartley and Zisserman (2003); Szeliski (2010); Triggs et al. (2000) offer quite expansive discussion on this topic.

VSLAM and SfM are essentially the same problem, the principle difference being that VSLAM requires real-time operation, which is necessary in many robotics applications. Although SfM allows us to gain some purchase on the seemingly intractable joint estimation of structure and motion underlying the SLAM problem, its emphasis on accurate reconstructions over run-time efficiency limits its usefulness as a paradigm for solving SLAM in real-time applications. For example, COLMAP, a modern state-of-



(a) (b)

Figure 1.3: **Outputs of PTAM.** (a) A sparse map. The red and green axes represent keyframe camera poses. Black dots represent $3D$ points in the sparse map. (b) The map projected into the current camera frame. Note the sparsity with which the map represents the desk. Figure adapted from Klein and Murray (2007).

the-art SfM system, takes over 3 hours to reconstruct a scene with 74394 images which equates to a frame-rate of $6.8hz$, well below the $20-30hz$ frame-rate of a standard camera (Schönberger and Frahm, 2016). Importantly, SfM systems typically require all frames up front before it can compute its solution. This further limits the use of SfM to offline and batch, rather than real time, settings. While in some robotic applications maps can be constructed offline using SfM and then used by a robot to localise and navigate in real time, in other applications such as exploratory robotics, mapping must be performed in real time. For example, during a natural or man-made disaster, prior maps are either not available or cannot be safely retrieved, and, even if they can be, may quickly become inaccurate. In this instance one could again exploit SfM by performing bundle adjustment over the frames accrued so far in the robot's exploration, allowing up-to-date maximum likelihood estimates of both a map and a set of camera poses, one for each frame. However, in practice the rapid growth in the number of points and camera poses that need to be estimated renders such an approach infeasible when real-time operation is required. This is a central requirement of SLAM algorithms given that they must be capable of maintaining real-time operation, even at large-scales.

As such, much of the focus within the field of visual SLAM has been on increasing the robustness, accuracy and extensiveness of mapping systems while keeping the practi-

calities and constraints of real-time performance in mind. Initial approaches to VSLAM utilised probabilistic frameworks such as Bayesian filtering and incremental graphical model smoothing as underlying state estimators, allowing the construction of sparse maps consisting of geometric primitives such as points and lines. Systems such as MonoSLAM (Davison et al., 2007) and Parallel Tracking and Mapping (PTAM, Klein and Murray, 2007) are good examples of the former and latter approaches to SLAM respectively. In order to maintain real-time operation, these frameworks *sparsify* the problem, each in their own way. Filtering approaches marginalise out previous poses, representing the map and the most recent pose estimate with a joint probability distribution conditioned on motion constraints and feature measurements. Smoothing approaches solve a sparse bundle adjustment, considering only a subset of keyframes in the optimisation. Strasdat et al. (2010b) offer an excellent discussion of the relative trade-offs between these two approaches.

PTAM showed how mapping and tracking can be decoupled into a real-time camera tracking frontend and a more accurate and expansive, but also more computationally costly, map constructing backend. PTAM carries out efficient model predictive camera tracking at frame rate while building a globally consistent map from historical keyframes with sparse bundle adjustment in a lower priority background thread (Klein and Murray, 2007). PTAM proved to be a paradigmatic shift in SLAM research. Indeed, many modern SoTA sparse SLAM systems, such as ORB-SLAM (Campos et al., 2021; Mur-Artal and Tardós, 2017b; Mur-Artal et al., 2015), still follow the frontend/backend divide. Figure 1.3 shows an example of a mapping session with PTAM.

Although VSLAM systems have broad applicability across a number of domains, the potential to drive applications in real-time augmented reality is a prominent motivation. In augmented reality, a users visual experience is mediated by a device that can virtually overlay augmentations on their visual feed. Ensuring that augmentations are fused into the scene in a way that is consistent with the scene's actual geometry, and that augmentations stay anchored in place as the user moves about, requires that the device can perform SLAM. Davison et al. (2007) were among the first to demonstrate a real-time augmented reality

(a) (b) (c)

Figure 1.4: **Dense models produced by systems described in this thesis.** (a) map produced by system described in Chapter 3. Figure shows results of a 3 camera session. (b) map produced by system described in Chapter 5. (c) map produced by system described in Chapter 6.

application using a hand-held camera operating in an a priori unknown space. The system uses MonoSLAM to quickly build a feature map of the space and track the motion of the camera. Users can insert virtual objects into the feature map by using a GUI to manually attach the objects to planes in the scene. Once placed in the map, objects can be rendered on top of live tracked frames, appearing anchored to the map as the camera moves. Klein and Murray (2007) offer a similar capability. Using PTAM instead of MonoSLAM, the system can automatically extract the dominant scene plane from among the points in PTAM's more expansive map. This dominant plane is then used to host augmented reality experiences such as games and scene annotation. The augmented reality setting highlights the importance of PTAM's frontend/backend divide. During extended exploration of the scene, the map size necessarily increases while the rate at which the mapping thread can integrate new keyframes decreases until the mapping thread can no longer keep up with keyframe creation, essentially precluding further exploration. However, since the tracking thread is dependent on the number of features in view of the camera, more so than the overall number of points in the map, real-time augmented reality can continue in established parts of the map.

In the introductory passages of the current chapter we noted that the limited representational power of sparse maps led to the development of systems for incrementally constructing dense surface models of scenes in real time. Recently, the state-of-the-art in the field has come to include a multitude of systems offering large-scale, high-precision dense mapping and tracking capabilities (Dai et al., 2017; Engel et al., 2014; Henry et al.,

2012; Keller et al., 2013; Kerl et al., 2013; Mur-Artal and Tardós, 2017b; Newcombe et al., 2011a,b; Whelan et al., 2014a, 2015). As with sparse SLAM, increasing the extensiveness with which a space can be mapped, while maintaining real-time performance, is one of the central challenges in dense SLAM. This is made even more challenging by the sheer number of parameters required to represent the geometry of even the smallest spaces such as rooms and buildings. Several approaches to scalable dense SLAM have been explored, including; space and time efficient map data-structures (e.g., Niessner et al., 2013; Vespa et al., 2018); shifting volume data-structures, that only keep on hand the volume of space in the immediate vicinity of the camera (Whelan et al., 2012); geometry simplification strategies that reduce the number of parameters required to represent the scene (e.g., Salas-Moreno et al., 2014; Whelan et al., 2014b); keyframe approaches that isolate a small number of salient frames from which the map is built (e.g., Kerl et al., 2013; Newcombe et al., 2011b); and, compact map representations that don't explicitly represent free or unobserved space (e.g., surfels, Keller et al., 2013). Figure 1.4 shows examples of dense maps produced by systems described in this thesis.

Newcombe et al. (2011b) demonstrated how it was possible to more accurately reason about occlusions and interactions between virtual objects and the geometry of the space with a dense map than a sparse map. The basis of their approach is a dense SLAM system (called DTAM, short for Dense Tracking and Mapping) that recovers the full complexity of the scene's geometry, and can track camera motion against it, in real time. An example augmented reality application involves a virtual car driving on the surface defined by the dense map with simulated physics. When the car is rendered from the viewpoint of the camera, as estimated by DTAM, and composited with live images, it gives the effect of the virtual car interacting with the physical environment and disappearing behind occluding geometry. Izadi et al. (2011) present a similar approach based on KinectFusion (Newcombe et al., 2011a).

Dense Tracking and Mapping (DTAM, Newcombe et al., 2011b) is a notable early dense mapping system. DTAM builds dense surface models from monocular images in real time, i.e. it does not use a depth sensor. The dense model is composed of dense depth

keyframes. Each keyframe is estimated from images with overlapping views of the scene using multi-view stereo techniques. DTAM highlights the fact that while a depth sensor simplifies the task of estimating dense scene geometry, the use of such a sensor is not essential to achieving dense maps. With a depth sensor, the SLAM system only has to *refine* estimates of the surface geometry, rather than compute a solution to the full problem of estimating $3D$ geometry from $2D$ images. In turn this frees up computing resources. However, DTAM relies heavily on the brightness constancy constraint and short baseline frames during depth estimation and direct camera tracking, which renders it unsuitable for outdoors environments and fast camera motion.

## Autonomous Driving

The work in this thesis, particularly the work in Chapters 5 and 6 on outdoor and fisheye SLAM, has potential to be applied to problems in the automotive domain. Following on from earlier efforts in autonomous rover research, SLAM has played an important role in the development of infrastructure-free autonomous driving and advanced driver assistance systems. SLAM finds application in the automotive domain as it is an essential component of autonomous navigation and exploration in unknown environments (Makarenko et al., 2002; Stachniss, 2009). There has been a continuity of work in this area evidenced by the large number of grand challenges, such as the DARPA autonomous vehicles challenges (Buehler et al., 2009; DARPA, 2014) and the Grand Cooperative Driving Challenge (Englund et al., 2016; Lauer, 2011), and large-scale research projects, such as the V-Charge project (Furgale et al., 2013).

Despite the fact that many types of sensors, and suites of complimentary sensors, have been used during this research, cameras have demonstrated an enduring appeal. This appeal might seem counter-intuitive, at first; many scene properties, such as depth, cannot be immediately inferred from a single image (without the help of a prior, such as can be captured by a neural network trained to predict depth from images), instead the same surface point must be repeatedly measured, and associated, across multiple frames before its $3D$ position can be inferred. However, if the initial difficulties of using a monocular

camera can be overcome, cameras confer a host of benefits; they are inexpensive, small, ubiquitous, energy efficient, can be deployed in harsh environments and produce rich measurements at high frame rates.

With the introduction of the KITTI autonomous driving benchmark dataset, Geiger et al. (2012a) ask "Are We Ready For Autonomous Driving?" The question was directed at a number of problems in computer vision, including VSLAM. Prior to KITTI, direct comparison between different VSLAM systems within the context of autonomous driving was difficult without a dataset that was both large-scale and came with accurate ground truth labels. Since KITTI's inception, a number of sparse monocular visual SLAM systems have had their performance evaluated on autonomous driving conditions (e.g., Ming et al., 2023; Mur-Artal et al., 2015). A number of additional large-scale autonomous driving datasets have followed KITTI, including for fisheye cameras (Liao et al., 2021; Yogamani et al., 2019) and for dense depth estimation (Guizilini et al., 2020), facilitating further research in the application of VSLAM to autonomous driving.

Janai et al. (2020), Bresson et al. (2017) and Cheng et al. (2022) offer in-depth surveys of the many examples of VSLAM being applied to the automotive domain. Here we will describe a few recent examples. A common approach among recent autonomous driving applications is to build highly accurate maps using offline techniques, then use the pre-built map for localisation and path planning during operation (e.g., Franke et al., 2013). However, this approach is onerous as transient changes in the environment require the environment to be remapped from scratch. In a system for automated parking, (Grimmett et al., 2015) stay closer to a real-time SLAM methodology. A base map is constructed by first manually driving the car around the target car park. Then, during automated parking, the base map is used for localisation and path planning. The base map is updated with new map points during subsequent visits to the car park using multi-session visual SLAM techniques, making the system somewhat robust to structural changes in the scene (see below for description of the multi-session SLAM problem).

Tripathi and Yogamani (2021) describe a trained trajectory automated parking system based on VSLAM. The system is concerned with the case where a person repeatedly

(a) (b)

Figure 1.5: Examples of sensors used in single and multi-sensor dense mapping. (a) The Asus Xtion Pro Live and (b) the Intel Realsense *R*200, both examples of commodity depth sensors, each costing approximately US$100. The depth measurements from such sensors are very noisy. However, when multiple overlapping measurements are fused a de-noised and accurate surface reconstruction emerges.

parks in the same place, e.g., at home or at work. Two distinct phases of operation are described. In the first phase, a human driver trains the system by manually parking the car in the desired location. While the car is being manually parked, a visual SLAM system extracts the cars trajectory and a sparse feature map of the surrounding environment. In the second phase, the trained map and trajectory are used to automatically park the car. This is done by matching features in the live camera feed to features in the trained map. Once matches have been found the current pose of the car relative to the desired parking trajectory can be determined and fed to the path planner. Dahal et al. (2022) leverage the trained trajectory approach in their system for automatically aligning an electric vehicle to a charge-pad based on a multitask near-field perception pipeline (object detection, semantic segmentation and distance estimation). The system uses VSLAM to greatly extend the range at which automatic alignment can be initiated by recording the location of the charge-pad relative to a sparse feature map. Once the car is within the area included in the sparse map automatic alignment with the charge pad can begin, even if the charge-pad is not in view of the car's cameras.

## Collaborative Applications

All the dense SLAM systems mentioned so far in this section assume a single camera mapping in isolation. While it is the case that many systems that address the issue of

collaborative SLAM in the context of sparse mapping have been presented in the literature, relatively few systems that address the problem of dense collaborative mapping have been presented. Saeedi et al. write of multi-robot mapping that,

> "While single-robot SLAM is challenging enough, moving to a platform of multiple robots adds another layer of challenge. In a multiple-robot environment, robots must incorporate all available data to construct a consistent global map, meanwhile localize themselves within the global map. Multiple-robot SLAM has benefits such as performing missions faster and being robust to failure of any one of the robots; however, these benefits come at the price of having a complex system which requires coordination and cooperation of the robots." - (Saeedi et al., 2016)

These complexities are amplified in dense collaborative SLAM where the size of dense maps and the amount of data coming from each robot's sensors present unique challenges. Questions around what data should be shared between robots and whether this data should be processed in a centralised or decentralised manner, or a mixture of the two, have important implications for a collaborative SLAM system's performance which we will explore further in Chapter 2. It is also necessary to consider the manner in which a collaborative mapping system's performance can be quantified and compared to the performance of other collaborative and single-camera mapping systems. Although many datasets are available for benchmarking single camera mapping few exist that can be used to measure the performance of a collaborative system (Saeedi et al., 2016).

One of the central problems of collaborative SLAM is that, in general, the relative starting positions of each of the agents is unknown. Therefore, there is no immediately obvious single global coordinate system within which sensor data from all robots can be combined. Instead, each robot must start off in its own map with a local coordinate system. A shared coordinate system must then be inferred online as mapping proceeds. This is typically done by finding data associations (i.e. overlap) between maps which act as constraints on the relative position of each map's local coordinate systems. Once the relationship between sub-maps is known, data from each agent can be combined within a

single map. Once a common map has been established updates to the map from multiple cameras must be facilitated and coordinated to avoid corrupting the map.

Collaborative mapping can also be used to address the issues of scalability and extensiveness of SLAM. When operating in large scale environments, one way of making the large-scale SLAM problem tractable, in terms of the time taken to map a space and in terms of the computational complexity of the problem faced by the robot's SLAM system, is to distribute the mapping effort across a team of collaborating robots. Each robot is then tasked with mapping in a smaller area (Cadena et al., 2016).

Collaborative SLAM is related to the problem of multi-session SLAM. In collaborative SLAM multiple agents operate contemporaneously while in multi-session SLAM a single robot makes repeated missions through the same environment. Therefore, multi-session SLAM can be seen as collaborative SLAM unrolled over time. One of the key issues of multi-session mapping is relating the current session's map to the global map from previous sessions so that a single coherent map is reused, updated and extended to new regions over time (McDonald, 2013). This is precisely the same problem as finding inter-map constraints in collaborative mapping. The principal difference between collaborative and multi-session SLAM is that a collaborative SLAM system must facilitate concurrent updates from multiple-agents whereas a multi-session SLAM system typically operates with a single agent.

Collaborative SLAM has the potential to be applied to many problems across robotics including multi-robot exploration, collaborative robotics and augmented reality based human-robot interaction. In multi-robot exploration the goal is to distribute the exploration of a space among a number of robots efficiently and such that each of the robots spends as little time as possible re-routing to avoid collisions with other robots (Stachniss, 2009). Doing this in a general way where no assumptions are made about the robots relative starting positions or the structure of the environment requires solving the multi-robot SLAM problem (Fox et al., 2006).

An emerging application domain, where SLAM is only beginning to be applied, is augmented reality based Human-Robot Interaction (HRI). Many recent HRI approaches

rely on infrastructure, such as motion capture systems, to localise the robot and the AR headset in a shared coordinate system. However, many works note how SLAM could be used to solve this problem in an infrastructure-free way (e.g., Walker et al., 2018, 2019). Qiu et al. (2020) create a shared augmented reality workspace where both the robot and the AR headset run their own local SLAM pipelines. The robot then finds the transformation between its own map and the AR headset map by detecting the human user in its visual feed. See Suzuki et al. (2022) for a survey of augmented reality based human robot interfaces.

## 1.4 Thesis Scope

The goal of this thesis is to close the gap between the state-of-the-art in dense visual SLAM and what is required for outdoor, multiagent dense mapping. To this end we address two key challenges in SLAM; $(i)$ the development of a collaborative dense mapping system for multi-camera mapping and tracking in shared workspaces; and $(ii)$ the development of dense mapping systems that are suitable for outdoor automotive scenarios. Whelan et al. (2015)'s ElasticFusion (EF) system forms the foundation of our approach. Our work goes beyond what the initial EF system is capable of, introducing several novel contributions:

- **A Collaborative Dense SLAM System**: The system permits multiple cameras to collaboratively build a single coherent global model of a shared workspace. The system does not require knowledge of the initial relative positions of the cameras. This work was published under the title *"Collaborative Dense SLAM"* in the proceedings of the Irish Machine Vision and Image Processing (IMVIP) conference, 2018 (Gallagher and McDonald, 2018).

- **An Information Theoretic Approach to Keyframe Selection**. The approach, implemented as an extension to ElasticFusion, allows the system to avoid processing redundant information. This is achieved by leveraging the dense map to quantify the amount of novel information in incoming camera frames. This work was published at the Computer Vision and Pattern Recognition (CVPR) workshop on $3D$ Scene Understanding for Vision, Graphics, and Robotics, 2019 under the title *"Efficient*

*Surfel Fusion Using Normalised Information Distance"* (Gallagher and McDonald, 2019).

- **A Hybrid Sparse-Dense Monocular SLAM System**: We extend EF to allow operation on passive cameras operating in outdoor environments, in high speed scenarios such as autonomous vehicles. This is achieved through a combination of sparse and dense SLAM techniques and neural network based depth prediction. The resulting system was published under the title *"A Hybrid Sparse-Dense Monocular SLAM System for Autonomous Driving Year"* in the proceedings of the European Conference on Mobile Robots (ECMR), 2021 (Gallagher et al., 2021).

- **A Dense Monocular Fisheye SLAM System**: The system extends the above hybrid system, reformulating key aspects of the system's pipeline in terms of the Kannala-Brandt fisheye camera model. The system was published under the title *"A System for Dense Monocular Mapping With a Fisheye Camera"* in the proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2023 (Gallagher et al., 2023).

## 1.5 Thesis Structure

In this chapter, we have introduced the two problems of outdoor and collaborative mapping and placed them within the wider contexts of SLAM and robotics. The remainder of the thesis is structured as follows. In Chapter 2 we will give a formal definition of the SLAM problem, in both its sparse and dense form, and discuss solutions to the problem that have been presented in the robotics literature. We will also discuss the problem of collaborative mapping and review the literature on early sparse collaborative SLAM systems through to more recent dense collaborative SLAM systems. In Chapter 3 we describe our approach to dense collaborative mapping, starting with an in-depth discussion of ElasticFusion and then present our approach to multiagent mapping and tracking. In Chapter 4, we present our information theoretic method for keyframe selection. In Chapter 5 we will present a monocular dense SLAM system designed to operate on passive cameras in

automotive scenarios. In Chapter 6 we build on this initial system, reformulating its geometry estimation, mapping, and tracking modules in terms of a fisheye camera model more suited to the requirements of the automotive domain. Quantitative and qualitative experimental findings on each system's performance are presented in their respective chapters. Conclusions specific to each system are also presented in their respective chapters. Finally, in Chapter 7 we present our overall conclusions of the work to date, as detailed herein, and propose promising avenues for future research to take.

# CHAPTER 2

# Background

Visual SLAM is the problem of estimating a map of an environment being explored by a camera while simultaneously tracking the motion of the camera relative to that map. The camera can be attached to a mobile platform, such as a robot or vehicle, or attached to a human user in some way, be it hand-held or embedded in a wearable device such as an AR headset. Visual SLAM systems can be organised into different taxonomical hierarchies depending on what classification criteria are used. For example sensor type (monocular, RGBD, stereo), camera model (pinhole, fisheye, omnidirectional), map representation (points, surfels, implicit surfaces), and optimisation backend (filtering, graphical model smoothing, dense alternation). In this chapter, we categorise SLAM systems according to the particular SLAM paradigm they belong to. There are three main SLAM paradigms that are relevant to the work of this thesis; sparse visual SLAM, dense visual SLAM and collaborative SLAM. This chapter provides a technical introduction and a brief survey of the literature for each paradigm. The aim of this chapter is to act as a general background by giving an overview of key developments in visual SLAM across the three paradigms. Some additional background material that is more specific to the contributions of this thesis is deferred to the relevant chapters (Chapters 3 to 6). It is worth noting that other SLAM paradigms exist, for example multi-session SLAM (e.g., McDonald, 2013), SLAM with heterogeneous sensors (e.g., Leutenegger et al., 2013), and dynamic SLAM (e.g., Newcombe et al., 2015), to name just a few. We recommend referring to Cadena et al. (2016), and the references therein, for a comprehensive review of research on the SLAM

problem.

The chapter is organised as follows: Sections 2.1, 2.3 and 2.5 are concerned with technical aspects of sparse, dense and collaborative SLAM respectively. Each of these sections are then immediately followed by a brief survey of relevant literature (Sections 2.2, 2.4 and 2.6).

## 2.1 Sparse Visual SLAM

Consider the problem of landmark-based visual SLAM. In this version of SLAM, a camera moves through a scene taking images $(I_0, ..., I_m)$ at discrete time steps $t \in [0, ..., m)$. At each time step $t$, the SLAM system must estimate the pose of the camera $\mathbf{P}_t$, which we will take to be a matrix representing a 6-DOF rigid body motion i.e. $\mathbf{P}_t \in SE(3)$, and the scene structure, consisting of a collection of $3D$ landmarks $\mathfrak{L} = (\mathbf{l}_0, ..., \mathbf{l}_n)$. For the purposes of this discussion, we will take each landmark $\mathbf{l}_j \in \mathfrak{L}$, $j \in [0, ..., n)$ to be a $3D$ point i.e. $\mathbf{l}_j \in \mathbb{R}^3$. At time step $t$, the camera measures a number of the landmarks in the current image $I_t$. Each measurement is of the form $\bar{\mathbf{z}}_{tj}$, where subscript $t$ denotes that the measurement was made in image $I_t$, and subscript $j$ denotes the measurement is of landmark $\mathbf{l}_j \in \mathfrak{L}$. Using a generative model of the image formation process, the error between the measured location of the landmark and its predicted location, in the image domain, can be calculated

$$\mathbf{e}_{tj} = \bar{\mathbf{z}}_{tj} - C\left(\mathbf{P}_t, \mathbf{l}_j\right) \tag{2.1}$$

where $C$ is a generative function that computes the expected location of landmark $\mathbf{l}_j$ in $I_t$ given the camera pose $\mathbf{P}_t$, $\bar{\mathbf{z}}_{tj}$ is the actual measurement of the landmark in the image, $\mathbf{e}_{tj}$ is the error between the actual measurement $\bar{\mathbf{z}}_{tj}$ and the generative model prediction. Landmark measurements are corrupted by zero-mean Gaussian noise. Thus, measurements are modelled by the following equation

$$\mathbf{z}_{tj} = C\left(\mathbf{P}_t, \mathbf{l}_j\right) + \mathbf{n}_{tj} \tag{2.2}$$

where $\mathbf{n}_{tj}$ is drawn from a zero mean Gaussian distribution. The probability of making a specific measurement $\bar{\mathbf{z}}_{tj}$ of landmark $\mathbf{l}_j$ in image $I_t$ is given by

$$\rho\left(\bar{\mathbf{z}}_{tj}|\mathbf{P}_t,\mathbf{l}_j\right) \propto \exp\left(\frac{1}{2}\mathbf{e}_{tj}^T\boldsymbol{\Sigma}_{tj}^{-1}\mathbf{e}_{tj}\right) \tag{2.3}$$

where $\boldsymbol{\Sigma}_{tj}$ is the Gaussian distribution's covariance matrix. By summing the measurement errors across all images and taking the negative log of the summands we get the following cost function

$$f(\mathbf{x}) = \sum_t \sum_j -\log\left(\exp\left(\frac{1}{2}\mathbf{e}_{tj}^T\boldsymbol{\Sigma}_{tj}^{-1}\mathbf{e}_{tj}\right)\right) \tag{2.4}$$

here $\mathbf{x}$ is a state vector that encapsulates the camera poses $\mathbf{P}_t$ and landmarks $\mathfrak{L}$. Note that in Equation (2.4) we assume that only those landmarks that are visible in $I_t$ are included in the summation. By minimising 2.4 with respect to $\mathbf{x}$ we find the maximum likelihood (ML) set of camera poses and landmark positions

$$\mathbf{x}_{ML} = \underset{\mathbf{x}}{\operatorname{argmin}}\, f(\mathbf{x}) \tag{2.5}$$

to minimise $f$, iterative non-linear optimisation methods are used. Second-order Newton methods approximate the local region of $f$ about the current estimate of $\mathbf{x}$ as a quadratic curve using a Taylor series expansion up to the $3^{rd}$ term (i.e. the Hessian matrix of second order partial derivatives of $f$). During each iteration $k$ of the optimisation procedure, a displacement $\delta\mathbf{x}$ is found such that $\mathbf{x}_k = \mathbf{x}_{k-1} + \delta\mathbf{x}$ reduces the cost function $f$

$$f(\mathbf{x}+\delta\mathbf{x}) = f(\mathbf{x}) + \mathbf{g}^T\delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^T\mathbf{H}\delta\mathbf{x} \tag{2.6}$$

for gradient vector $\mathbf{g} = \frac{df}{d\mathbf{x}}(\mathbf{x})$ and Hessian matrix $\mathbf{H} = \frac{d^2f}{d\mathbf{x}^2}(\mathbf{x})$. The update $\delta\mathbf{x}$ can then be found by setting $\frac{df}{d\mathbf{x}}(\mathbf{x}+\delta\mathbf{x})$ equal to zero, yielding

$$\delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g} \tag{2.7}$$

starting from an initial estimate of $\mathbf{x}$ and iterating leads to quadratic convergence to the minimum, assuming the function is smooth, and the optimisation is well initialised. If some prior information about the camera positions or scene structure is available this can be incorporated into the cost function and the maximum a posteriori (MAP), rather than ML, estimate of $\mathbf{x}$ is computed.

The above describes a naive version of what is essentially bundle adjustment. Bundle adjustment is a general geometric parameter estimation framework with application beyond SLAM, and indeed beyond visual reconstruction. It is naive in the sense that there are a vast number of considerations to take into account when implementing bundle adjustment for a particular problem. For a full discussion of these considerations see Triggs et al. (2000) and the references therein. Here we are concerned with a number of these considerations that are pertinent from the point of view of deriving a formulation of sparse visual SLAM from generalised bundle adjustment; $(i)$ initialisation $(ii)$ correspondences and $(iii)$ scalability.

Firstly, the above procedure assumed the availability of an initialisation point $\mathbf{x}_0$ for the structure and motion parameters. In practice, such an initialisation point can be found through multi-view reconstruction techniques, for example, by finding the fundamental matrices between pairs of views or the trifocal tensor connecting triplets of views, recovering subsets of motion and structure parameters. Hartley and Zisserman (2003, Ch. 18) provide a detailed procedure for initialising bundle adjustment.

Both initialisation and bundle adjustment assume knowledge of *correspondences*. For initialisation, one must have knowledge of $2D \leftrightarrow 2D$ image correspondences. For bundle adjustment one must have knowledge of $2D \leftrightarrow 3D$ image to landmark correspondences. In order to find such correspondences, salient features, designed to be matched across changes in view point, are extracted from images and matched. In SoTA SfM systems such as COLMAP (Schönberger and Frahm, 2016), initialisation and bundle-adjustment, are performed in an iterative loop. As better estimates of the scene structure and camera parameters are found, better estimates of $2D \leftrightarrow 3D$ correspondences can be computed, which bundle adjustment needs in order to converge to an accurate solution.

The issue of scalability is particularly relevant to landmark-based visual SLAM. In theory, we could take the above procedure and for each new camera frame we could re-compute the solution to Equation (2.5). However, in practice this is not a feasible approach. At a minimum a real-time SLAM system should provide real-time camera tracking and, ideally, scene structure estimates should not be too far behind. Bundle adjustment requires either inversion or factorisation of large matrices. In our particular naive bundle adjustment described above, the Newton Method requires the calculation, and then inversion of the Hessian matrix of second order derivatives. Using the Gauss-Newton method, one can avoid explicitly calculating the Hessian, instead approximating it as

$$\mathbf{H} \approx \mathbf{J}^T \mathbf{W} \mathbf{J} \tag{2.8}$$

for weight matrix $\mathbf{W}$ and Jacobian matrix of partial derivatives $\mathbf{J}$. Note that this approximation is only valid if the residual errors $\mathbf{e}_{tj}$ are small and the generative function $C$ is close to linear. However, in either case, the Jacobian and Hessian matrices for a given sequence of input images tend to exhibit a sparsity pattern. This pattern arises from the particular co-visibility of landmarks across the image sequence in question. The key to leveraging bundle adjustment to achieve landmark-based SLAM is to be judicious about exploiting the problem structure and sparsity pattern of the underlying Jacobian and Hessian matrices.

Generalised bundle adjustment in its most abstract sense of geometric parameter estimation can be seen as inference over a graph (Dellaert, 2005). To naively solve the full bundle adjustment graph from scratch upon every new camera frame is clearly not a scalable approach to real-time SLAM. The two primary frameworks for solving the real-time VSLAM problem are filtering methods and smoothing methods. Filtering methods sparsify the problem by marginalising out all poses but the current one and any features deemed to be redundant, poorly constrained by the set of measurements or otherwise spurious. All the information in the model is summarised in a probability distribution over the remaining parameters. This probability distribution is then propagated forwarded to the next time-step (i.e. it is filtered). However, there are many limitations of filtering methods, not least that the marginalisation that seems to be the source of the efficiency ultimately

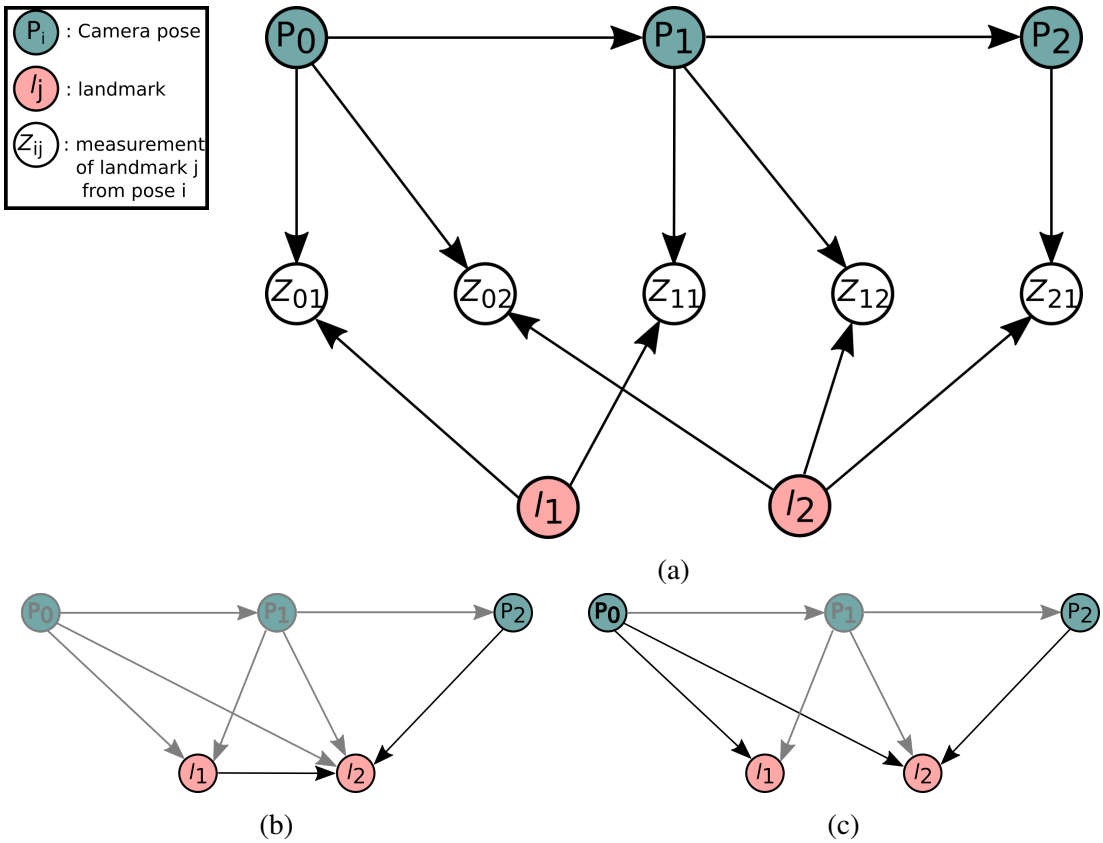



Figure 2.1: **Graphical representations of the SLAM problem.** (a) A toy SLAM problem represented as a Bayesian network. Landmarks in the map are conditionally independent given camera poses. The unknown poses and landmarks are given by the coloured circular nodes. The known measurements are given by black circles. (b) Graph structure of filtering approach to the same SLAM problem without explicitly representing the measurements. Only the current pose is being estimated and landmarks become connected. (c) Graph structure of keyframe bundle adjustment. Only a subset of frames is being used, and no marginalisation means no dense fill in between the nodes.

leads to dense fill-in of the state-space covariance matrix as the number of images increases which means that sparse factorisation techniques are no longer applicable. Many works attempting to deal with this problem have been presented (Eustice et al., 2006; Thrun et al., 2004; Walter et al., 2007). However, further work has shown that Kalman filter-based SLAM is inherently inconsistent, meaning that over the course of a long enough mapping session the map and robot pose estimates will diverge from the true robot pose and feature locations (Julier and Uhlmann, 2001).

Smoothing approaches are concerned with the *full* SLAM problem where the object is to recover the full robot trajectory as well as a map of landmarks (though some works on pose graph relaxation focus on recovering the trajectory only e.g, Newman et al., 2009;

Thrun and Montemerlo, 2006). Full SLAM smoothing methods do not marginalise out prior poses or map points and therefore the underlying covariance structure of the problem does not become densely connected. Internally, updates to the solution $\delta\mathbf{x}$ are ultimately computed via solving the normal equations

$$\mathbf{A}^T\mathbf{A}\delta\mathbf{x} = \mathbf{A}^T\mathbf{b} \tag{2.9}$$

for system Jacobian matrix $\mathbf{A}$ and residual vector $\mathbf{b}$. Typically, this system is solved via sparse *Cholesky* factorisation. The specific column ordering of the Jacobian matrix $\mathbf{A}$ can lead to improvements in performance of the sparse Cholesky decomposition (Dellaert, 2005). Incremental SAM methods improve efficiency even further by updating a running *QR* factorisation of $\mathbf{A}$, rather than updating $A$ then re-factorising it (Kaess et al., 2008). A further set of smoothing approaches sparsify the problem by considering only a subset of keyframes. Keyframes can be selected based on their being salient or based on their being within a fixed temporal window around the current frame (i.e. fixed lag smoothing, see for example Kaess and Dellaert (2006), and windowed bundle adjustment, see for example Strasdat et al. (2011)). Once a subset of frames has been identified, what is essentially a sparse bundle-adjustment is solved to update the motion and structure parameters of the current active keyframe selection (e.g., Campos et al., 2021; Klein and Murray, 2007; Mur-Artal and Tardós, 2017b; Mur-Artal et al., 2015).

Current experimental results indicate that smoothing methods offer the best accuracy per unit of compute time when compared to filtering methods (Strasdat et al., 2010b). The graphical structure of both filtering and smoothing approaches to SLAM is visualised in Figure 2.1.

## 2.2 Review of Sparse Visual SLAM Methods

This section reviews relevant literature on sparse visual SLAM. There are many important works in the literature that predate what are considered to be the first real-time visual SLAM systems. Here we start with a brief overview of this earlier body of work, including

works on visual odometry and passive navigation. From there we review more recent developments in sparse visual SLAM. This section does not serve as a comprehensive review of the literature, rather we provide a brief summary of important developments that helped support later advances in real-time dense reconstruction systems.

During the 1980s, Moravec (1980) developed the first robots that could navigate using stereo visual odometry (VO). In this setup, the robot is tasked with autonomously navigating through a cluttered environment from its start location to a user designated end location. The robot has no prior knowledge of the obstacles in the environment, it must discover them on its own, based on live sensor data, and plan its route accordingly. In order to do this the robot would have to be able to monitor its own progress towards the end point as well as infer the position and general shape of obstacles relative to it.

Periodically, the robot would stop to take a stereo capture with its onboard camera. By isolating salient feature points in the images and triangulating their $3D$ position a local map of the robots surroundings is estimated. At the next capture site a new local map is constructed in the same way. Corresponding points in the old and new local maps are found by matching their respective image space features. The ego-motion of the robot is inferred by bringing the old and new local maps into alignment in an iterative non-linear least-squares procedure based on Newtons method. Residuals in the underlying cost function are weighted in proportion to the inverse of the combined scalar uncertainty of the point as measured in both the new and the old local maps.

Matthies and Shafer (1987) improved on this technique by first noting that error in stereo triangulation of $3D$ points was better modelled by a normal distribution than a scalar value. A Kalman filter is then used to update the $3D$ landmark positions with measurements from the new local model. The authors propose using the method of spatial uncertainty propagation (Smith and Cheeseman, 1986) for extracting an estimate of the global $3D$ robot position and bearing in the $xy$ ground plane from the relative pose offsets computed by VO (Matthies and Shafer, 1987). Speed-ups and improvements brought stereo range estimation into the realm of near real-time operation (Matthies, 1992; Okutomi and Kanade, 1993) and increased robustness of stereo VO algorithms (Olson et al., 2003).

Bruss and Horn (1983) devised a method for passive navigation, i.e. tracking the motion of a hand-held camera, by extracting parameters describing the motion of the camera from the optical flow between successive images. This work proved to be prescient in two ways: (i) the framing of the problem as one of passive navigation of a hand-held camera i.e. there is no robot control inputs and motion can be loopy and irregular; and (ii) the proposed tracking function would be considered *dense* in the sense that it operates over the whole image optical flow field. Dense methods later re-emerged as a paradigm as dense tracking costs were developed for dense visual SLAM systems.

Issues of real-time performance experienced by early methods aside, VO as a paradigm has two important caveats worth mentioning. Firstly, errors during feature matching, triangulation of $3D$ points, and from other sources, inevitably bleed through and become errors in estimates of the robot's position and orientation, no matter how robust the underlying algorithm. Accumulating error frame-by-frame leads to the robot pose estimate drifting over time. The pose estimate deviates without bound from the true pose. The degree of accumulated drift often becomes obvious when the robot closes a loop with its own trajectory; even though in reality the robot has come back to a place it has already been its estimated pose is way off. It has been shown that orientation error accounts for a large proportion of the total error in a robot's pose. In certain domains, the particular axis (or axes) of rotation (e.g., roll, pitch and yaw) contributing most significantly to error can be further identified. An additional signal is needed to periodically correct the robot position estimate. Therefore, readings from a sun sensor, compass, gyroscope or other such sensor can be used to reduce the error rate of a VO algorithm (Olson et al., 2003). Another option is to keep a map summarising historical data. The system can then periodically compare live data against the map data to see if a loop has been closed in the robot's path. If it has, the displacement of the robot's current estimated pose from its pose relative to the historical map data can be used to reduce accumulated drift. The problem of identifying loops and reducing drift is referred to as the *loop closure* problem in SLAM literature.

The second caveat worth mentioning is that VO methods lack such a persistent map that summarises historical sensor measurements. VO methods only build maps of $3D$

points to the extent that those points help to track the robot. Once points fall out of view of the camera they are no longer considered useful and are discarded. In many scenarios this is not a problem, the robot may never revisit the same location twice, so a persistent map is of little use. However, for applications where a robot operates in a confined space or where a robot periodically revisits locations (i.e. closes loops in its trajectory) keeping a persistent map makes sense. While VO has its place within an autonomous system, the full problem of SLAM is of particular importance to robotic platforms that are capable of long-term autonomy.

Harris and Pike (1988) introduced an early visual tracking and mapping system that built a $3D$ point cloud from images. Each point in the map was represented as an independent $3D$ probability distribution. $3D$ map point covariances are used to constrain the search for feature matches. Once a set of feature matches has been established, an iterative model predictive camera tracking procedure tracks the motion of the camera, again using the $3D$ map point and feature location uncertainties. Features in each frame were used to update the map points using Kalman filtering. Critically, theoretical results show that map points are independent only if the camera pose is known, thus the per-point probability distributions are only justified if the camera tracking is correct (Montemerlo et al., 2002; Murphy, 1999).

Davison et al. (2007); Davison (2003) introduced a seminal visual SLAM system that played a pivotal role in bridging the gap between real-time structure-from-motion research of the computer vision community and the already well-developed literature on real-time SLAM of the robotics community, by showing how the Extended Kalman Filter (EKF) could be applied to the case of pure vision SLAM with a hand-held camera. By using features capable of long-term stability and propagating the full joint uncertainty over camera and map state, the system was capable of achieving accurate and repeatable camera localisation. Even after periods of neglect, the system could revisit old regions of the map and reuse landmarks for localisation, reducing the effects of drift in long sessions where the camera moves to poorly mapped regions and then back to well mapped regions.

Aside from the $\mathcal{O}(n^2)$ complexity of the EKF, and possible issues around incon-

sistency, the system had two main limitations during system initialisation and feature initialisation. To initialise the system a fiducial marker of known size is placed in front of the camera. This served two purposes: $(i)$ to fix the scale of the scene, which is otherwise not observable from projections of a general scene into a monocular camera, and $(ii)$ to allow the system to enter its main SLAM loop immediately by inserting known features of essentially zero uncertainty directly into the map. The reason for purpose $(ii)$ above becomes more obvious when we consider the systems need for a feature initialisation procedure. Since the depth of new features cannot be known from a single image, they must go through an initialisation procedure before they can be inserted into the map. The procedure uses a $1D$ particle filter to refine the feature's depth over several frames. Once the particle's density assumes a Gaussian profile, the feature is added to the map as a normal $3D$ point. Starting with a map with no features in it would greatly hinder tracking and mapping accuracy until enough features could be resolved to reduce map and camera uncertainty. As the system relied heavily on top-down processing to limit the amount of computation required at each step, one would expect the system to perform more slowly in such a start-up.

From a Bayesian point of view, the need for a separate feature initialisation protocol is a direct consequence of the parameterisation of map points as $3D$ $XYZ$ Euclidean points. During feature initialisation the direction of point is known but not its position. Therefore, a semi-infinite line extending from the camera to infinity with large uncertainty must be represented during initilisation. Such uncertainties are not well captured by Gaussian distributions and do not perform well during the linearisation step of the EKF formulation. (Montiel et al., 2006) showed that, using an inverse depth representation features can be immediately inserted into the map and used for SLAM.

Klein and Murray (2007) introduced a parallel tracking and mapping (PTAM) SLAM system. As was noted in the introduction to this thesis, PTAM represented a paradigmatic shift in SLAM research. MonoSLAM focused on probabilistic filtering of the joint camera pose and map state, delivering updated camera pose and map estimates every frame. One of the key insights of PTAM was to notice that while camera tracking needs

to operate in real time, mapping, in some applications (like augmented reality), does not. The two problems of localisation and mapping could be decoupled; given a map, find the pose of the camera; given the pose of the camera, update the map. By decoupling the two problems their computations could be separated into a camera tracking frontend running on a high priority real-time thread, and a map building back-end running on separate low priority background thread. With separation, and the consequent increase in available resources and compute time budget, a tracking procedure, based on non-linear optimisation of a robust cost function, could be used to track the camera during rapid movements. Mapping also benefited, no longer coupled to the hard real-time constraints of camera tracking, expensive batch bundle adjustments could be performed producing larger, more accurate maps.

Instead of operating over the full stream of tracked images, the frontend identifies and passes a sparser set of keyframes to the backend. This helped to curb the growth in complexity of the bundle adjustment, especially during sequences where the camera stays in well mapped regions of the scene. During such sequences the system runs an expensive full bundle adjustment over all keyframes and map points. However, during exploratory sequences where new keyframes are identified at a much higher frequency, the system runs a bounded local bundle adjustment. Local BA optimises only the most recent keyframe, its 5 nearest neighbours in the map, all the map points observed in those keyframes and all the keyframes that contain a measurement of those points. To bootstrap the map, two keyframes with an approximately $10cm$ baseline are captured in a user guided initialisation procedure. The essential matrix, encoding the geometric relationship between corresponding points in the two images, is estimated using a technique called random sample and consensus (RANSAC Fischler and Bolles, 1981). An initial map is then triangulated and further refined via bundle adjustment.

Klein and Murray (2008) introduced several extensions to the original PTAM algorithm aimed at improving tracking during rapid camera motions and tracking recovery from failure. Edgelet landmarks, i.e. $1D$ line segments, which are more resilient to motion blur than point features, are added to the mapping and tracking optimisations. A

procedure for estimating the rotational component of inter-frame camera displacement is proposed. The procedure minimises the sum-of-squared differences (SSD) directly over pixel intensity values. This procedure replaces the motion model used by the previous version of the system to initialise camera tracking.

Despite the additional measures put in place, tracking failures were still inevitable. When tracking fails, incoming frames are downsampled and compared to a downsampled version of each keyframe (again using the SSD). The pose of the live image in the coordinate frame of the matched keyframe is then found using the rotation prealignment procedure from which point full SLAM mode can be restarted.

One of the main challenges faced by monocular SLAM systems is the unobservability of the scale of the reconstruction. Scale can only be introduced by some additional source of information, by either observing an object of known scale or distance, or by using non-holonomic constraints (Scaramuzza et al., 2009). Scale is one of 7 degrees of gauge freedom in monocular SLAM (the other 6 correspond to a $6DOF$ rigid body transformation of the whole map). In many applications in virtual reality and robotics the map needs to be of the same scale as the environment in order to be useful. This led to the need for additional map initialisation procedures in monocular SLAM pipelines such as those used in MonoSLAM and PTAM. In large-scale mapping, even with such initalisation procedures in place, drift in the unconstrained scale gauge is inevitable. If left unacknowledged this drift in scale can lead to difficulties when a loop is closed.

Strasdat et al. (2010a) introduced a keyframe-based monocular SLAM system suitable for large-scale mapping. The chief insight is that loop closure correction can be split into a pose graph optimisation and a structure-only bundle adjustment. Once a loop has been identified, all keyframe camera poses are treated as members of the Lie group of $7DOF$ similarity transforms $Sim(3)$, which incorporates a scaling parameter. The loop closure constraint is the similarity transform between the live frame and the matched keyframe $\Delta\mathbf{S}_l^k \in Sim(3)$. Each absolute camera pose and the relative poses between pairs of cameras, including $\Delta\mathbf{S}_l^k$, are optimised with pose graph optimisation. Once optimisation is finished, points are corrected with the updated keyframe poses. In a final step, the map

is optimised with structure-only BA. The effect of this is that, upon a loop closure, the drift in scale that has occurred can be measured and corrected, allowing the robot to revisit and reuse old portions of the map.

Keyframing works well to reduce growth in BA complexity in local, small scale sequences, such as a camera hovering over a desk. However, keyframing does not help substantially during large scale, exploratory sequences such as a robot mapping a city. Many approaches to dealing with this scalability issue were proposed including pose graph optimisation and sub-mapping. However, each of these approaches comes with its own problems. In any case, during loop closure both of these approaches have to perform a global update of all parameters to stay close to the optimal solution. Sibley et al. (2009) introduced the concept of relative bundle adjustment (RBA) which was later used as the basis for large scale SLAM (Mei et al., 2010; Sibley et al., 2010). Instead of defining a single privileged Euclidean coordinate system like classical BA, RBA represents the scene using a Riemannian manifold implemented as a graph. Nodes in the graph represent frames and their measurements of landmarks in the map. Edges in the graph represent relative transformations between frames. Landmarks are stored relative to a single base frame but can be measured from many frames. During optimisation the full kinematic chain of relative poses connecting a landmark's source frame to subsequent frames it is measured in must be composed in order to compute measurement Jacobians. Due to the relative formulation, an active region of the graph containing parameters that might need to be updated can easily be computed. Breadth-first search from the current frame finds frames whose average reprojection error changes above a threshold. Such reprojection error quickly dissipates as the search moves away from the source frame, revealing the active region of the map. This effectively limits the size of the bundle adjustment problem that needs to be solved, especially during loop closures.

One issue with the RBA approach as discussed above is that, although its map is a Riemannian manifold (i.e. locally it should be Euclidean but not necessarily globally), the active region is not explicitly constrained to be Euclidean which could have negative effects when performing SLAM in localised small scale environments. Strasdat et al.

(2011) attempt to overcome this with a double window optimisation scheme. Constraints between $3D$ points and camera poses are harvested from an inner window. Relative pose-to-pose constraints are harvested from an outer window. Both sets of constraints are jointly optimised in a single cost function which is essentially a combination of a bundle adjustment and pose graph optimisation. The map consists of keyframes organised in a *covisibility* graph. Nodes represent keyframes and consist of a pose and the map points which are visible in the frame, as well as their projections in the image. Edges between nodes are weighted, representing covisibility between the two keyframes, that is, the number of map points they both observe. The inner window consists of the subset of keyframes that are most visible, while the outer loop consists of a wider subset of less covisible keyframes. Incoming frames are tracked relative to the local map around the reference (latest) keyframe. The system searches for both local metric loop closures and global loop closures. Local loop closures bring the current keyframe back into alignment with what is the true local map by searching for map points that are visible in the keyframe but are not considered part of the current local map. Global loop closures are found via an appearance-based mechanism and can find loop closures between more distant keyframes. In either case, a loop closure results in a new edge constraint on the relative transformation between the current reference keyframe and an existing keyframe.

The SLAM systems presented up to this point are feature-based. $2D \rightarrow 2D$ image correspondences and $2D \rightarrow 3D$ data-associations, found through feature matching, are the basis for the optimisations being performed by these systems. Features are useful for several reasons. For one, they can be matched in an image without any prior assumptions about the cameras pose (although priors can be used to limit the extent of the search). This allows for mapping and tracking even when the camera is undergoing large inter-frame displacements and some degree of illumination variation is present in the scene. It is also the basis for many approaches to camera relocalisation and loop closure detection. However, there are also several drawbacks to using a feature-based approach. Extracting and matching features every frame is computationally expensive and systems that use features tend to underutilise the gradient information in the image. Features are designed

to detect points that can serve as landmarks (i.e are likely to be detected in other images from other view points), not for precise localisation of those landmarks within an image (Triggs et al., 2000). Feature matching and localisation implicitly depend on the image intensities of the underlying pixels. To match a known feature to a new image one must compare its descriptor (a function of image intensities) to patches of image intensity of the new image. Additionally, in order to precisely localise a feature in an image, non-linear sub-pixel refinement must be performed (Lowe, 2004).

Engel et al. (2014) introduced a feature-less visual SLAM system. Similar to PTAM and the SLAM system of Strasdat et al. (2010a), the map consists of a collection of Keyframes. Each keyframe consists of a colour image, a camera pose in $Sim(3)$ and a dense depth map. A depth map is a frame of the same dimension as the image frame where each pixel site stores the depth from the image plane to the corresponding source surface in the scene. Keyframes are collected together in a pose graph. Keyframes are nodes in the graph while relative similarity transforms connecting two keyframes are edges. Using similarity transforms in this way ensures that scale drift between keyframes is explicitly modelled and can be reduced during loop closure. Instead of extracting and tracking features, the system performs direct alignment of the current frame to most recent keyframe directly over the image intensities. The keyframe depth map provides the information necessary to perform projective data-association between the two frames (also called image warping). The pose of the previous frame is used as an initalisation point for tracking the current frame. Once the frame is tracked, it is used to refine and fill in the keyframe depth map using the depth map filtering method of Engel et al. (2013). If the camera moves too far away from existing keyframes a new keyframe is inserted, with its depth map initialised from the previous keyframe.

Whenever a new keyframe is added to the map, constraints between it and the existing keyframes must be added to the pose graph. The new keyframe is aligned to each of its neighbouring keyframes via direct image alignment w.r.t the space of $Sim(3)$ transformations. Matching keyframes from further away in the map are searched for using an appearance-based (i.e. image-based) loop closure detection mechanism. Such matches

represent large scale loop closure constraints. The pose graph is continuously optimised in the background.

Mur-Artal et al. (2015) presented ORB-SLAM, a feature-based monocular visual SLAM system. Prior to ORB-SLAM's introduction, many insights and advances had been made on the various sub-problems in visual SLAM; tracking, mapping, relocalisation and loop closure. However, a SLAM system that incorporated the best solutions and most up-to-date understanding of the SLAM problem had yet to be presented. The value of ORB-SLAM was to do exactly this. ORB-SLAM has its roots in PTAM and in the keyframe BA SLAM paradigm. Instead of PTAM's two threads, operating a tracking frontend and a mapping backend, ORB-SLAM runs three threads; a real-time camera tracking frontend thread, a local mapping backend thread, and a global mapping full BA thread. ORB-SLAM's map consists of a collection of $3D$ points and set of keyframes, collected together in a covisibility graph similar to Strasdat et al. (2011). Each $3D$ point is described with an ORB descriptor (Rublee et al., 2011). ORB features are used for all tasks in each of the three threads.

Camera tracking is initalised with a constant velocity motion model and motion only BA w.r.t the previous frame. If the camera is lost, a global relocalisation procedure finds a matching keyframe using image-based matching technique. Data-associations between ORB features in the map and in the lost frame are then used to resolve the pose of the camera in the coordinate frame of the matched keyframe by solving the perspective-$n$-point problem (PnP Lepetit et al., 2009) wrapped in a RANSAC loop. Finally, the initial estimate of the camera pose is refined with motion only BA w.r.t a local map. The local map is constructed from keyframes that share map points with the live frame and keyframes that share map points with those keyframes. Covisibility information is retrieved from the covisibility graph defined similarly to Strasdat et al. (2011).

If the current frame is well tracked and contains sufficient amount of new information relative to its most similar keyframe in the map, it is inserted as a new keyframe. Keyframes are added liberally and then pruned if they contain redundant information. This enables good camera tracking performance during periods of exploration but miti-

gates against unbound growth in complexity by removing large numbers of overlapping keyframes. In contrast, PTAM's more conservative approach to keyframing insertion, based on distance to nearby keyframes, negatively affects tracking performance as the camera moves away from the map.

Local BA optimises the new keyframe, all the keyframes connected to it in the covisibility graph, and all the map points observed by these keyframes. All keyframes that see those points but are not directly connected to the new keyframe are also included in the optimisation but are kept fixed. Once a keyframe has been included in the map via local mapping, it is used to search for loop closure candidates. Similar to relocalisation, an image-based matcher finds candidate keyframes. For each candidate, the 7DOF similarty transform from the current keyframe to the candidate is found. To do this, a set of $3D \rightarrow 3D$ correspondence between map points in view of each frame is found, using the map points ORB descriptors. A similarity transform aligning the two point clouds is then found using Horn's method (Horn, 1987) in a RANSAC loop. To correct the loop, a scale drift aware pose graph optimisation is performed, similar to Strasdat et al. (2010a). The similarity transform found above acts as the loop closure constraint during optimisation. Instead of operating over the full covisibility graph during loop closures, ORB-SLAM uses the *essential* graph. The essential graph is composed of the spanning tree of the covisibility graph, strongly connected edges and loop closure edges.

An interesting result from this work shows that the feature-based localisation of ORB-SLAM outperforms the direct alignment method used by LSD-SLAM. This discrepancy is partially explained by the viewpoint and illumination invariance of the ORB features being used. Also, the susceptibility of direct methods to visual artefacts caused by rolling shutter and automatic camera functions such as auto-exposure can cause issues during camera tracking. Torr and Zisserman (2000) present a deeper analysis comparing direct and feature-based methods. Their work shows that incorrect correspondences have a much worse effect on direct methods than they do on feature-based methods. This is due in large part to the fact that the reprojected position of features in the image domain is going to be invariant to photometric and geometric distortions between images. While such

distortions can lead to incorrect matches, with robust optimisation techniques and iterative recalculation of matches, they have little effect on optimisation. Cost functions used by direct methods need to be invariant to these distortions, or they will fail. Although some methods attempt to remove these distortions by more completely modelling the image formation pipeline (e.g Engel et al., 2017), if the model does not account for all sources of distortion, it will lead to systematic degradation of tracking performance and will need to be further extended (e.g Schubert et al., 2018). Torr and Zisserman (2000) also highlight that for BA to converge to the ML solution, the error residuals must be independently distributed. While this is true of feature-based methods, it is not necessarily true of direct methods, potentially leading to biased estimates. However, semi-direct methods have shown that both approaches, feature-based and direct, can be combined in a single tracking procedure to great effect (Forster et al., 2014).

Mur-Artal and Tardós (2017b) extended ORB-SLAM to include RGB-D and stereo-based SLAM and a localisation only mode. Sparse systems such as those presented in Mur-Artal and Tardós (2017b) and Campos et al. (2021) continue to provide state-of-the-art SLAM capabilities. The feature-based approach taken by ORBSLAM-3 makes accurate camera tracking possible even during adverse lighting conditions and fast camera motion. In Chapters 5 and 6 we leverage ORBSLAM-3's robust camera tracking capabilities in the development of dense visual SLAM systems suitable for outdoor applications, such as in the automotive domain, where lighting is not fixed and the camera may move at speed. One of the main limitations of ORBSLAM-3, and the other sparse visual SLAM systems discussed in this section, is that they produce sparse maps which limit their usefulness in some applications. In Sections 2.3 and 2.4 we shift our focus to dense VSLAM and systems that can produce high fidelity $3D$ maps.

## 2.3 Dense Visual SLAM

In comparison to sparse SLAM systems, dense SLAM systems aim to estimate a high fidelity $3D$ reconstruction of the environment being explored. Typically, this involves using as many of the pixels in each frame as possible to fit a model of the $2D$ surface that

underlies the scene being viewed. As we shall see in Section 2.4, some dense systems estimate a discrete sampling of geometric primitives that lie on this surface (e.g Keller et al., 2013; Kerl et al., 2013). While being densely populated, these representations are ultimately discrete and, therefore, do not capture higher-level properties of the scene's surface geometry than what is, in principle, possible with a sparse system. In Section 2.4 we shall also discuss other types of dense systems that do go beyond representing the scene as geometric primitives and attempt to model the topological structure of the surface itself (e.g Newcombe et al., 2011a; Schöps et al., 2020). This section starts by introducing the mathematical notion of a surface and then proceeds to discuss how early work on offline surface reconstruction formed the foundation for later developments in dense visual SLAM.

A surface $M$ embedded in $3D$ space can be seen as a $2D$ manifold (see O'Neill, 2006, for a detailed explanation). The goal of dense visual reconstruction is to recover a surface $S$ that that approximates the true surface $M$ from a set of noisy measurements $X = (\mathbf{x}_0, ...\mathbf{x}_n)$ of $M$. Each measurement point $\mathbf{x}_i \in X$ is a perturbation of a point $\mathbf{y}_i$ lying on the true surface $M$, and takes the form

$$\mathbf{x}_i = \mathbf{y}_i + \mathbf{e}_i \tag{2.10}$$

where each $\mathbf{e}_i \in \mathbb{R}^3$ is a zero mean normally distributed noise vector. The measurements $X$ could, for example, be from a laser range finder or a sparse SfM system such as COLMAP (Schönberger and Frahm, 2016). Given an appropriate estimate of the upper-bound on measurement error $\delta$ and the sampling density $\rho$, with which $X$ was captured, it is possible to make reasonable estimates of which points $\mathbf{p} \in \mathbb{R}^3$ lie on $M$ (Hoppe et al., 1992). A point $\mathbf{p} \in \mathbb{R}^3$ with $d(\mathbf{p}, X) > \delta + \rho$, where $d$ is distance between $\mathbf{p}$ and the closest point to it in $X$, clearly does not lie on $M$ (Hoppe et al., 1992).

To compute $S$, the data points $X$ can be used to construct a *truncated signed distance function* (TSDF). The TSDF maps each point $\mathbf{p} \in D$, where $D \subset \mathbb{R}^3$ is some bounded space around $X$, to its signed distance to the nearest point on $M$. Distances outside the surface are given a negative sign while distances inside the surface are given a

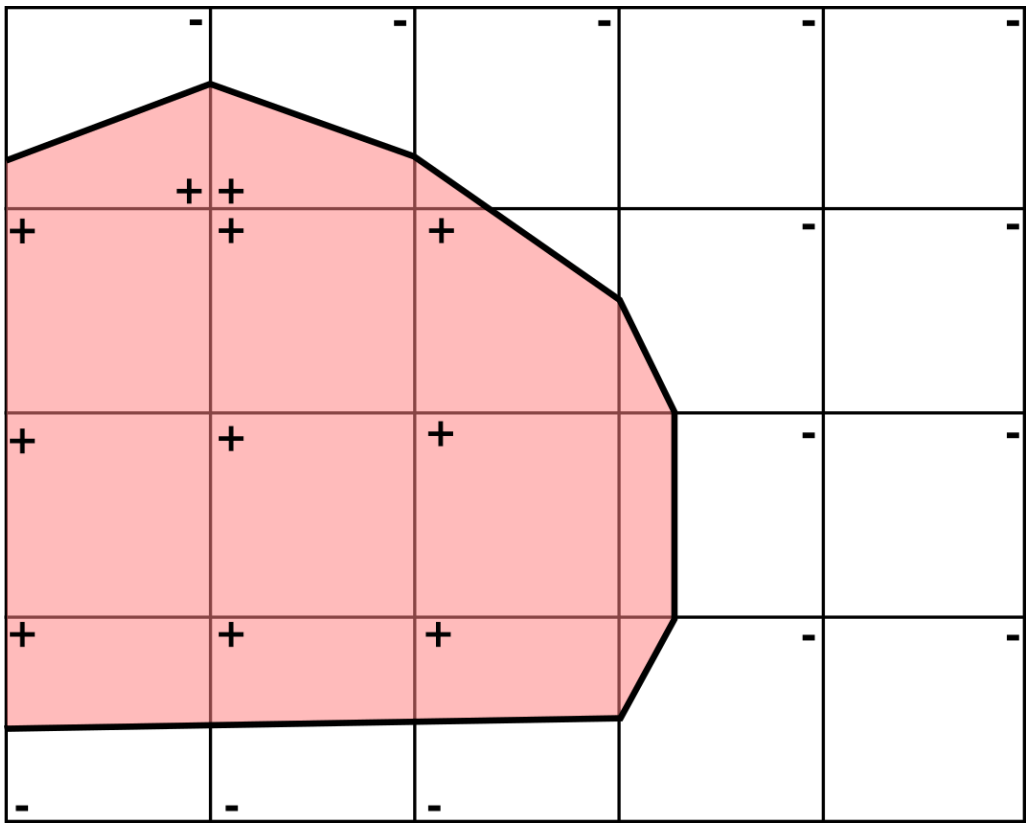


Figure 2.2: **An example of a** $2D$ **TSDF approximated as discrete grid.** In $3D$ each grid cube is referred to as a *voxel* (volume element) and the grid is referred to as a voxel grid. The number of voxels and their side length determine the resolution and size of the reconstruction the voxel grid can support.

positive sign. The surface $S$ estimating $M$ is then implicitly defined as the zero level set of the TSDF. In reality $M$ is unknown. However, it can be approximated in piece-wise linear fashion. To do so, each point, and its neighbours, are used to estimate the tangent plane of best fit (in the least squares sense) to the surface $M$. The TSDF can then be found by computing the distance of each point in $D$ to the nearest tangent plane. The zero level set of the TSDF can then be extracted with a contour tracing algorithm for example marching cubes (Lorensen and Cline, 1987). For more details see Hoppe et al. (1992), and Hoppe (1994) for an extended treatment. A visualisation of a TSDF can be seen in Figure 2.2.

The above discussion does not take into account any specific features of the visual SLAM problem. Rather it considers the general problem of inferring a surface model from unorganised point clouds. In visual SLAM, the data comes from a camera, i.e. measurements are arranged on the image plane in a regular grid pattern and related to the $3D$ world by a well-defined camera projection function. This regularity can be taken

advantage of to simplify the reconstruction process. Further to this, in SLAM the scene is imaged in an incremental fashion, thus a procedure that can incrementally refine a surface reconstruction based on new information is required.

Given a set of depth maps $\mathcal{D}_t : \phi \rightarrow \mathbb{R}$, for discrete depth map domain $\phi \subset \mathbb{N}^2$, with known global poses $\mathbf{P}_t \in SE(3)$, a maximum likelihood surface $S$ can be estimated (Curless, 1998; Curless and Levoy, 1996). Per-frame surfaces can be estimated by fitting a piece-wise linear mesh to the depth values in the depth map. Per-frame weighted TSDF volumes $s_t$ and weight functions $w_t$ can be extracted, updating the TSDF along the lines of sight of the sensor. The overlapping frame surfaces can then be fused within a single global cumulative weighted TSDF in incremental fashion

$$\begin{aligned} D(\mathbf{x})_{t+1} &= \frac{D_t(\mathbf{x})W_t(\mathbf{x}) + d_{t+1}(\mathbf{x})w_{t+1}(\mathbf{x})}{W_t(\mathbf{x}) + w_{t+1}(\mathbf{x})} \\ W(\mathbf{x})_{t+1} &= W_t(\mathbf{x}) + w_{t+1}(\mathbf{x}) \end{aligned} \tag{2.11}$$

for cumulative TSDF $D$, cumulative weight function $W$ and $3D$ point $\mathbf{x}$. The weight functions $w_t$ express variations in uncertainty between different depth maps due to noise in the formation of the depth maps. By fusing multiple overlapping frames, the effects of this noise, and the effects of slight misalignment of the frames, on the final surface reconstruction can be reduced. As before, $S$ can be retrieved as the zero level set of $D$ using marching cubes (Lorensen and Cline, 1987), or a similar contour tracing algorithm. TSDF updates along a given ray direction are only considered within a restricted zone around the surface hence *truncated*. This prevents opposite sides of a surface along a ray influencing each other. It also helps to maintain a level of computational efficiency during processing as surface updates need only occur within the vicinity of the surface. The truncation region is set to accommodate the expected degree of sensor noise. See Curless and Levoy (1996) and Curless (1998) for more details.

A standard voxel grid implementation of a TSDF exhibits a large degree of sparsity. Most of the voxels are used to represent either free or unobserved space. The sparsity of the TSDF can be exploited to represent the surface more compactly by only explicitly storing voxels with surface measurements (Niessner et al., 2013).

The above procedure assumed known sensor poses which, in the context of real-time dense visual SLAM, cannot be known a priori and must be estimated in real time. An early paradigm for full dense visual SLAM utilised existing sparse SLAM methods such as PTAM to perform the essential SLAM operations of tracking, sparse mapping, loop closure and relocalisation. Dense reconstruction schemes were layered on top of this sparse backbone, utilising the camera poses and points, but the dense geometry itself did not feed back into the SLAM system (Newcombe and Davison, 2010; Pollefeys et al., 2008; Stühmer et al., 2010).

Later, visual SLAM systems based on dense alternation removed the dependency on sparse SLAM systems. Unlike sparse joint optimisation, dense alternation splits the SLAM problem into two separate steps. In the first step, the map is assumed to be correct and is used to track the motion of the camera. This is typically done with direct, whole image alignment between the current live frame and rendering of the model into a virtual frame. Once aligned, the camera pose is assumed to be correct and used to extend the map by integrating measurements from the current live frame. DTAM is representative of this class of algorithms (Newcombe et al., 2011b). Such a decoupling is clearly a departure from the more probabilistically principled joint estimation frameworks of sparse SLAM, built on the knowledge that scene points are only probabilistically independent if the camera pose is known perfectly which is rarely the case in reality. However, thus far it has proved necessary in order to cope with the amount of data involved. Indeed, it appears that it is the shear amount of data being used in a dense SLAM system that enables it to overcome the lack of joint estimation (Platinsky et al., 2017).

Though the above discussion talked mainly of the use of TSDFs as the representation of the dense surface models, the question of representation in dense SLAM is very much an open research problem. A suitable representation of dense surfaces must meet many requirements including efficiency (it must support real-time operation), scalability (it must extend to large scenes and high resolutions), usability (it must be straightforward for downstream software to extract relevant information) and the ability to represent complex surfaces with arbitrary topologies. Many representations have been proposed including;

dense depth keyframes (Newcombe et al., 2011b), surfel lists (Keller et al., 2013), implicit functions (Newcombe et al., 2011a), occupancy grids (Loop et al., 2016), planes (Salas-Moreno et al., 2014) and neural radiance fields (Sucar et al., 2021). However, there is no 'one size fits all' representation that is suitable for all application domains as each representation imposes trade-offs in performance.

The question of how per-frame depth maps are computed in the first place was also left largely unaddressed. Many multi-view stereo techniques to infer dense depth maps have been proposed e.g., Gallup et al. (2007) propose an efficient plane-sweeping algorithm which can be used for this task. However, active depth sensors like the Microsoft Kinect drastically simplified the depth acquisition process. An emergent approach uses deep neural networks that leverage global and local cues in RGB images to estimate per-pixel depth values (Eigen et al., 2014; Zhou et al., 2017). In this thesis, we demonstrate dense mapping with both active depth sensors and neural network-based depth prediction.

## 2.4 Review of Dense Visual SLAM Methods

In this section we review key contributions in the literature surrounding dense visual SLAM. For a comprehensive review of the broader SLAM literature see the survey of SLAM presented Cadena et al. (2016). Here we are interested in systems that perform live dense surface reconstructions.

One of the earliest dense visual SLAM systems to offer what would be considered a full SLAM pipeline was that of Henry et al. (2012). In their system, the authors combine visual feature-based RANSAC alignment with dense iterative closest point (ICP) for joint photometric and geometric frame-to-frame tracking of a RGBD camera. A subset of keyframes that form discriminative views of the scene are kept in a vocabulary tree for use in a visual place recognition system. This allows loop closure candidates to be identified by comparing new keyframes to spatially nearby keyframes that are already in the vocabulary tree. Once potential loop closures have been detected the rigid body transformation that closes the loop is determined by aligning the matching keyframes with a RANSAC-based 3-point algorithm. They compare the use of sparse bundle adjustment and pose graph

optimisation for correcting the estimated camera poses and $3D$ map points in accordance with the loop closure. In a post-processing step a dense surfel map is generated from the intermediate keyframes. While the possible extent of mapping and the inclusion of a loop closure mechanism is impressive, the system operates at a low frame-rate (less than the $30hz$ at which many commodity RGBD sensors operate). On top of this, the final map representation is only made available after mapping is complete, which makes it unsuitable for many robotics applications that require an up-to-date map in real time.

Newcombe et al. (2011a) were among the first to put forward an approach that combined active depth sensing cameras and general purpose GPU computing for online and incremental estimation of a single dense volumetric reconstruction of a room-sized environment. Central to the approach is a dense frame-to-model predictive alignment scheme for tracking the pose of the camera in every frame. Once the camera's pose has been resolved the current frame can be fused into the global model. Internally the system represents this global model with a volumetric truncated signed distance function (TSDF). The TSDF itself stores the signed distance from each point in its volume to the nearest point on the surface it is representing. Positive distances indicate that a point is 'outside' the surface and negative distances indicate that a point is 'inside' the surface. The surface itself occurs at the zero-crossings of the TSDF. Visible points that are greater than some distance threshold from the nearest surface are clipped to a maximum allowable distance set to match the region of uncertainty around the estimated surface location. In order to store the TSDF in GPU memory it is discretised at a certain resolution into a *voxel grid*, with a voxel being the $3D$ analogue of a pixel.

During camera tracking, model-based predictions of the view of the scene are made by ray casting the TSDF volume into a vertex map and a normal map using the camera matrix from the previous time step. The camera's pose is then estimated by aligning the predicted frame to the current live frame using a variant of the ICP algorithm that is embedded in a 3 level coarse-to-fine vertex and normal map pyramid. The new camera pose estimate is then used to fuse the latest frame into the TSDF model. Voxels in the global TSDF that need to be updated are found by projecting each voxel into the current

camera frame. If a voxel projects to a valid pixel location then the ray emanating from that pixel and passing through the camera's optical centre can be computed using the camera's inverse calibration matrix. Voxels that lie on this ray and are within the truncation region are updated in accordance with the corresponding depth measurement, using a simple running average that, over the course of multiple measurements, acts to denoise the TSDF.

Newcombe et al. (2011a) and their KinectFusion system marked a pivotal point in the evolution of visual mapping, showing that live dense volumetric reconstructions were possible and in doing so their work excited a flurry of follow-on research. In particular, it initiated several tracts of research aimed at improving on certain aspects of the original algorithm's performance and removing a number of its limitations. One of the fundamental limitations of the system was that the scale of the physical spaces that could be mapped was limited to room-sized scenes since the TSDF volume was fixed in size, by the amount of GPU memory available and the chosen voxel grid resolution, and in space, by the initial position of the camera. Related to this is that while the predictive frame-to-model camera tracking pipeline of the system was capable of maintaining locally consistent trajectories, it had no explicit loop closure mechanism and so could not correct for camera drift over trajectories of a larger scale.

A further limitation of the system was its handling of scene dynamics. The running average used to fuse measurements did provide some level of robustness to transient levels of dynamic object motion, however, the system by and large assumed a static scene and so did not attempt to actively model scene dynamics. This had an impact on both camera tracking, which relied solely on scene geometry, and model accuracy. Even in situations where dynamics were not an issue the tracking process's reliance on scene geometry made it prone to failure in cases where the camera was facing surfaces with low geometric diversity. Such surfaces could not sufficiently constrain the underlying ICP objective function along each of its 6 degrees of freedom (3 for translation and 3 for rotation) leading to poor camera pose estimates in such cases.

Prior to KinectFusion the focus of dense mapping was on the recovery of the low level geometric structure of scenes, however, the advent of KinectFusion and the

capabilities it brought allowed research in the area to focus on other aspects of dense mapping. Naturally the question of how can mapping at higher levels of abstraction, such as at the level of objects or at the level of semantics, be incorporated into a dense mapping pipeline became a focal point of subsequent research in the area.

Whelan et al. (2014a) addressed a number of these problems with the development of the Kintinuous mapping system, including the problems of mapping scalability (Whelan et al., 2012), global drift (Whelan et al., 2013b) and robustness of camera tracking (Whelan et al., 2013a). First, the authors showed how the volume within which mapping is taking place can be unanchored from its initialisation position and dynamically recentred around the camera as it moves by storing the TSDF in a cyclical buffer on the GPU. This allows KinectFusion style volumetric fusion on a scale orders of magnitude larger than what is possible with the original KinectFusion algorithm. The system gathers portions of the TSDF that have 'moved out' of the volume that is currently being used for mapping and segments them into slices which are streamed to the CPU where they are used to construct a mesh representation of the map (Whelan et al., 2012).

The susceptibility of camera tracking to drift in scenes where the geometry does not provide adequate constraints is addressed by adding a photometric error term to the camera tracking objective function. The photometric term minimises the frame-to-frame pixel intensity error which acts together with the frame-to-model geometric error to provide a more robust tracking algorithm, exploiting all the frame data in a principled manner. As with the original KinectFusion algorithm, the optimisation is performed iteratively by embedding it in a 3 level course-to-fine pyramid scheme (Whelan et al., 2013a).

While the joint visual and geometric odometry of Kintinuous is robust across local camera trajectories, it was not designed for, nor was it capable of, maintaining globally consistent trajectories. To deal with this problem Whelan et al. (2013b) instituted a *backend* module that is responsible for incrementally building both a pose graph of every camera pose estimate and running a visual place recognition system to identify loop closures. When a loop closure is identified a RANSAC-based algorithm is used to make a course grained estimate of the rigid transformation between the current camera coordinate frame

and the actual coordinate frame, according to the place recognition system. This initial rough estimate is refined through Levenberg-Marquardt minimisation. The refined estimate is then added as a constraint to the pose graph which is maintained and optimised by the iSAM system of Kaess et al. (2008). The corrected pose graph is then used to bootstrap a deformation graph with surface-to-surface constraints. The map is non-rigidly deformed in accordance with the loop closure by applying the optimised deformation graph directly to the surface (Whelan et al., 2013b).

Whelan et al. (2014a) were not the only ones to address the limitations of the KinectFusion algorithm. Izadi et al. (2011) introduced novel additions to the core KinectFusion pipeline for modelling scene dynamics. Motion within the scene that is independent of the camera's motion is identified as patches of connected outliers in the projective data association stage of camera tracking. Outlier patches are segmented from the background, which is assumed to be largely static, and excluded from tracking using a outlier *mask* image. KinectFusion's camera tracking is resilient enough to transient levels of scene dynamics such that tracking can continue while dynamic scene regions are identified and masked. However, without this extension extended periods of dynamic scene motion would cause irrecoverable degradation in camera tracking.

Keller et al. (2013) showed the benefits of foregoing a volumetric map representation in favour of a flat list of surfels (surface elements that have scale and orientation). At a coarse-grained level, the pipeline of the system closely resembles that of Newcombe et al. (2011a), different map representations aside, alternating between model predictive camera tracking and depth map fusion. Here, for every frame the system performs iterative frame-to-model ICP alignment to track the motion of the camera, however, in contrast to Newcombe et al. (2011a), the authors take advantage of the explicit point list map representation, using a standard graphics pipeline to efficiently render a predicted model view and perform projective data association between the frames. Following this the live frame is fused into the model, using a weighted average to fuse intersecting points. The averaging scheme used is similar to that used by Newcombe et al. (2011a) but goes a step further, explicitly modelling the uncertainty of measurement points as Gaussian

distributions parameterised by their radial distance from the camera centre. By fusing intersecting model points in this way, the memory footprint of the map is reduced and the underlying map denoised at the same time, increasing the scale at which mapping can proceed and capturing the denoised characteristics of KinectFusion's maps.

Building on the work of Izadi et al. (2011), Keller et al. (2013) use outliers from the projective data association stage of ICP to find candidate points in the input frame that potentially lie on dynamically moving objects. Then in a hierarchical region growing step (using the coarse-to-fine image pyramid constructed in the camera tracking stage) connected regions of candidate dynamic points are identified and are used to build a dynamics map which segments the frame into static and dynamic regions. Over time confidence is lost in model points that are too frequently associated with dynamic points, and they are marked as unstable and so are not used for camera tracking making it more stable in the face of dynamic scenes. Despite the large scale over which it can map and its robustness to dynamic scene motion, the system has no explicit method for combating camera drift, leading to significant mapping and tracking errors over time.

Whelan et al. (2015) and their ElasticFusion system focuses on exploiting the benefits of a point-based fusion approach demonstrated by Keller et al. (2013) but within a more complete mapping framework that incorporates online map optimisation through local and global loop closures. By defining a temporal window that segments the map into an active and inactive region, Whelan et al. (2015) attempt to apply non-rigid deformation-based surface loop closures frequently, as the camera transitions between the active and inactive regions, maintaining local map consistency and facilitating loopy camera trajectories. To reduce global camera drift keyframes are harvested and kept in a fern encoding database, then as each new frame is processed it is checked against the database for frames containing a similar view. If such a match is found then the two frames are used to constrain a deformation graph which, when applied to the surface, non-rigidly 'closes the loop'. On top of estimating each surfel's position, normal and radius, ElasticFusion also estimates each surfel's colour, again using a weighted running average to fuse multiple measurements. During camera tracking colour information in the map is used to predict a

full colour image which is used to perform photometric frame-model tracking alongside geometric frame-model tracking, similar to Kintinuous.

More recently Dai et al. (2017) try to better approximate the maximum likelihood reconstruction characteristics of a bundle adjustment framework within a volumetric fusion mapping pipeline by performing a hierarchical global optimisation over the complete set of camera poses, allowing on-the-fly removal and reintegration of frames into the volume based on continually improving camera pose estimates. Corresponding SIFT features are found between each new frame and all previous frames. Subsequently, the correspondences go through a series of filters designed to decrease the number of false positive matches, after which they are used to hierarchically optimise camera poses over the entire sequence. At the lowest level in the hierarchy the sequence of input frames is divided into chunks of consecutive frames with some overlap between neighbouring chunks (similar to the concept of a *window* in bundle adjustment and sparse visual SLAM). The first frame in each chunk defines the coordinate frame for the entire chunk. Then, intra-chunk feature correspondences and dense photometric and geometric matches are used to construct a local intra-chunk energy function that, upon minimisation, aligns frames within a chunk. At the next level of the hierarchy, each chunk puts forward a keyframe (the first frame in the sequence) and an aggregate keyframe feature set which define an energy functional that, upon minimisation, globally aligns all chunks. At each level the final non-linear energy function is minimised with respect to the camera poses using a novel parallel optimisation framework that exploits the specific sparsity characterisitics of the energy function's Jacobian.

Schöps et al. (2019) combined a direct whole image alignment RGBD camera tracking frontend with a novel direct bundle adjustment backend optimisation. The frontend to their BAD-SLAM system tracks the camera and extracts keyframes. Keyframes are then sent to a backend which performs a global optimisation of the keyframe poses and a surfel map. The direct BA cost function is defined over both geometric and photometric constraints. In order to maintain efficiency, the optimisation alternates between updating the different parameters during each iteration. During each iteration, new surfels are

created, overlapping surfels are merged and spurious surfels are deleted. Updates on surfel normals and radii are performed outside the BA procedure to simplify optimisation. Surfel position updates are constrained to move only in the direction of the surfel normal to prevent input points with poor local gradient information from corrupting the map.

Surfel map representations confer many benefits from a SLAM point of view; they are easily updated with new information, they are easily deformed to account for loop closures and they can handle a range of scene scales. However, surfel maps are essentially discrete, which makes them unsuitable for many downstream applications which assume a continuous surface model. Schöps et al. (2020) introduce an online meshing algorithm to incrementally build meshes of surfel maps, such as those built by ElasticFusion (Whelan et al., 2015). For meshing, the surfel map is streamed to the CPU and organised in an octree to enable spatial accessing of the map, necessary for quick retrieval of surfel neighbourhoods during mesh triangulation. The basis of the mesh triangulation step is a projective mesh growing algorithm (Marton et al., 2009). When new surfels are added or old surfels are updated (either through loop closure or fusion of a new frame), a remeshing step marks locally affected regions of the mesh for remeshing. Methods for intensive regularisation of the surfel map and denoising of the input depth maps are included as mesh triangulation is highly sensitive to noisy surfels.

Other tracts of research in dense mapping includes work on more directly modelling scene dynamics and semantics. Newcombe et al. (2015) attempt to model dynamic non-rigidly deforming scenes, more comprehensively than is done in the works of Keller et al. (2013) and Izadi et al. (2011), using a per surface element *warp field* to map the surface back to a canonical pose upon each new frame from which point the frame can be fused into the reconstruction with the resultant effect being the online generation of dense models of highly dynamic scenes. Innmann et al. (2016) propose a similar approach to dynamic mapping, adding that sparse image based features can help improve surface tracking stability.

McCormac et al. (2016) use an encoder-decoder convolutional neural network (CNN) to perform per pixel semantic labelling of image pixels with class membership

probabilities in the frontend of a variant of ElasticFusion. Probabilities of pixel labels are fused into the probabilities of surfel labels using a Bayesian update rule over the probability of a label for a surfel conditioned on the previous images and the probability of pixel having that label conditioned on the current image.

Salas-Moreno et al. (2013) demonstrated dense mapping at the level of objects with their SLAM++ system. In this paradigm of dense SLAM a set of domain specific objects are presupposed to occur frequently in a specific scene. $3D$ models of these objects are captured using KinectFusion and stored in a database. Frame-to-model camera tracking proceeds much like it does in KinectFusion, however, in contrast, a model-based view prediction is made by rendering the predicted *object level* view of the scene using the estimated camera matrix from the previous time-step. Once the camera's pose has been resolved, objects in the object model database are searched for in the current frame. Pose estimates for cameras and objects form camera-camera and camera-object edges in a pose graph which is used to jointly optimise the historical camera pose estimates and object pose estimates. One drawback of their approach was strong prior assumptions about the object categories expected in the scene and their respective $3D$ geometries. McCormac et al. (2018) presented a similar method that relaxes these restrictive assumptions. Mask-RCNN (He et al., 2017) is used to identify object instances across a broad range of object categories and geometries in incoming images. Detections are fused into per-object instance TSDFs, aligned within a common global coordinate frame. The object TSDFs are joined together with a pose graph to perform object level loop closure detection and correction. Many similar works have also been presented: Rünz and de Agapito (2018) use ElastiFusion to build an object level of map of multiple moving objects; Pham et al. (2019) perform volumetric semantic mapping followed by conditional random field (CRF) refinement over higher order object level constraints to produce an instance level semantic map; Sünderhauf et al. (2017) build an instance aware semantic mapping algorithm on top of ORB-SLAM2.

More recently, SLAM systems have been proposed that explore the full scope of the representational power of neural networks. Tateno et al. (2017) propose to combine CNN-based monocular depth estimation and semantic segmentation with a LSD-SLAM

derived backend. Bloesch et al. (2018) propose a compact representation of dense scene geometry based on variational auto-encoders. During a training phase, an optimisable latent space encoding of dense geometry is learned. Following this, the latent space is used within a PTAM-like SLAM system. Keyframe poses and latent vectors representing dense scene geometry are jointly optimised. The application of learned depth codes in the context of SLAM is further explored in Czarnowski et al. (2020). Sucar et al. (2021) represent the scene as a radiance field, implemented using a multi-layer perceptron (MLP). During operation, incoming images are used to train the weights in the MLP via a reconstruction loss between the input image and a prediction rendered from the MLP. Historic keyframes are repeatedly replayed through the MLP to prevent the network from 'forgetting' old parts of the scene.

## 2.5 Collaborative SLAM

In Sections 2.1 to 2.4 we discussed single-robot SLAM. In this section, we switch focus to collaborative, or multi-robot, SLAM. Collaborative SLAM is an important SLAM paradigm. In the same way that a team of people can out-perform a single person on some tasks, a team of robots can out-perform a single robot. For example, in large scale search and rescue, a team of drones can split up and cover ground more quickly than a single robot could. Provided there is a channel through which information can be shared between robots, SLAM can also benefit from the addition of multiple robots. Multiple robot systems present many challenging research questions such as the distribution of sub-tasks and coordination between robots. In this section and in Section 2.6 we focus specifically on challenges surrounding collaborative SLAM.

In particular, when a team of robots is deployed in an a priori unknown environment, each robot must perform SLAM so that they can understand, and localise themselves within, their environment. Multi-robot SLAM systems are generally built on top of single-robot SLAM systems but must solve additional problems so that they can take advantage of the additional robots. Saeedi et al. (2016) identify the following pertinent multi-robot SLAM problems:

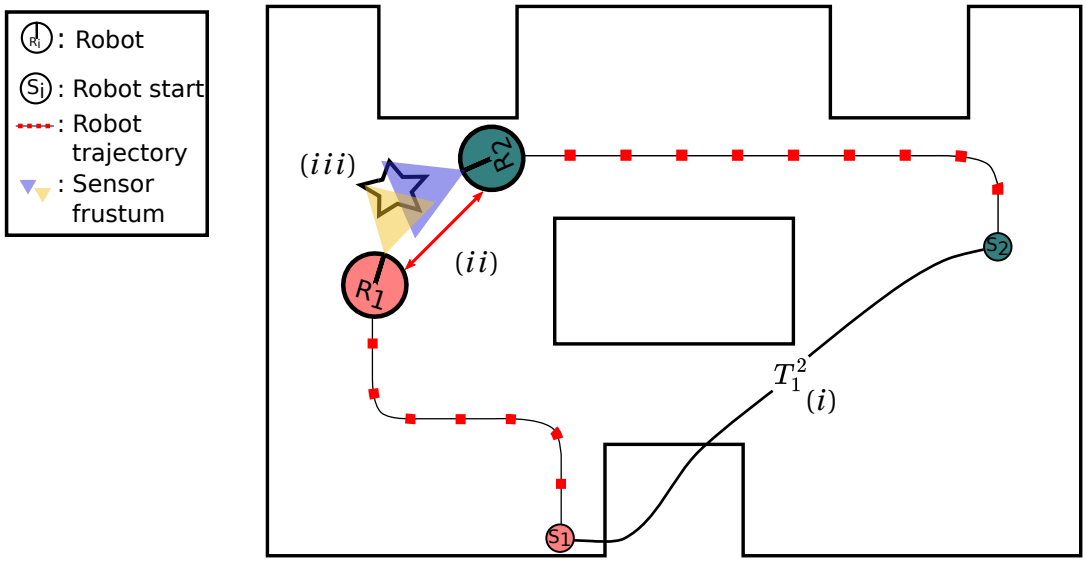



Figure 2.3: **Challenges in collaborative multi-robot SLAM (I): unknown relative poses of robots.** A hypothetical collaborative SLAM session. Two robots, $R_1$ and $R_2$ begin mapping a space. $(i)$ Neither robot has prior knowledge of the pose of the other robot relative to itself. More precisely, this means that the quantity $T_1^2$, representing the transformation between the two robots' local frames of reference, is initially unknown. $T_1^2$ could be, for example, a rigid body transformation, or it could be a similarity transform if the robots cannot directly measure the scale of the environment (e.g., if they only have single monocular camera). During the mapping session, $T_1^2$ can be determined either directly when two robots encounter each other $(ii$, red line), or indirectly, when two robots make overlapping observations, e.g., of the star $(iii)$.

- **Relative Poses of Robots**: In a general multi-robot scenario the relative transformation between robot specific coordinate systems will be unknown at the start of mapping. The other way of stating this problem is that there is no way of knowing, a priori, a global coordinate system and the pose of each robot within that coordinate system. Therefore, each robot must start in its own local coordinate system and a global coordinate system must be inferred via observations shared by multiple robots as mapping proceeds (Figure 2.3, $(i)$).

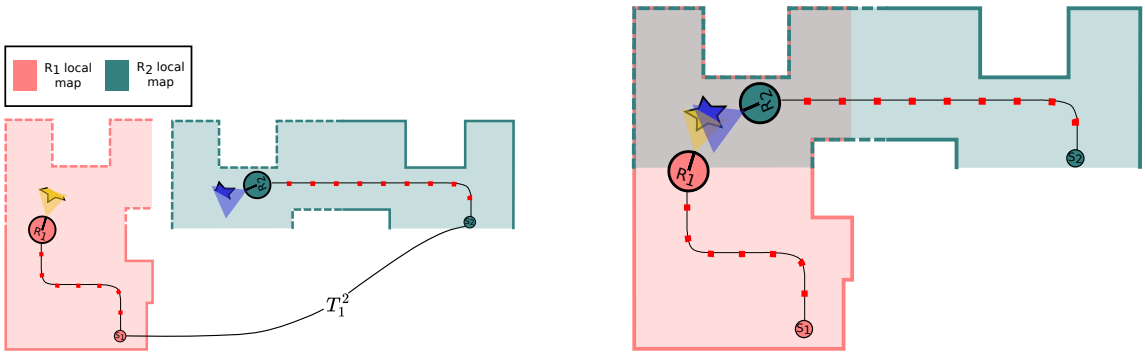
- **Loop Closure**: Loop closures in single robot SLAM require a robot to identify when it has revisited a place, irrespective of even the most catastrophic tracking failures. In a multi-robot session, inter-robot loop closures are also possible. Inter-robot loop closures occur when one robot recognises a location that has previously been visited by another robot (Figure 2.3, $(iii)$). This type of loop closure is the basis for inferring relative robot-robot poses during mapping. They are also useful even

when relative robot poses are known, as multiple loop closures can be combined to remove the effects of outliers and noise.

- **Robot-Robot Observations**: During mapping, two or more robots may directly observe each other. Observations of this kind provide a strong constraint on the relative transformations between robots (Figure 2.3, $(ii)$). This is due in part to the use of salient tags that allow robots to easily identify one another. However, the true benefit of robot-robot observations, is that, when robots meet, they can actively verify the estimated relative transformation by arranging to rendezvous at a second location. However, relying solely on direct robot-robot observations ignores constraints on relative transformations that arise when observations from two robots overlap, i.e. indirect encounters.

- **Uncertainty of Relative Poses**: In a single robot setting, uncertainty from many sources is represented via a covariance matrix. The covariance matrix includes pose-pose correlations (over poses in the robot's own trajectory), pose-landmark correlations and landmark-landmark correlations. In multi-robot settings, the total covariance of the system must also include uncertainty in the relative poses between each robot's local coordinate system.

- **Updating Maps and Poses**: Once a robot-robot loop closure has been identified and relative transformation computed, the robot local maps must be aligned and overlapping geometry merged. Robot sub-maps can be combined within a single map structure (Figure 2.4(b)) or kept in separate sub-maps joined together in a loosely coupled manifold structure (Figure 2.4(a)). In the former case, duplicated geometry is more easily identified and eliminated, but the loop closure constraint is 'baked' into the map. This can corrupt the map in the case of a false positive loop closure. In the later case, the map will more likely contain duplicated geometry but the global coordinate frame estimate can be refined with further loop closure constraints. False positive loop closures are less catastrophic with this approach. In either case, once multiple robots have been joined together in a shared map, fusion




(a) (b)

Figure 2.4: **Challenges in collaborative multi-robot SLAM (II): map representations.** Consider the same hypothetical collaborative SLAM session as in Figure 2.3. Each robot initially builds its own local map. Once an inter-map loop closure has occurred, robot specific local maps need to be combined into a unified map. Local maps can be either (a) kept separate using a manifold structure, for example a pose graph, or (b) merged together into a single global map. (a) Although manifold approaches can recover from false positive inter-map loop closures, since the maps are never explicitly combined, this robustness comes at a cost. Specifically, the unified map contains duplication of structures in the environment where the local maps overlap (dashed border). (b) merging local maps avoids duplicate representations in the unified map, but the inter-map loop closure is effectively baked into the map. This makes it difficult to correct the inter-map transformation, if the initial loop closure is wrong, or if subsequent robot-robot encounters or overlapping observations become available.

of the latest sensor measurements from each robot as well as intra-map loop closures, must be carried out in such a way that the map is not corrupted.

- **Complexity**: A SLAM system must run in real time. A multi-robot SLAM system is no different. A critical issue in single robot SLAM is the scalability of the system as the mapping session extends over large spaces and long periods of time. A new dimension of scalability must also be considered in multi-robot systems - the number of robots.

- **Communications**: Multi-robot teams must communicate over some kind of network. This network may not always be reliable. Thus, multi-robot SLAM algorithms must be robust in the face of an unreliable network. Other communication challenges include: deciding whether robots should share raw sensor data or filtered/smoothed map and pose data; implementing the SLAM algorithm as either centralised or decentralised/distributed process.

- **Heterogeneous Platforms**: In many multi-robot applications the robot teams may consist of robots with differing sensor platforms and capabilities. For example, teams of ground robots supported by drones flying overhead have been developed in several domains. From a multi-robot SLAM perspective such heterogeneity can represent a significant challenge. In the drone/ground robot team from our example, the drone's perspective is significantly different from that of the ground robot. This can make finding data associations between their respective sensor readings and maps difficult.

A multi-robot SLAM system that addresses these challenges (or at least a critical subset of them, e.g., solving the unknown initial relative poses of each robot, facilitating map and pose updates from multiple robots and maintaining real-time performance), can allow a team of robots to map an unknown space more efficiently. By combining all sensor data into a single, shared map representation, individual robots gain a more complete model of the environment as a whole, even if they have not directly explored each region. What's more, the unified map can be used to facilitate downstream collaborative tasks. For example, robots can use the shared map to coordinate further exploration of the scene (e.g., Fox et al., 2006).

## 2.6 Review of Collaborative SLAM Methods

In this section we review relevant literature on collaborative SLAM. We start with a brief discussion of the emergence of early multi-sensor SLAM systems from existing traditional single-sensor SLAM frameworks. We then focus on systems that offer solutions to the key challenges in collaborative SLAM that we face with our work in dense collaborative mapping. These challenges include; unknown initial relative poses; updating maps and poses; and solving loop closures. A recent and comprehensive review of collaborative mapping can be found in Saeedi et al. (2016). We will also cover some more recent research on multi-session SLAM, a problem that is closely related to multi-sensor SLAM and can be viewed as multiple SLAM 'missions' through the same area. Many of the systems

discussed here perform sparse and, or, low dimensional planar mapping and in most cases do not use vision based sensors like RGBD cameras or even monocular colour cameras. However, we focus on the representations used by these systems and the mathematical frameworks they use for solving the different constituent collaborative mapping problems.

Many early attempts at collaborative SLAM took pre-existing frameworks from single-sensor SLAM and extended them such that they could represent multiagent mapping. One of the earliest collaborative SLAM systems was that of Nettleton et al. (2000), who were the first to propose using the information filter to represent the multi-robot SLAM problem. In this work the authors simplify the problem by constraining the robotic platforms to 1-DOF Brownian motion and linear observations of a set of continuously observable landmark features. The key contribution of this work was the insight that the collaborative SLAM problem could be naturally represented in information space, since the final map information was simply the sum of the information gained from each robot.

Fenwick et al. (2002) use an Extended Kalman Filter (EKF) as the state estimator in a collaborative mapping system. In their algorithm, one agent starts building a feature based map using a SICK laser scanner, which a second agent will attempt to find itself in using a joint compatibility branch and bound data association algorithm (Neira and Tardos, 2001). Once the second agent is localised in the map of the first agent an augmented state and covariance matrix, representing the conjoint state of all vehicles and map points and the covariance between them, is maintained using an EKF in the same way as proposed by Fenwick (2001).

Continuing this pattern of extending pre-existing SLAM frameworks, Williams et al. (2002) build upon the Constrained Local Sub-map Filter. In this framework, a global map is built by fusing together a series of globally aligned local maps. Periodically local maps are added to the global map and a new local sub-map is initialised with the sensor's current global pose. The main benefits of this approach are that local sub-map points can be maintained in decorrelated fashion, meaning their local state estimation accuracy is independent of the global map and their initialisation pose, and that data fusion can be delayed until local sub-map accuracy is sufficiently high to support robust data association

between points in the local map and points in the global map. Intuitively, this framework is amenable to collaborative mapping with multiple agents. In their extended framework, Williams et al. (2002) maintain a single global state, consisting of a map of feature points and the position of each agent within the map. New independent sub-maps are spawned for each vehicle in the system. As each agent maps within its own local sub-map it remains uncorrelated with the other sub-maps and with the global map. Once an agent has built up a map of its local environment it can be fused into the global model. If an agent's initial pose relative to the global map is unknown then it is resolved by finding a set of corresponding map points between the agent's local map and the global map. The transformation between the agent's local map and the global map can then be found through a non-linear least-squares optimisation over the correspondences, with respect to the translation and rotation of the coordinate frames.

Thrun and Liu (2005) go back to considering the collaborative SLAM problem in information space, this time using a sparse Extended Information Filter (EIF), but again noting that the additivity and locality of information updates makes it a natural and convenient way of representing the collaborative mapping problem. Their approach permits multiple agents with unknown initial relative poses to collaboratively build a single map of the environment. Given two sub-maps the relative transformation between them is found by matching features between them to get an initial estimate of the alignment. Features for each landmark are calculated based on the relative angles and distances between configurations of triples of points that fall within a radius of the landmark in question. Triples are stored in a Sphere/Rectangle-tree for efficient matching later (Katayama and Satoh, 1997). An initial alignment estimate is calculated by matching similar configurations between maps and using geometric techniques to derive the rotation and translation between the two coordinate frames. Using the candidate alignment the space of possible correspondences between the two maps is searched for the optimal configuration. This search is guided by trying to balance the reduction in the likelihood of the map due to associating two landmarks, with the increase in likelihood due to the negative information that would be gained by both robots seeing just one landmark if in actual fact there were

two separate landmarks. During fusion, duplicate point measurements are easily collapsed into one measurement by adding the corresponding information matrix and information vector rows and columns.

Howard (2006) uses a Rao-Blackwellised particle filter as their starting point for a collaborative SLAM system. A particle filter represents the potentially non-Gaussian posterior distribution of the map and trajectory with a weighted set of sample maps and trajectories, where the weight of a sample represents its likelihood conditioned on its trajectory and the map. More specifically each particle represents a complete robot trajectory, up to the current point in time, and a separate map conditioned on that particle's trajectory. At each time step all particles are updated independently using the latest action updates and sensor measurements in a Bayesian update step. The general idea is that the weights of particles that are inconsistent with respect to their measurements and the map decrease over time. Particles with a low weight can be pruned and replaced by resampling the particle sample set, following a resampling scheme, such as randomly initialising particles in the vicinity of the most highly weighted particle.

The authors note how readily this framework can be extended to allow collaborative SLAM. Namely, by expanding the posterior distribution being estimated to include trajectories and sensor measurements of the additional agents. Like before, each particle consists of a weight and a map but now also contains multiple trajectories, one for each agent. To account for the unknown initial relative poses of agents the authors employ a *rendezvous* strategy, whereby one agent starts mapping and subsequent agents are added to the particle filter once they directly encounter the mapping agent and measure their pose relative to it. Observations made by the newly inserted agent are divided into those made prior to being inserted and those made subsequently. During the particle update step the first set are fed into the update in a reversed time ordering, acting as a virtual agent travelling backwards through the map, while the second set are fed into the update as normal. The updates to the map, trajectories and weights for each particle are themselves carried out in essentially the same manner as in the traditional single-agent particle filter. However, there are some notable differences; a reverse action model is required for updating the estimated trajectory

of the virtual reversed-time agent; the weight of each particle is now the product of the weights of each agent, both virtual and real; and the map is updated from the observations of all agents, again both virtual and real.

Howard et al. (2006) present a novel manifold map representation consisting of a series of overlapping patches. Each patch has an *extent* and a planar coordinate system. Objects on the manifold can be described by the patch they lie on and their pose relative to the patch. Patches on the manifold overlap and represents points on the underlying planar map with duplicity, however, the projections between the manifold patches and the planar map, and thus the relationships between patches, is not known *a priori*. Using a scan matching algorithm, a set of relations between corresponding points on two patches can be found. The maximum likelihood set of patch-to-plane transformations that minimises the projection error over point-to-point relations is computed using the Levenburg-Marquardt non-linear least squares technique. The manifold representation naturally lends itself to multiagent mapping, where initially each agent's map consists of an unconnected *island* in the manifold's topology, reflecting the fact that the relative pose of each agent is initially unknown. Islands are merged by identifying inter-island point correspondences. Howard et al. (2006) follow a robot rendezvous strategy to generate these correspondences.

Each agent localises itself independently of the other agents. This is done by initialising a new manifold patch around the agent's current pose estimate (incorporating odometry updates). A local map consisting of well localised nearby patches is predicted from the manifold. Note here that the notion of 'nearby' is in the context of the planar projection of the manifold patches. Relations between points in the new patch and points from the local map are found by feature matching. As described above the new patch is located with respect to local map according to the planar projection parameters that minimise the point-to-point error over the relations. The agent's pose with respect to the manifold can be found by projecting the patch back into the manifold map and finding the patch in the local map that is closest to the new patch and overlaps with it. The agent's pose with respect to this patch is the composition of the new patches pose and the inverse of the pose of the selected manifold patch. The new patch is itself added to the manifold

map if it covers a significant area of the manifold not already covered by the local map.

Loops are closed by identifying relations between unrelated patches. These relations are identified by mutual agent-to-agent observations. Identifying new relations in this way causes topological changes to propagate throughout the manifold. The loop closure mechanism alternates between identifying new relations and optimising the manifold by refitting patch parameters. This process is iterated until no new relations are identified. Islands are merged by identifying inter-island relations. Islands are formed by sets of patches that are connected, either directly or indirectly, to one another through relations. Thus the set of intersections of two islands is empty. Once inter-island relations have been identified topological changes are propagated throughout the manifold and are optimised in the same manner as loops are closed.

In the context of multi-robot SLAM, one of the main benefits of the manifold map representation is that it can be used to allow robots to directly participate in confirming inter-map loop closure candidates before they are used to correct the manifold's topology. Given that a robot-robot encounter has occurred, the two robots can agree to rendezvous at a second location. If both robots arrive at the second location and there is another direct encounter, then there is additional evidence that the inter-map loop closure is correct. Conversely, if no direct encounter occurs, then the loop closure candidate can be discarded as spurious.

Zhou and Roumeliotis (2006) use an EKF as the underlying state estimator in a collaborative mapping system. Initially each agent maintains an independent state vector and covariance matrix consisting of a pose and set of landmark feature points. Like Howard et al. (2006), a rendezvous strategy is used for aligning agent specific local coordinate frames into a single shared global coordinate frame. The relative transformation between two agents measured during the rendezvous is used to transform one agent into the coordinate frame of the other agent. Once in the same coordinate frame the local maps of each agent can be merged. During map merging duplicate landmarks are identified by considering the Mahalanobis distance (Mahalanobis, 1936) between pairs of landmarks. If the distance is less than some threshold then the landmarks are considered the same and

averaged together to increase the accuracy of the resultant map. The duplicate landmark is removed by deleting the corresponding row of the state vector and the row and column of the covariance matrix.

Castle et al. (2008) extend the original PTAM map representation to multiple sub-maps. Each sub-map is anchored to a place of interest such as a particular room. As the target domain is AR with a wearable camera, the geometric relationship between each of the sub-maps is not considered hugely important. A human can navigate between the regions of interest themselves. Rather, when the camera becomes lost (perhaps because it has 'gone off the edge' of the current map), an appearance-based relocaliser searches the keyframes of each map for a matching view. This allows the camera to switch between sub-maps if the person has moved from one interest region to another. PTAM's split between tracking and mapping naturally permits extension to multiple cameras. Each camera corresponds to its own tracking frontend while a single mapping backend builds and maintains the sub-map database. Though multiple cameras could potentially submit keyframes for addition to the same sub-map, the mapping process can easily decide to reject one in the case that both keyframes are too close together. In comparison to other multi-camera SLAM systems, provided an existing map or set of maps exist, additional cameras do not perform local mapping. Instead, they merely relocalise with respect to the existing database of maps from which point they can start tracking and mapping.

Kim et al. (2010) extend the iSAM system of Kaess et al. (2008) to formulate collaborative mapping as a pose graph optimisation problem. A pose graph consists of nodes, which represent poses, and edges between nodes which represent spatial constraints between nodes. The map can be recovered by optimising the spatial configuration of nodes in the graph. Initially each agent's trajectory is represented with what is essentially an independent pose graph. Two new edge types are introduced into the graph to encode direct encounters, which occur when two agents observe each other, and indirect encounters, which occur when two agents observe the same scene. These inter-graph edges connect nodes between two agents pose graphs. The spatial and encounter constraints encoded in the graph edges are used to solve a non-linear optimisation in the agent poses using

iSAM. To solve the problem of unknown initial relative poses an anchor node is prepended to each pose graph. The anchor for a pose graph encodes its offset from some canonical global coordinate frame. With the inclusion of anchor nodes, encounter constraints must pass through each of the corresponding anchor nodes such that the encounter is specified in the global coordinate frame.

The works discussed so far use either direct encounters (Howard, 2006; Howard et al., 2006; Zhou and Roumeliotis, 2006), indirect encounters (Castle et al., 2008; Fenwick et al., 2002; Thrun and Liu, 2005), or both direct and indirect encounters (Kim et al., 2010) to estimate the unknown initial relative pose of each robot. Although direct encounters may result in a very clear and unambiguous estimate of the relative pose of two robots, relying solely on such encounters has a couple of drawbacks. First, direct encounters, as the name suggests, rely on two robots meeting at the same point in time and mutually recognising each other. In practice, guaranteeing that such a meeting occurs requires solving a circular problem - both robots would need a map to agree on where to meet. Thus, such an approach leaves a lot to chance. Second, robots need to be equipped with a means of recognising each other. For example, Kim et al. (2010) fit each robot with an easily detected checkerboard while Howard (2006) and Howard et al. (2006) fit each robot with a tag that can be easily identified by a robot's laser scanner.

Indirect encounters do not rely on individual robots being able to recognise each other. Rather, robots attempt to match their observations with observations made by other robots. All available observations, including historical ones, can be used for this task, increasing the possibility for an encounter. One of the main drawbacks of this approach is that observations can exhibit aliasing; if two observations, taken from distant regions of the environment, appear identical when measured by a robot's sensor then an incorrect match is possible, which, in turn, can lead to a corrupted map. While not impossible, incorrect matches are far less likely in direct encounter mechanisms that use markers specifically designed for accurate detection. In the dense collaborative SLAM system developed in Chapter 3 of this thesis we take an indirect approach. Each agent keeps a database of distinctive keyframes. Individual agents can search the keyframe databases of other agents

for a view that matches their current frame. If a match is found then an indirect encounter has occurred and the relative pose of the agents involved can be estimated. Matching in our system is based on both texture and structure, witch mitigates the risk of aliasing in texture-less scenes or scenes with low structural variation.

Forster et al. (2013) introduce a centralised algorithm for collaborative SLAM with multiple MAVs. Each MAV performs its own feature detection, visual odometry and keyframe extraction. Keyframes and their initial pose, calculated via VO, are streamed from each MAV to a central server running collaborative SfM. The central server integrates each MAV's keyframes into a local map and begins searching for inter-map loop closure constraints. Scale drift in the per-MAV VO and scale drift in the centralised SfM module occur at different rates. This scale divergence is tracked and accounted for during map building. Once an inter-map loop closure has been detected, the two triggering sub-maps are geometrically aligned with 7-DOF similarity transformed and merged into a single map. Intra and inter-map loop closures are corrected with a scale drift aware pose graph optimisation similar to Strasdat et al. (2010a). In a merged map, where keyframes from more than one MAV are used to update the map, *mutex* locks around keyframes are used to facilitate coherent multi-threaded updates to the map.

Riazuelo et al. (2014) devise a cloud-based collaborative SLAM system called C$^2$TAM. Built on PTAM, the system runs a local tracking frontend client on each edge node and a global BA mapping backend on a server in the cloud. Frontend nodes send tracked keyframes to the mapping server which responds, at a much lower frequency, with an updated map. Initially, separate maps are maintained for each frontend node. Visual place recognition across all maps in the database provides inter-map loop closure candidates. If a loop is accepted, the two maps are aligned and fused into a single map. Once a merge is complete, each frontend client receives a map delta update, consisting of the keyframes and map points of the other map. In the case of tracking failures, Riazuelo et al. (2014) distinguish between a robot that is lost due to a transient issue, such as an occlusion, and a robot that is catastrophically lost. In the former case, the robot is likely to still be in the same sub-map and, therefore, the relocalisation module can limit its search to

this sub-map. In the latter case, the robot has been lost for some time or has just joined the mapping session, therefore, it can be in any one of a number of maps. In this case, a two stage relocalisation scheme allows the frontend client to relocalise against all maps in the database. The first stage finds the most relevant keyframe across all maps in the database and sends this keyframe, along with a set of neighbouring keyframes, to the client. The second stage searches among the keyframe neighbourhood for a more accurate location in case of camera movement during server-side relocation.

Some recent work on the problem of multi-session mapping is relevant to this thesis. The multi-session system of McDonald et al. (2013) is representative of this work, and we will discuss it here. Labbé and Michaud (2014) is similar, but includes a memory management strategy to reduce the computational load and maintain real-time performance. McDonald et al. (2013) present a system that is somewhat similar to that of Kim et al. (2010) in that the underlying state estimator is a pose graph and that anchor nodes are used to specify the relative constraints between mission specific pose graphs. However, McDonald et al. (2013)'s pose graph representation extends that of Kim et al. (2010) to 6-DOF mapping and tracking with a stereo camera. The system is split into a frontend responsible for intra-session visual odometry and windowed bundle adjustment optimisation. A separate backend is responsible for place recognition and pose graph optimisation, performing the multi-session aspects of the system. Frame-to-frame visual odometry is calculated in three phases. First a GPU-based implementation of the Kanade-Lucas-Tomasi (KLT) tracker is employed to track a set of features into the current frame. Next a RANSAC model fitting is used to compute an initial estimate of the camera's pose. This initial estimate is refined using Levenberg-Marquardt to minimise the reprojection error. Sequences of consecutive frames are segmented into windows which are either incrementally or batch-wise optimised with respect to the poses, map and features. Once a window grows to a certain size it is passed to the backend where it is used to extend the inter-session pose graph and a new frontend window is initialised around the current camera pose. Both the frontend window optimisation and the backend pose graph optimisation are performed using the iSAM system.

Similar to the approach of Kim et al. (2010), each agent's trajectory is represented by a separate pose graph. Session specific pose graphs are related to each other via anchor nodes within a common global coordinate frame. Inter and intra-session loop closures are identified using a visual place recognition system. The loop itself is closed in the same way as frame-to-frame camera tracking is performed. Intra-session loop closures induce constraints between nodes within a single pose graph, whereas inter-session loop closures induce constraints between nodes in separate pose graphs and so are connected through their anchor nodes such that they are specified in the global coordinate frame.

ORBSLAM-Atlas proposes a multi-mapping extension to ORB-SLAM (Elvira et al., 2019). The Atlas data structure contains multiple sub-maps where each sub-map contains its own map data similar to the original ORBSLAM algorithm (Mur-Artal et al., 2015). Intra-sub-map loop closures can be found by searching the currently active sub-map's place recognition database for loop closure. Inter-sub-map loop closures can be found by searching for loop closure candidates within the place recognition database of other non-active sub-maps. Once a inter-map loop closure has been identified the two triggering maps are aligned together and overlapping map points are merged. A local BA in the region of overlap helps to fuse the two maps together. This provides the basis for multi-session mapping with ORBSLAM-Atlas allowing for ORB-SLAM to be initialised with sub-maps relevant to the current mapping session. ORBSLAM-Atlas was included as part of the ORBSLAM3 release (Campos et al., 2021).

Collaborative mapping has come back into focus in SLAM research in recent times. Golodetz et al. (2018) propose a novel approach to dense collaborative mapping that uses the InfiniTAM system of Prisacariu et al. (2017) to construct *separate* voxel-based maps for each camera. The system uses a random forest relocaliser to find estimates of the relative transforms between each of these camera specific maps. These relative transforms act as constraints in a pose graph which, once optimised, yields a set of optimal global poses, one for each of the sub-maps. Although these global poses align the sub-maps into a common reference frame they are never merged directly, instead each camera maintains its own sub-map throughout. This allows for the inter-map loop closure constraint between

two sub-maps to be refined with subsequent observations (Golodetz et al., 2018).

Dubois et al. (2020) propose a decentralised multiagent dense mapping framework based on peer-to-peer sharing of sub-maps. Each mapping agent builds a series of TSDF sub-maps and a factor graph of odometric and inter-sub-map constraints. When two agents are within communication range they synchronise map information with one another, transferring any sub-map information they acquired since their last rendezvous. Initially, agents keep the map information from each of the other agents they encounter in a separate reference frame. Once sufficient overlap between the agent's map and its copy of another agent's map has been detected, the maps are merged. It is important to note that, since this algorithm is decentralised, each agent runs its own instance of the algorithm and builds its own local copy of the full multiagent map.

As noted above, the system proposed by Golodetz et al. (2018), uses a pose graph to encode the relative transformation between agent specific sub-maps while keeping the sub-maps themselves separate. Although this representation may be beneficial when the objective is to quickly map a large space, and it is the final model itself that is of interest, it has one main drawback; it leads to replication of structures where multiple cameras traverse the same regions of the environment, even after inter-map loop closures have been corrected. For example, if multiple agents all map the same chair, then there are multiple chairs in the final map. The fact that there is, in actuality, only a single chair can only be discovered when all instances of the chair are dereferenced to a single frame of reference. This level of indirection could potentially be cumbersome in scenarios where robots are using the collaborative SLAM system to facilitate collaborative operation within a shared workspace. Since the pose graph is subject to refinement, this dereferencing operation may need to be repeated multiple times and would be much simplified if there was only one map, shared by all robots. Additionally, in large-scale mapping, the additional storage required by the system due to duplication in the map would be more pronounced. In contrast, our system, and the system of Dubois et al. (2020), performs map merging, allowing multiple cameras to fuse their measurements into a single *shared* reconstruction. This allows cameras to directly exploit and update regions of the environment previously

mapped by other cameras within the system, avoiding duplication of structures. One of the limitations of this approach is that inter-map loop closures are 'baked' into the map once sub-maps are merged.

The network architecture of a dense collaborative SLAM system has a significant impact on its performance characteristics. Dubois et al. (2020) implement a fully decentralised architecture where each agent runs its own instance of the full algorithm. Agents then synchronise their map data with other agents that are within communication reach. On the one hand, this approach is fault-tolerant, where failures of individual agents are guarded against by redundantly replicating the full map across all agents. On the other hand, synchronising map data across all agents requires sharing a significant amount of information. Each agent must then process all the map data it receives. Doing this across multiple agents requires a significant amount of compute resources, albeit in a distributed manner. Our algorithm, described in Chapter 3, and that of Golodetz et al. (2018), opt instead for a centralised approach. Golodetz et al. (2018) follow a client-server architecture. Clients are responsible for producing accurate local poses (poses in the frame of reference of the clients sub-map), then sending frame-pose pairs to a central mapping server. The central mapping server is responsible for fusing the posed frames into dense sub-maps for each agent as well as building the pose graph that connects all the sub-maps together. Splitting responsibilities between clients and the server in this way frees server resources so that they can be focused on building a global scene representation. The main drawback of this approach is that there is no mechanism for reflecting local intra-map loop closures (i.e. loop closures within a sub-map) in the global map being built by the central server. This essentially limits clients to running some form of accurate visual, or visual-inertial, odometry pipeline for real-time operation. If a client, running a full SLAM pipeline, were to undergo a loop closure correction, then its poses would become misaligned from its corresponding server-side dense sub-map. However, without closing loops agents have no way of reducing drift or understanding the true topology of the world, i.e. each sub-map will be homotopic to a long corridor (Cadena et al., 2016), no matter how accurate the odometry pipeline being used is.

In our system, we take a fully centralised approach. Clients send raw and un-posed frame data to a single central SLAM server that performs all mapping and tracking operations. The system does not require complex peer-to-peer synchronisation and processing of map data across mapping agents, as is the case in Dubois et al. (2020). Additionally, in contrast to Golodetz et al. (2018), our system performs intra-map loop closures on local, agent-specific sub-maps, prior to inter-map loop closures, and on global maps shared by multiple agents, once inter-map loop closures have been solved.

## 2.7 Summary

The goal of this thesis is to close the gap between the state-of-the-art in dense visual SLAM and what is required for outdoor, multiagent dense mapping. In order to put this goal into context, in this chapter we reviewed key developments in three SLAM paradigms: sparse visual SLAM, dense visual SLAM and collaborative SLAM. For each paradigm, a description of the central problems presented by the paradigm was provided and then followed by a review of key developments from the relevant literature. First, in Sections 2.1 and 2.2, we reviewed sparse VSLAM. Sparse VSLAM systems tend to perform probabilistically principled joint estimation of scene geometry and camera motion. Although both feature-based and direct approaches to sparse VSLAM have been explored, feature-based approaches in particular offer accurate camera tracking even during challenging outdoor conditions and fast camera motion.

One of the main limitations of sparse systems is the sparsity of the scene representations they produce. In Sections 2.3 and 2.4 we reviewed dense VSLAM. While visually impressive, early dense VSLAM systems have a number of limitations; they either don't perform loop closures (Newcombe et al., 2011a,b; Pollefeys et al., 2008), don't propagate loop closures to dense geometry if they do perform them (Newcombe and Davison, 2010; Stühmer et al., 2010, to the extent that the underlying PTAM system closes loops), or don't actually produce a dense map in real time (Henry et al., 2012). Later, dense VSLAM systems were developed that did close loops, correct dense surface geometry and produce dense maps in real time (e.g., Dai et al., 2017; Schöps et al., 2019; Whelan et al., 2014a,

2015). These systems are not suitable for outdoors applications where scene lighting may be adverse, camera motion fast, and potentially non-standard fisheye cameras are being used, for example automotive applications. This is in large part due to their reliance on active depth sensors, use of direct camera tracking techniques (in the case of Schöps et al., 2019; Whelan et al., 2014a, 2015), and their assumption that the camera can be modelled using the pinhole camera model. In Chapter 5 we present a dense visual SLAM system that can operate in challenging outdoor conditions by combining the robust feature-based camera tracking of ORBSLAM-3 (Campos et al., 2021) with the dense SLAM provided by ElasticFusion (Whelan et al., 2015) in a hybrid SLAM architecture. Reliance on an active RGBD sensor is eliminated by using a convolutional neural network to predict a depth map for every image. In Chapter 6, we build on this system so that it can support fisheye camera geometries.

A further limitation of the dense VSLAM systems discussed in Sections 2.3 and 2.4 is that they are inherently single-sensor systems. They do not allow for multiple independently moving sensors to collaboratively reconstruct a unified representation of the environment being explored. In Sections 2.5 and 2.6 we introduced the problem of multiagent collaborative SLAM and discussed developments in the area, starting with early developments on sparse collaborative SLAM and finish on more recent developments on dense collaborative SLAM. In Chapter 3 we present a novel dense collaborative SLAM system that is the first contribution of this thesis.

# CHAPTER 3

# Collaborative Dense SLAM

In this chapter we present the first contribution of this thesis: a system for multi-camera collaborative dense SLAM. The system extends ElasticFusion (Whelan et al., 2015), a state-of-the-art surfel-based dense-alternation VSLAM system. The chapter starts with an in-depth description of ElasticFusion in Section 3.1. Following this, in Section 3.2, we describe how we extended ElasticFusion to support collaborative SLAM. Finally, in Section 3.3, we provide both quantitative and qualitative analyses of our approach to collaborative mapping using the synthetic ICL-NUIM dataset (Handa et al., 2014) and the real-world Freiburg dataset (Sturm et al., 2012). In our analyses we measure the impact of multi-camera mapping on surface reconstruction accuracy, camera pose estimation accuracy and overall processing time. We also include qualitative results in the form of sample reconstructions of room-sized environments with up to 3 cameras undergoing intersecting and loopy trajectories.

Our system builds on ElasticFusion to allow a number of cameras starting with unknown initial relative positions to maintain local maps utilising the original algorithm. Carrying out visual place recognition across these local maps the system can identify when two maps overlap in space, providing an inter-map constraint from which the system can derive the relative poses of the two maps. Using these resulting pose constraints, our system performs map merging, allowing multiple cameras to fuse their measurements into a single *shared* reconstruction. The advantage of this approach is that it avoids replication of structures subsequent to loop closures, where multiple cameras traverse the same regions

of the environment. Furthermore, it allows cameras to directly exploit and update regions of the environment previously mapped by other cameras within the system.

The work in this chapter was published in

- L. Gallagher and J. McDonald. Collaborative Dense SLAM. In *Proceedings of the 2018 IPRCS, Irish Machine Vision and Image Processing Conference*, 2018

## 3.1 Dense SLAM With ElasticFusion

This section will describe the EF dense mapping algorithm (Whelan et al., 2015, 2016). The aim of this section is to equip the reader with an understanding of the core EF algorithm necessary for full appreciation of the dense collaborative mapping algorithm that will be presented in the remainder of the chapter. However, we recommend the reader refer to the original sources listed above for a more comprehensive treatment.

### 3.1.1 RGB-D Sensors

A key component of EF is the sensor it uses to make measurements of its surroundings. These measurements are the inputs from which the system incrementally builds a model of the scene. Many types of input sensor have been used in SLAM research from passive sensors, such as monocular colour cameras (e.g., Davison et al., 2007), to active sensors, such as sonar (e.g., Leonard and Durrant-Whyte, 1991). The type of sensor used by EF is of the later type. More specifically it uses an active depth sensing colour camera (or RGB-D sensor for short) which consists of an integrated colour camera and infrared radiation (IR) projector and sensor pair, offset from one another by a short baseline $B$. Depth values for each pixel are estimated using the principle of *structured light* whereby a known pattern is projected onto the scene using the IR projector and then imaged with the IR sensor. Disparities (i.e. horizontal motions) between the image of the projected pattern and a reference image of the pattern at a known depth are calculated. Using these disparities the

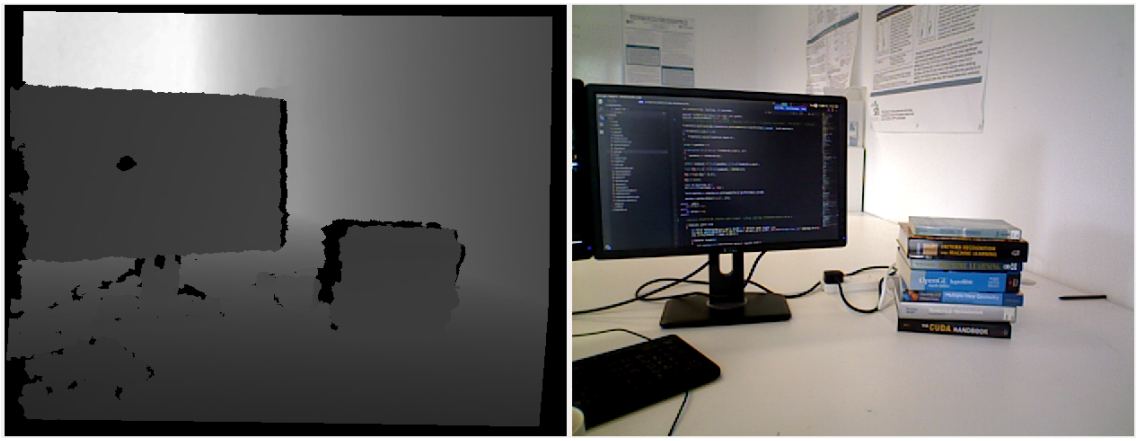


Figure 3.1: An example of an RGB-D frame captured by the ASUS Xtion Pro Live. On the right we see a colour image of the scene. On the left we see a depth map visualised as a greyscale image. Lighter grey levels represent greater distances. Black indicates that no depth was measured for that pixel location.

depth, $Z \in \mathbb{R}$, of a pixel can be retrieved using the following relationship

$$d = f\frac{B}{Z} \tag{3.1}$$

where $B$ is the baseline separating the projector and sensor, $f$ is the focal length of the sensor, and $d$ is the disparity. See Szeliski (2010, § 12.2 & §11) for more details of active range finding techniques. Both of the sensors shown in Figure 1.5 are examples of RGB-D sensors that can be used with EF. An example of an RGB-D frame is depicted in Figure 3.1.

The input to EF is a series of RGB-D frames $\mathcal{F}_t$, $t \in [0,...,m)$. Each frame $\mathcal{F}_t$ is a tuple of the form $\langle I_t, \mathcal{D}_t \rangle$ for colour image $I_t$ and depth map $\mathcal{D}_t$. A colour image $I$ is a mapping from image domain $\phi \subset \mathbb{N}^2$ to colour channels $\{r, g, b\} \in \mathbb{N}^3$ where $r$, $g$ and $b$ represent the red, green and blue spectral responses of a given pixel location. A depth map $\mathcal{D}$ is a mapping from the image domain to depth values $d \in \mathbb{R}$. The value of the depth map at each pixel location corresponds to the perpendicular distance from the $XY$ plane of the camera coordinate frame to the $3D$ point that maps to that pixel under perspective projection (see below).

### 3.1.2 Camera Geometry

The camera's estimated trajectory is represented within a global frame of reference centred at the origin of a 3-dimensional Cartesian coordinate system. The camera's trajectory consists of sequence of camera matrices, $\mathbf{P}_t$, representing the camera's internal and external orientation within the global reference frame at each point in time $t$. Each camera matrix $\mathbf{P}_t$ is a $3 \times 4$ homogeneous matrix of the form

$$\mathbf{P}_t = \mathbf{K}[\mathbf{R}_t|\mathbf{t}_t] \tag{3.2}$$

the matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the camera's calibration matrix, and it describes the internal orientation of the camera, meaning it describes the orientation of the image plane with respect to the camera's coordinate frame. The matrix $\mathbf{R}_t \in SO(3)$ and the vector $\mathbf{t}_t \in \mathbb{R}^3$ describe a rotation and a translation, respectively, representing the camera's external orientation at time $t$, that is, the orientation of the camera's coordinate frame with respect to the world coordinate frame. ElasticFusion assumes a pinhole camera model and therefore the structure of the calibration matrix $\mathbf{K}$ is as follows

$$\mathbf{K} = \begin{bmatrix} fm_x & 0 & c_x m_x \\ 0 & fm_y & c_y m_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where $f$ is the focal length of the camera, $m_y$ and $m_x$ are the number of pixels per unit length in the $x$ and $y$ directions respectively. The point $(c_x m_x, c_y m_y)$ is the principal point of the image plane and is the origin of the camera's sensor array with respect to the image plane. It is given in units of pixels.

In EF, it is assumed that all frames are captured by a single pre-calibrated moving camera, thus during the tracking phase of the pipeline only the external orientation, also known as the *pose*, of the camera, is estimated. Therefore, $\mathbf{K}$ can be factored out of the

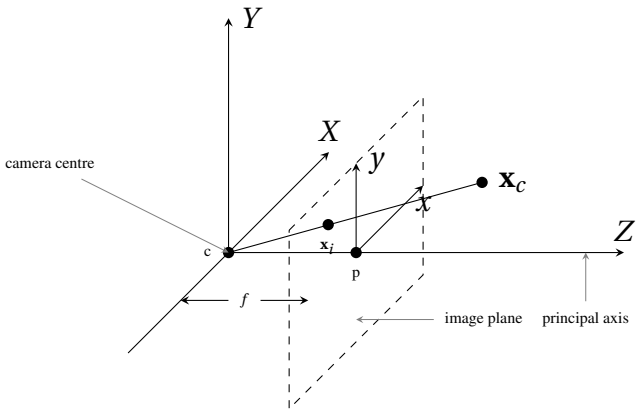



Figure 3.2: Perspective projection of a point $\mathbf{x}_c$, specified with respect to the camera's coordinate frame, onto the image plane defined by the camera calibration matrix $\mathbf{K}$, resulting in the image point $\mathbf{x}_i$. Note that the origin of the image plane coordinate system is here assumed to be coincident with the principal point of the camera coordinate frame, i.e. $(c_x, c_y)^T = (0,0)^T$, which is not the case in general but is convenient for visualisation purposes. The reader should direct their attention to Hartley and Zisserman (2003, Ch. 6) for more details of camera calibration where it is discussed at great length.

definition of $\mathbf{P}_t$, leaving the following simplified form

$$\mathbf{P}_t = \begin{bmatrix} & \mathbf{R}_t & & \mathbf{t}_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

where $\mathbf{R}_t \in SO(3)$ and $\mathbf{t}_t \in \mathbb{R}^3$ are a rotation and a translation as before. The set of orientation preserving orthogonal matrices $\{\mathbf{R} \in \mathbb{R}^{3\times3} | \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}^{3\times3}, det(\mathbf{R}) = +1\}$ form what is called the *special orthogonal* group, or $SO(3)$ for short, and together they describe the space of possible $3D$ rotations. Taking $SO(3)$ together with the set of possible $3D$ translations forms the *special Euclidean* group $SE(3)$, the space of all possible rigid body motions where $SE(3) = \{\left(\begin{smallmatrix}\mathbf{R} & \mathbf{t}\\ 0 & 1\end{smallmatrix}\right) | \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3\}$. $SE(3)$ is a *Lie* group and as such it is also a differentiable manifold. An important characteristic of Lie groups is that they can be locally linearised at a point $\mathbf{P} \in SE(3)$ to produce a Lie algebra $\hat{\zeta} \in se(3)$, which corresponds to the tangent space of the manifold at $\mathbf{P}$. The exponential map $\exp : se(3) \rightarrow SE(3)$ maps elements of the Lie algebra onto corresponding elements of the Lie group. Ultimately, a Lie algebra gives a first order linear approximation to the manifold of the Lie group at a given point, which, as we discuss further in Section 3.1.3, offers a means of incrementally estimating the transformation undergone by a camera between two frames by composing

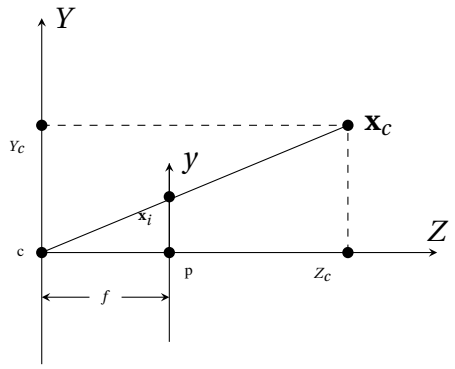



Figure 3.3: Looking at the $YZ$ plane of Figure 3.2 we can see that the $y$ component, $y_{cam}$, of $\mathbf{x}_i$ can be found since $\frac{f}{Z_c} = \frac{y_{cam}}{Y_c} \implies y_{cam} = f\frac{Y_c}{Z_c}$. A similar argument can be applied to find $x_{cam}$, the $x$ component of $\mathbf{x}_i$.

a series of infinitesimal linearised transforms, using exp to map them back into $SE(3)$. A more detailed discussion of the algebraic structure of $SE(3)$ and its utilisation in $3D$ reconstruction can be found in Ma et al. (2003).

During the imaging process a $3D$ point is mapped from the $3D$ world onto the $2D$ image plane of the camera. EF models this imaging as a perspective projection through an idealised pinhole camera. Given a point $\tilde{\mathbf{x}}_w = [x, y, z, 1]^T$, represented in homogeneous world coordinates, its $2D$ image $\tilde{\mathbf{x}}_i$, also represented in homogeneous coordinates, on a camera plane, represented by $\mathbf{P}_t$, defined as in Equation (3.4), can be found by first transforming $\tilde{\mathbf{x}}_w$ into the camera's coordinate frame, yielding $\mathbf{x}_c$, and then finding the point of intersection between the image plane and the ray that starts at $\mathbf{x}_c$ and passes through the camera's centre of projection. The point of intersection can be found by way of similar triangles as shown in Figure 3.3. This computation can be performed directly using the matrix expression

$$\tilde{\mathbf{x}}_i = \mathbf{K}[\mathbf{R}|\mathbf{t}]\tilde{\mathbf{x}}_w \tag{3.5}$$

to get the final $2D$ image point in inhomogeneous coordinates we define the *dehomoginisation* operation $\pi$ which rescales the image space coordinate vector such that the final coordinate is equal to 1 and then drops it

$$\pi(\tilde{\mathbf{x}}_i) = (x_{cam}/z_{cam}, y_{cam}/z_{cam})^T = \mathbf{x}_i \tag{3.6}$$

where $x_{cam}$, $y_{cam}$ and $z_{cam}$ are the $x$, $y$ and $z$ components respectively of $\tilde{\mathbf{x}}_i$. This process is visualised in Figure 3.2 along with the layout of the camera's coordinate frame. Conversely, using a metric depth map $\mathcal{D}_t$, the $3D$ *back-projection* of a $2D$ image point $\mathbf{x}_i$ in a camera frame with pose $\mathbf{P}_t$ can be found by first inverting the perspective projection of the point onto the image plane and then transforming it from the camera's coordinate frame into the global coordinate frame. Again, this calculation can be succinctly expressed in matrix algebra

$$\mathbf{x}_c = (\mathbf{K}^{-1}[\mathbf{x}_i|1]^T)\mathcal{D}(\mathbf{x}_i) \tag{3.7}$$

$\mathbf{x}_c$, now in camera centred coordinates, can be transformed to homogeneous world coordinates

$$\tilde{\mathbf{x}}_w = \mathbf{P}^{-1}\,[\mathbf{x}_c|1]^T \tag{3.8}$$

### 3.1.3 Main Processing Loop

Over time successive frames are fused into a single model, or map. This map is represented as an unordered list of surface elements (surfels), $\mathcal{M}$. Each surfel is an estimate of a discrete patch centred around a point that lies on the continuous surface underlying the scene. A surfel is characterised by its position $\mathbf{p} \in \mathbb{R}^3$, normal $\mathbf{n} \in \mathbb{R}^3$, radius $r \in \mathbb{R}$, weighting $w \in \mathbb{R}$, colour $c \in \mathbb{N}^3$, initialisation time $t_0$ and the time it was last observed $t$. The attributes of position, normal, radius and colour describes a surfel's physical location, orientation, scale and colour. The weight of a surfel is intended to signify the level of confidence in its estimation. $\mathcal{M}$ is split along temporal lines into two non-intersecting sub-lists $\Theta$ and $\Phi$. $\Theta$ contains surfels that are considered active, meaning they have been observed by the camera within a fixed period of time prior to the current timestep. Similarly, $\Phi$ contains those surfels that have not been observed within this same period of time. Visualisations of a surfel and of a surfel map are presented in Figure 3.4

Frames are processed sequentially, in the order that they are captured and one at a time. Initially, the camera's centre of projection is assumed to be at the origin of the global coordinate frame with its *principal axis* (i.e. $Z$-axis) coincident with the $Z$-axis of the global coordinate frame. The main processing pipeline is as follows.

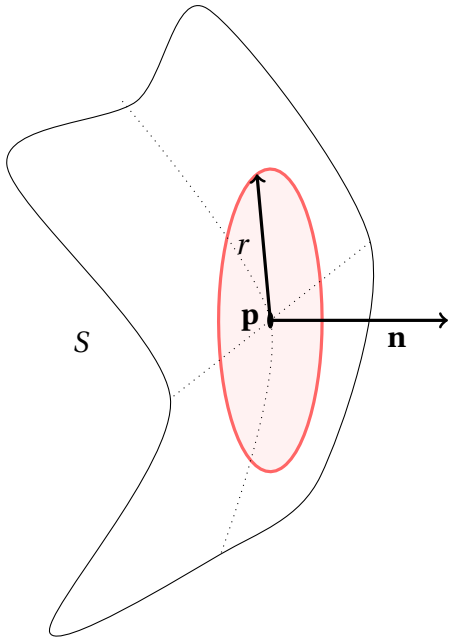



Figure 3.4: A visualisation of a single surfel lying on a hypothetical surface $S$ in an idealised situation. The main plane of the surfel is tangent to $S$ at the point $\mathbf{p}$, that is, $\mathbf{p}$ lies on $S$ and $\mathbf{n}$ is normal to $S$ at $\mathbf{p}$. The radius of the surfel $r$ determines the extent of the region in which the surfel is said to represent the local geometry of $S$. The value of $r$ is set to minimise visible holes and is determined by the resolution of the sensor. During fusion, the radius, and the surfels other geometric and photometric properties, are refined by moving the sensor closer to $S$ or by viewing the point $\mathbf{p}$ from a less acute angle. However, conversely, measurements viewing the surface point from further away or from a more acute viewing angle, are not used to update the surfels geometric and photometric properties as they would coarsen the model. The surfels time $t$ and confidence weighting $w$ are updated in either case.

### Tracking

Camera tracking is the first stage of the pipeline, the aim of which is to compute an initial estimate of the pose, $\hat{\mathbf{P}}_t \in SE(3)$, assumed by the camera at that frame. Note that it is only an initial estimate since the estimated pose of the camera is subject to change in subsequent stages of the pipeline. To track the camera EF employs a *frame-to-model* alignment scheme. A predicted view of the model, consisting of a depth map and a colour image, is rendered from $\Theta$, the active region of the map, using $\mathbf{P}_{t-1}$, the pose of the camera at the previous timestep, to define the image plane. The predicted frame is aligned to the current live frame by finding an estimate of $\mathbf{T} \in SE(3)$, the rigid body transformation between them. The alignment itself is expressed as an energy minimisation over the following joint geometric

and photometric error between the two frames

$$E_{joint}(\boldsymbol{\zeta}) = E_{geom}(\boldsymbol{\zeta}) + w_{rgb}E_{photo}(\boldsymbol{\zeta}) \tag{3.9}$$

$$E_{geom}(\boldsymbol{\zeta}) = \sum_k \left(\left(\mathbf{v^k} - \exp\left(\hat{\boldsymbol{\zeta}}\right)\mathbf{T}\mathbf{v_t^k}\right)\cdot\mathbf{n^k}\right)^2 \tag{3.10}$$

$$E_{photo}(\zeta) = \sum_k \left(g\left(\mathbf{u^k}, I_t^l\right) - g\left(\pi\left(\mathbf{K}\exp\left(\hat{\boldsymbol{\zeta}}\right)\mathbf{T}p\left(\mathbf{u}^k, \mathcal{D}_t^l\right)\right), I_{t-1}^p\right)\right)^2 \tag{3.11}$$

$\mathbf{T} \in SE(3)$ is the current estimate of the transformation from the previous camera pose to the current one, $\boldsymbol{\zeta} = (\boldsymbol{\omega}, \mathbf{x})^T \in se(3)$ is a 6-vector in the tangent space of $SE(3)$ at $\mathbf{T}$, consisting of rotational component $\boldsymbol{\omega}$ and translational component $\mathbf{x}$. In matrix form $\boldsymbol{\zeta}$ is given by

$$\hat{\boldsymbol{\zeta}} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \mathbf{x} \\ \mathbf{0} & 0 \end{bmatrix} \tag{3.12}$$

where $[\boldsymbol{\omega}]_\times$ is the skew symmetric form of $\boldsymbol{\omega}$ and $\mathbf{0}$ is a column 3-vector of zeros. The function $\exp(\hat{\boldsymbol{\zeta}})$ is the matrix exponential that maps members of the Lie algebra $se(3)$ to members of the Lie group $SE(3)$. The weighting $w_{rgb}$ is used to control the relative importance of the photometric term in the optimisation. The vectors $\mathbf{v^k}$ and $\mathbf{n^k}$ are the $k^{th}$ model predicted vertex and normal respectively, $\mathbf{v_t^k}$ is the $k^{th}$ vertex from the current frame. The model normal is used here since it is deemed to be of a higher quality (Newcombe, 2012). The function $p$ performs the $3D$ back-projection of image point $\mathbf{u^k} \in \phi$ using a depth map $\mathcal{D}$. The function $g$ computes the grey level of image $I$ at pixel site $\mathbf{u^k}$ from the pixels $r$, $g$ and $b$ components.

$E_{joint}$ is iteratively minimised with respect to $\boldsymbol{\zeta}$ using the Gauss-Newton non-linear least squares method. At the start of each iteration correspondences are found using the projective data association technique of Blais and Levine (1995). The basic assumption behind projective data association is that the frames are initially roughly aligned meaning that vertices in the source frame can be easily associated with vertices in the destination frame by projecting them into the destination image plane, using the current estimate of $\mathbf{T}$ to transform them into the destination coordinate frame. Within each iteration an incremental update that descends the error surface defined by the cost function is found

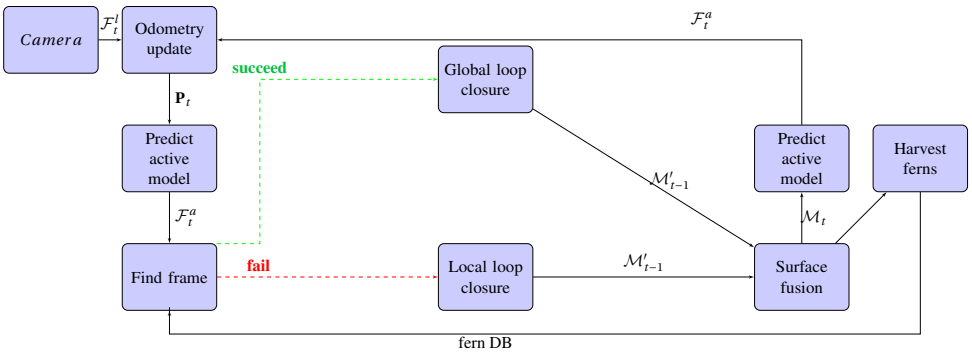



Figure 3.5: ElasticFusion's processing pipeline (Whelan et al., 2016).

by computing the tangent space of $SE(3)$ at the point $\mathbf{T}$, which is the vector space of directional derivatives of $SE(3)$ at $\mathbf{T}$. The tangent space offers a linear Euclidean space over which small, incremental updates to $\mathbf{T}$ can be computed and the overall cost function minimised. Setting the gradient of the linearised cost function with respect to $\boldsymbol{\zeta}$ to zero leads to the *normal equations*

$$\mathbf{J}^T\mathbf{J}\boldsymbol{\zeta} = -\mathbf{J}^T\mathbf{r} \tag{3.13}$$

for Jacobian $\mathbf{J}$ and residual vector $\mathbf{r}$, which can be solved for $\boldsymbol{\zeta}$ using Cholesky decompostion to exploit the sparsity of the final Jacobians.

The minimisation is embedded in a three level Gaussian pyramid, constructed from the input depth map and colour image. This ensures that in cases of large inter-frame displacement the minimisation will still converge. Corresponding images at each level in the pyramid are aligned. Each level uses the alignment computed in the previous level to initialise its own alignment, starting at the coarsest level in the pyramid and working towards the original full resolution image, performing a number of minimisation iterations at each pyramid level. To initialise the first level the two images are assumed to be coincident and so the initial transformation between them is set to the $4 \times 4$ identity matrix (i.e. no translation or rotation).

The result of this minimisation is a transformation $\mathbf{T} \in SE(3)$ that when applied to $\mathbf{P}_{t-1}$ yields an up to date estimate of the global pose of the camera for this time step.

### Global Loop Closures

Over time, as the camera moves through the scene and it is tracked, the estimation of its pose tends to drift from its true pose in the world. This drift is a consequence of many factors interacting, such as the limitations of the numerical computation techniques used to track the camera and the noisy nature of sensor measurements, that ultimately leads to an accumulation of error in the estimation of the cameras pose and of the map. This error becomes most apparent when the camera loops back around on its own trajectory. Even though the camera has come back to the same place in the world, due to drift the system estimates the camera to be in a different location which leads to a duplicated representation in the map of what is, in reality, the same surface. This is not an issue specific to ElasticFusion. Indeed, many SLAM systems (if not all that have been published to date) exhibit this problem. The attempt to first identify when a camera has looped back on its own trajectory in this way and then reduce the accumulated error by rectifying the map and camera poses is what is meant by the term global loop closure. There are three main phases of operation in closing loops. Firstly, a mechanism for recognising when the camera is revisiting a previously mapped location is required. This can be, for example, a visual place recognition system that stores visually distinctive *keyframes* and then periodically attempts to match incoming frames with keyframes in the store. A mechanism of this kind is what is used by EF, as we will discuss below. Once a loop has been identified a way of globally repositioning the camera based on the loop closure candidate is needed. Finally, the geometric constraints imposed by the loop closure must be propagated throughout the map and the full trajectory of camera poses in order to correct them.

In EF global loop closures are identified with a fern-based visual place recognition system as proposed by Glocker et al. (2015). In brief, a fern encoding is a way of encoding an RGB-D image into a binary code which allows for frames to be efficiently compared with one another, where the similarity of frames is measured in terms of the hamming distance between their respective fern encodings. To encode frames, each fern of a *conservatory* of ferns is assigned a fixed location uniformly sampled from the image

domain. A fern consists of 4 nodes, one for each image channel, and each node in turn carries a randomly sampled binary threshold parameter. The conservatory of ferns is embedded into incoming frames and the thresholded response at each node of each fern is computed resulting in a series of binary codes, one for each fern. These binary codes are used to index into a look-up table that maps from fern code blocks to the ids of frames that have the same fern code. This table along with the corresponding frame poses constitutes what is referred to as a fern-encoding database. One can think of this process as being akin to feature extraction and matching, where the fern codes correspond to features and the table look-up corresponds to feature matching. By counting the number of co-occurring blocks one can compute the block-wise hamming distance between frames, or between a frame and a set of frames, and can thus get a measure of the similarity between them. Incoming frames that are sufficiently dissimilar to all previously encoded frames (i.e. the hamming distance is large) are harvested and kept in the database as keyframes. Similarly, where the hamming distance between an incoming frame and a frame in the fern database is low a potential loop closure has been found.

After the global pose of the camera has been tracked in the current time step, it is used to render an updated predicted view of the active model. The new active model predictions are then used to search the fern database for a matching view. If one is found then the two views are registered together in the same way that the camera pose is updated, bringing the current active model prediction into alignment with the matched fern. The final transformation from this registration is used to generate a set of surface-to-surface constraints. These constraints in turn are used to optimise the nodes in a space deforming graph, which is similar to the one put forward by Sumner et al. (2007) but connects nodes temporally and includes two additional cost functions, one to pin the inactive model in place and the other to ensure previous deformations are not 'undone' by subsequent ones. Once optimised, the graph can be applied directly to the surface. To do this for each surfel the set of graph nodes nearest to it in time are found, from which the closest nodes to it in space are taken. This set of nodes forms what is called the influence region for that surfel. The final pose of a surfel is given by the weighted application of the affine

transformations held in each of the nodes in the surfel's influence region. The weight of a given node is a function of its distance from the surfel, with nodes that are further away having less influence. A deformation is accepted if two conditions are met: (i) the alignment between the current model active prediction and the matched fern is successful; (ii) the deformation requires a significant realignment of the maps geometry and that realignment is consistent with the map's geometry as defined by the cost function for the deformation graph optimisation. If the deformation is accepted then the pose of the camera is updated as per the registration with the fern view.

#### Local Loop Closures

Local loop closures between $\Theta$ and $\Phi$ are sought, assuming that a global loop closure has not occurred at the current time step. The result of a local loop closure is the alignment of the active and inactive portions of $\mathcal{M}$ and the reactivation of inactive surfels. Applying local loop closures regularly helps to maintain local surface consistency during locally loopy trajectories. Local loops are closed by performing a surface registration between the portion of $\Theta$ in view of $\mathbf{P}_t$ to the portion of $\Phi$ in the same view. Similar to global loop closures this registration yields a set of surface-to-surface constraints that are used to optimise the nodes of a deformation graph. The deformation graph is optimised in the same way as in global loop closures except the cost function that ensures the integrity of past deformations is not included in the optimisation. Once the deformation graph has been applied the inactive and active portions of the map are brought into alignment and as before the pose of the camera is updated in accordance with its registration with the inactive view.

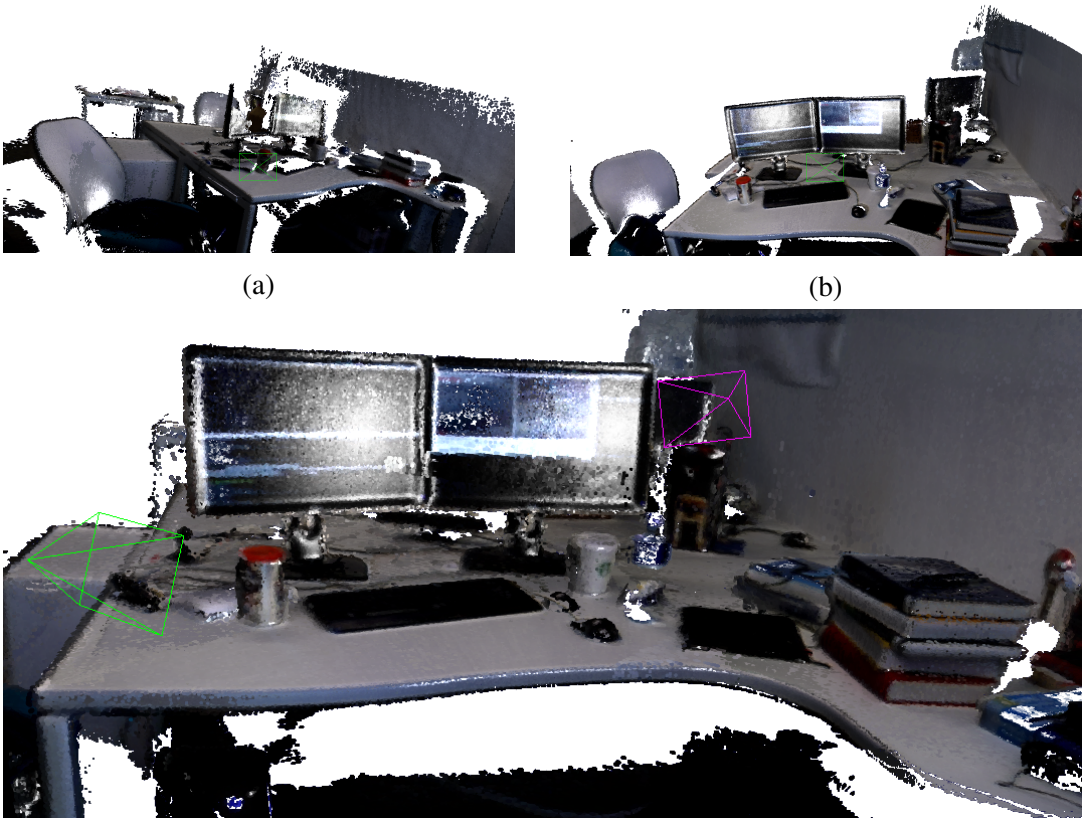
#### Fusion

Fusion of the current frame can proceed once the camera has been tracked and all loop closures have been resolved. Surfels in $\Theta$ are projected onto the image plane of the camera using $\mathbf{P}_t$ and $\mathbf{K}$ and are stored in an *index map*. Vertices in the current frame can then be associated with surfels in the current active model by searching the index map for valid

surfels in the neighbourhood of the respective vertex image coordinates. Validity here is determined by the angle between the normal of the surfel and the normal of the camera vertex, along with the distance of the surfel from the ray defined by the vertex. The normals, vertices, colours and radius of associated points are merged with a simple weighted average of the model surfel and the newly measured surfel. The weight of the surfel is updated in accordance with the normalised radial distance of the measured surfel from the camera centre, under the assumption that measurements increase in accuracy the closer they are to the camera centre and the assumption that any inaccuracy in measurements follows a Gaussian distribution. If there is no corresponding surfel for a vertex then a new unstable surfel is initialised and inserted into the model. Surfels become stable after their confidence counter has reached a certain threshold through repeated measurement, from which point they can be used for camera tracking.

## 3.2 Collaborative ElasticFusion

In this section we present the principle contribution of this chapter whereby we extend the original single-camera ElasticFusion dense mapping system to allow for multi-camera collaborative mapping. In extending ElasticFusion to permit collaborative mapping we identify two distinct phases of processing for any given input stream. In the initial phase, the system assumes that each camera is positioned at the origin of a frame of reference that is independent to that of other cameras, essentially processing via an independent EF mapping pipeline. As mapping progresses the aim is for inter-camera loop closures to occur, thereby providing the necessary transformations between each camera's sub-map. These transformations permit alignment of sub-maps into a common frame of reference which makes it possible to fuse the measurements from each camera into a single global map. Figure 3.6 shows a visualisation of a collaborative mapping session consisting of two cameras mapping a small desk environment with our system.



(a) (b)

(c)

Figure 3.6: A two camera collaborative mapping session in our system. All cameras started in general position with no *a priori* knowledge of the relative poses between them. Two cameras (a) and (b) compute local maps using EF. (c) A fern based visual inter-map loop closure allows the local maps of (a) and (b) to be merged.

### 3.2.1 Centralised Collaborative SLAM Architecture

We implement our collaborative dense SLAM system using a centralised client-server architecture. During a collaborative mapping session, a set of $n$ camera client nodes $\{\mathcal{C}_i\}, i \in [1..n]$ explore a shared space. As they explore, they send raw RGBD frames to a central server that runs the full collaborative SLAM process. Communication between clients and the server is handled by a flexible data interface subsystem built on top of Lightweight Communications and Marshalling (LCM; Huang et al., 2010). The purpose of the data interface is to provide a layer of abstraction between the specific number and type of physical cameras and the SLAM system. Two types of clients are supported by our data interface: $(i)$ Untethered clients consisting of an RGBD camera attached to a mobile computer, e.g., a laptop; and $(ii)$ log files containing previously captured exploration

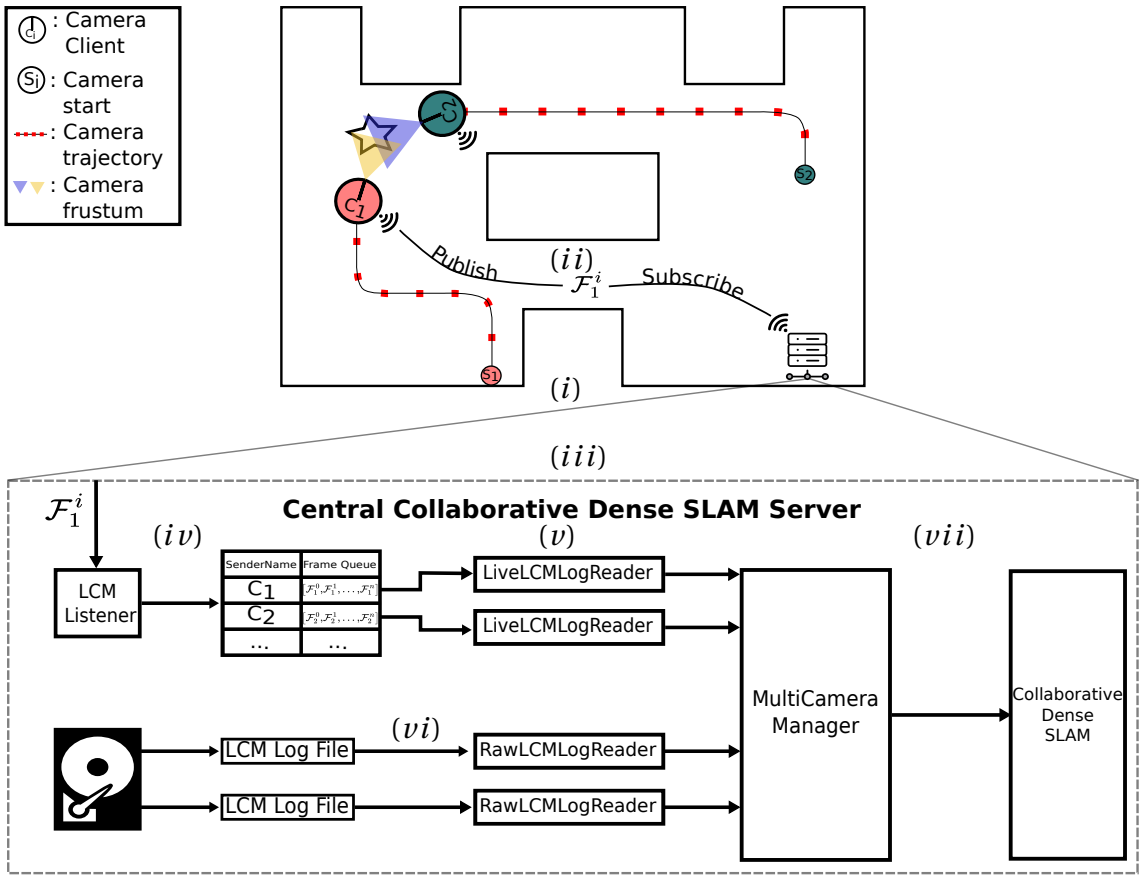



Figure 3.7: **Centralised collaborative SLAM architecture.** $(i)$ A hypothetical collaborative mapping session involving two live camera clients, $\mathcal{C}_1$ and $\mathcal{C}_2$, and two pre-recorded exploration sequences, is used to illustrate our centralised architecture. As $\mathcal{C}_1$ and $\mathcal{C}_2$ explore, they send frames to a central server using LCM's publish/subscribe semantics. $(ii)$ Frame $\mathcal{F}_1^i$, i.e. the $i^{\text{th}}$ frame from camera client $\mathcal{C}_1$, is encoded using the specification outlined in Figure 3.8 and published on a known channel. $(iii)$ The server-side LCM listener node, which subscribes to that channel, intercepts $\mathcal{F}_1^i$. $(iv)$ The value of $\mathcal{F}_1^i$'s 'senderName' field is used to place $\mathcal{F}_1^i$ in the frame queue corresponding to camera $\mathcal{C}_1$. $(v)$ Each frame queue is encapsulated by a separate LiveLCMLogReader object, which is used by the SLAM system to access the frames from each camera. $(vi)$ Two log files, containing previously captured sequences are loaded up from server storage and encapsulated in RawLCMLogReader objects. $(vii)$ MultiCameraManager collates all RawLCMLogReader and LiveLCMLogReader object instances into a single vector, polymorphically converting the data type of each object to the LogReader parent class type. MultiCameraManager then exposes this vector of LogReader objects to the SLAM system, thereby decoupling the SLAM system from the specific type and number of cameras.

sequences. Untethered clients and the central mapping server are assumed to be connected to the same local area network. Any combination of untethered clients and log files can be used together to map a space. For example, a session could consist of an untethered client and a log file, or two untethered clients, and so on. The data interface consists of three components:

**(i) Message type specification:** We use LCM's type specification language to define a common format for all messages passed between untethered clients and the server, and for encoding data from exploration sequences into log files. We define a 'Frame' message type, laid out according to the specification given in Figure 3.8. This specification is known and agreed upon ahead of time by both the clients and the server. Based on this specification, LCM automatically generates language specific bindings that include efficient methods for encoding Frame objects into a binary format suitable for transmission over the network and, subsequently, for decoding binary messages back into Frame objects.

**(ii) Message passing:** Frame messages are passed between untethered clients and the server using LCM's publish/subscribe semantics (Huang et al., 2010). During mapping, clients package frame data into Frame objects and publish them on a known channel (e.g., 'DenseSLAM'). Internally, LCM publishes a message by first encoding the Frame object and then broadcasting it over the local area network via UDP multicast. Meanwhile, the SLAM server runs a listener end-point on a separate thread that subscribes to the same channel. When the server intercepts messages on the channel it attempts to decode them back into Frame objects. To handle the fact that multiple cameras will be connected during a collaborative mapping session, the server listener node stores a hash table, mapping each camera to a queue of Frame objects. Each queue stores the frames received from the corresponding camera. Each client is assigned a different name, which it uses to fill out the 'senderName' field of the Frame objects it sends. The server-side listener then uses the value of the 'senderName' field to query the hash table and retrieve the correct queue to store the frame.

In addition to publishing messages on the network, clients can also compile Frame messages into a log file using LCM's log file functionality (Huang et al., 2010). Log files can then be used in two ways. Firstly, the log file can be played back over the network to simulate a live, untethered camera. Secondly, the log file can be stored directly on the server and used as a data-stream (see below).

**(ii) Data-stream abstraction:** Our data-stream abstraction layer extends ElasticFusion's

existing mechanism for encapsulating data sources[1]. In short, ElasticFusion uses an abstract interface called LogReader to avoid having a dependency on a specific type of data source, e.g., a log file or a live camera connected to the computer via USB. The LogReader class exposes two key members; $(i)$ member buffers containing the current RGB and depth frame in the image sequence, and; $(ii)$ a getNext member method that loads the next frame in the sequence into those buffers. The concrete implementation of getNext is left to child classes that derive the abstract LogReader interface. ElasticFusion depends only on the LogReader abstraction and not on any specific concrete implementation of LogReader.

We provide two concrete implementations of LogReader; RawLCMLogReader, for LCM log files, and LiveLCMLogReader, for Live cameras connecting to the server via the message passing mechanism outlined above. When working together, the server listener node and the LiveLCMLogReader establish a producer-consumer relationship. The listener node, running on its own thread, acts as a producer by placing incoming Frame messages onto queues. The first message received by a client is used to instantiate a new LiveLCMLogReader that contains a reference to its corresponding Frame queue. The SLAM system then consumes the frames in these queues through the getNext method of each LiveLCMLogReader instance.

The multiple, and potentially varied, clients used during a collaborative mapping session are handled by a MultiCameraManager class. This class is responsible for instantiating concrete LogReaders based on the data sources being used in the current mapping session. MultiCameraManager exposes a vector of LogReaders that the SLAM system iterates through, calling the getNext method of each LogReader to retreive the next frame in the stream for processing. This effectively decouples the SLAM system from the specific type and number of data sources. An illustration of how the architecture achieves its decoupling effect is given in Figure 3.7.

This architecture is extremely modular and can be easily extended to support many types of client - provided the camera intrinsics are known, and the camera client can package its frame data into Frame objects. In theory, any number of camera nodes can come online

[1] https://github.com/mp3guy/ElasticFusion

```
package eflcm;

struct Frame
{
  boolean trackOnly; // if true, then this frame is for camera tracking purposes only and is not fused.
  boolean compressed; // is the frame data compressed or not.
  boolean last; // is this the last frame in the input stream.

  int32_t depthSize; // the size in bytes of the depth data for the frame
  int32_t imageSize; // the size in bytes of the image data for the frame

  byte depth[depthSize]; // the depth data for the frame
  byte image[imageSize]; // the image data for the frame

  int64_t timestamp; // the wall time of the frame

  int32_t frameNumber; // monotonic timestamp for this frame, i.e its position in the current input stream

  string senderName; // the LCM channel name of the sender. Distinguishes frames coming from different cameras.
}
```

Figure 3.8: LCM Packet structure for encapsulating frame data. Each data field is accompanied by a brief explanatory comment (green text opposite the data field).

and join the collaborative mapping session by publishing their frames on the shared LCM channel.

### 3.2.2 Multi-Camera Mapping

Given that there are multiple cameras, each incoming frame is of the form $\mathcal{F}_i^t$ where the superscript $t$ indicates the frame's timestamp and the subscript $i$ denotes that this frame is from camera $\mathcal{C}_i$. We note that the cameras need not be synchronised in time.

If we assume momentarily that all cameras are globally aligned then multi-camera fusion proceeds in essentially the same way as in the original EF algorithm. At each time-step $t$ the frames from each camera for that time-step are processed in *sequence* by EF. That is, each frame is processed in full by EF before the next camera's frame for that time-step is processed. Note that the ordering of the cameras here is arbitrary.

An important difference between our system and EF is that each camera has its own active and inactive map regions, $\{\Theta_i\}$ and $\{\Phi_i\}$. This allows a camera to close local

loops in situations where it transitions to regions of the map that are in the active regions of other cameras, but that are inactive with respect to itself. To achieve this, the previously scalar last seen time attribute of a surfel takes the new form of an vector $t = \{t_0, ..., t_n\}$ where $t_i$ contains the time that the surfel was last seen by camera $\mathcal{C}_i$, for $i \in [0..n]$. When a measurement from frame $\mathcal{F}_i^t$ is fused into surfel $\mathcal{M}^s$ then the last seen time of that surfel is updated as $\mathcal{M}^s_{t_i} = t$. Each camera uses its own active and inactive regions for mapping and tracking. For global loop closure detection, all cameras in a given *reference frame* (see below) collectively maintain a single Fern database which they use to independently check for and close global loops.

Returning now to the initial phase of processing where cameras are operating independently and the transformations of each camera relative to a shared coordinate frame are unknown. Periodically, loop closures between camera specific sub-maps are sought. In Section 3.2.3 we outline our strategy for determining inter-map loop closures. Once a loop between two sub-maps has been identified and the transformation between them computed the two sub-maps can be merged into a single map from which point multi-camera fusion can proceed as described above. We capture this situation with a *reference frame*, which is a tuple of the form $\mathcal{R}_j = \{\mathcal{M}_j, \{\mathcal{C}_i\}\}$, for surfel map $\mathcal{M}_j$ and set of cameras $\{\mathcal{C}_i\}$ which incrementally update and track against $\mathcal{M}_j$ through multi-camera fusion. Additionally, each $\mathcal{R}_j$ maintains its own fern database for inter and intra-map loop closures to which all cameras $\{\mathcal{C}_i\}$ under $\mathcal{R}_j$ contribute.

Each camera connects to the central mapping server by sending its first frame over the shared channel. A hash map is used to track the relationship between cameras and their reference frames. Upon connection to the server, a camera is assigned a unique key value. Initially, this key value points to a newly initialised reference frame for that camera. As inter-map loop closures are corrected, the hash map is updated to reflect the fact that multiple cameras may share each reference frame.

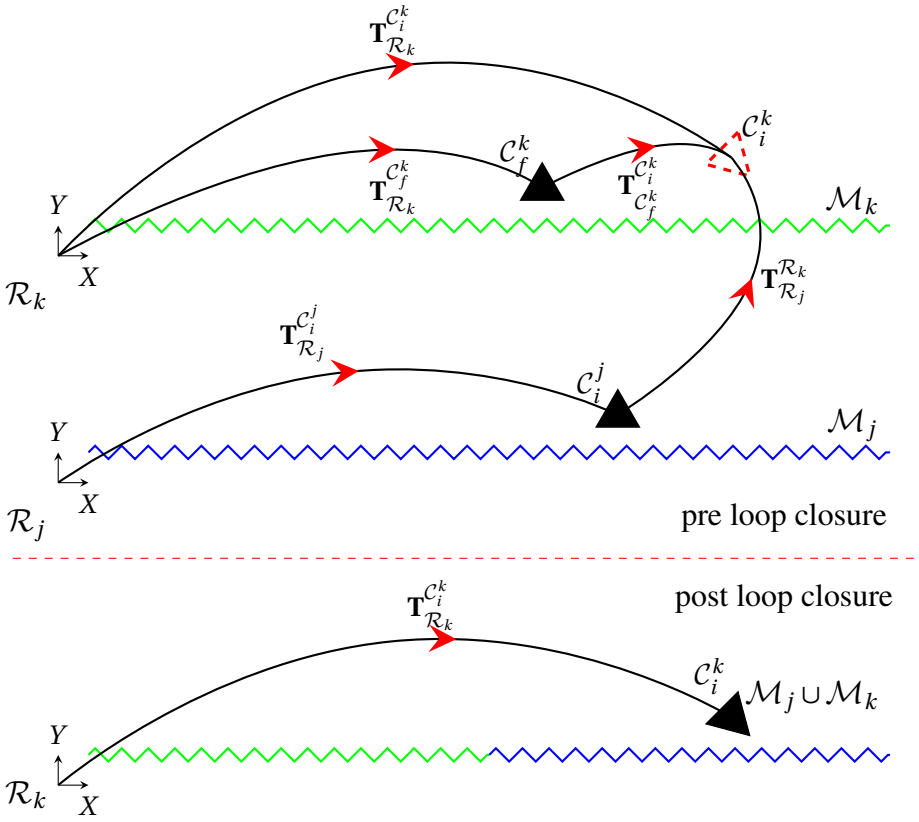



Figure 3.9: **Geometry of a fern based inter-map loop closure.** Camera $\mathcal{C}_i^j$ matches its current view of $\mathcal{M}_j$ to a fern in $\mathcal{R}_k$'s fern database. The surfaces in each frame are geometrically consistent and so a loop is closed between $\mathcal{R}_j$ and $\mathcal{R}_k$. The loop closure itself results in $\mathcal{C}_i^k$, the pose of $\mathcal{C}_i$ in $\mathcal{R}_k$, the reference frame of map $\mathcal{M}_k$. However, by composing transformations as follows we can transform the pose of $\mathcal{C}_i$ and the surfels of $\mathcal{M}_j$ into $\mathcal{R}_k$, the reference frame of $\mathcal{M}_k$; let $\mathbf{T}_{\mathcal{R}_k}^{\mathcal{C}_i^k} = \mathbf{T}_{\mathcal{R}_k}^{\mathcal{C}_f^k} \mathbf{T}_{\mathcal{C}_f^k}^{\mathcal{C}_i^k}$ then $\mathbf{T}_{\mathcal{R}_j}^{\mathcal{R}_k} = \mathbf{T}_{\mathcal{R}_k}^{\mathcal{C}_i^k} \left( \mathbf{T}_{\mathcal{R}_j}^{\mathcal{C}_i^j} \right)^{-1}$

### 3.2.3 Fern-based Inter-Map Loop Closures

Our approach to inter-map loop closures is an extension of EF's existing intra-map global loop closure mechanism described in Section 3.1. For each current active frame prediction $\mathcal{F}_i^a$, we search the fern databases of all the other reference frames for a matching view. If one is found then we compute the transformation between the reference frame of the corresponding camera, $\mathcal{C}_i$, and the reference frame of the matched fern using the camera odometry from Section 3.1. For example, if we find a loop closure between reference frames $\mathcal{R}_j$ and $\mathcal{R}_k$ by matching frame $\mathcal{F}_i^a$, captured by camera $\mathcal{C}_i^j$ in reference frame $\mathcal{R}_j$, to a fern in the fern encoding database of $\mathcal{R}_k$ then the result of aligning the fern and the frame is a transformation $\mathbf{T}_{\mathcal{C}_f^k}^{\mathcal{C}_i^k} \in SE(3)$ that aligns the matched fern and $\mathcal{F}_i^a$. For clarity, we

have added the superscripts to the camera to denote the frame of reference of the camera. $\mathbf{T}_{\mathcal{C}_f^k}^{\mathcal{C}_i^k}$ gives an initial estimate of the transformation that aligns $\mathcal{R}_j$ with $\mathcal{R}_k$ as follows

$$\mathbf{T}_{\mathcal{R}_k}^{\mathcal{C}_i^k} = \mathbf{T}_{\mathcal{R}_k}^{\mathcal{C}_f^k} \mathbf{T}_{\mathcal{C}_f^k}^{\mathcal{C}_i^k} \tag{3.14}$$

$$\hat{\mathbf{T}}_{\mathcal{R}_j}^{\mathcal{R}_k} = \mathbf{T}_{\mathcal{R}_k}^{\mathcal{C}_i^k} \left( \mathbf{T}_{\mathcal{R}_j}^{\mathcal{C}_i^j} \right)^{-1} \tag{3.15}$$

$\mathbf{T}_{\mathcal{R}_k}^{\mathcal{C}_i^k}$ in Equation (3.14) represents the pose of camera $\mathcal{C}_i$ in reference frame $\mathcal{R}_k$. When composed with $\left( \mathbf{T}_{\mathcal{R}_j}^{\mathcal{C}_i^j} \right)^{-1}$ in Equation (3.15) we get the transformation from $\mathcal{R}_j$ to $\mathcal{R}_k$.

To further refine the transformation and verify that it is valid, we first predict a full resolution frame, ${}_k\mathcal{F}_i^a$ using map $\mathcal{M}_k$ and $\mathcal{C}_i^k$, which is the camera $\mathcal{C}_i$ in frame of reference $\mathcal{R}_k$. That is, we predict the view that $\mathcal{C}_i$ has of map $\mathcal{M}_k$. Taking this predicted frame we align it to the original frame $\mathcal{F}_i^a$, again using the camera odometry described in Section 3.1. The result of this is a refined estimate, $\mathbf{T}_j^k$, of the transformation from $\mathcal{R}_j$ to $\mathcal{R}_k$. Before accepting the loop closure as a valid one we inspect the final error and inlier count of the camera tracking cost function. For the loop closure to be accepted the error must be low while the number of inliers must be high. Inliers here refer to depth measurements in $\mathcal{F}_i^a$ which have been associated with a corresponding depth value in ${}_k\mathcal{F}_i^a$. As per the original EF algorithm the final state of the cost function covariance matrix is also checked to ensure that the cost function was sufficiently constrained along each of the 6 axes of camera motion. If all these conditions are met then the inter-map loop closure is accepted and the maps are merged.

To merge the two maps we apply $\mathbf{T}_j^k$ to the positions and normals of all the surfels $\mathcal{M}^s \in \mathcal{M}_j$, and all the camera poses $\mathbf{T}_{\mathcal{R}_j}^{\mathcal{C}_i}$ in frame of reference $\mathcal{R}_j$ as follows

$$\begin{aligned} \mathcal{M}_{k_p}^s &= \mathbf{T}_j^k \mathcal{M}_{j_p}^s \\ \mathcal{M}_{k_n}^s &= \mathbf{R}_j^k \mathcal{M}_{j_n}^s \\ \mathbf{T}_{\mathcal{R}_k}^{\mathcal{C}_i^k} &= \mathbf{T}_j^k \mathbf{T}_{\mathcal{R}_j}^{\mathcal{C}_i^j} \end{aligned} \tag{3.16}$$

where $\mathbf{R}_j^k \in SO(3)$ is formed from the upper left $3 \times 3$ sub-matrix of $\mathbf{T}_j^k$. Vector $\mathcal{M}_{j_p}^s$

represents the original position of surfel $\mathcal{M}^s$ in reference frame $\mathcal{R}_j$, while vector $\mathcal{M}^s_{k_p}$ represents the surfel's position once it has been merged into reference frame $\mathcal{R}_k$, post inter-map loop closure. Similarly, vectors $\mathcal{M}^s_{j_n}$ and $\mathcal{M}^s_{k_n}$ represent the surfel's normal vector before and after merging respectively. Note that we have not explicitly shown conversions to and from homogeneous representations for brevity's sake. This process is visualised in Figure 3.9. The algorithm is listed in Algorithm 1. The full system's architecture is visualised in Figure 3.10.

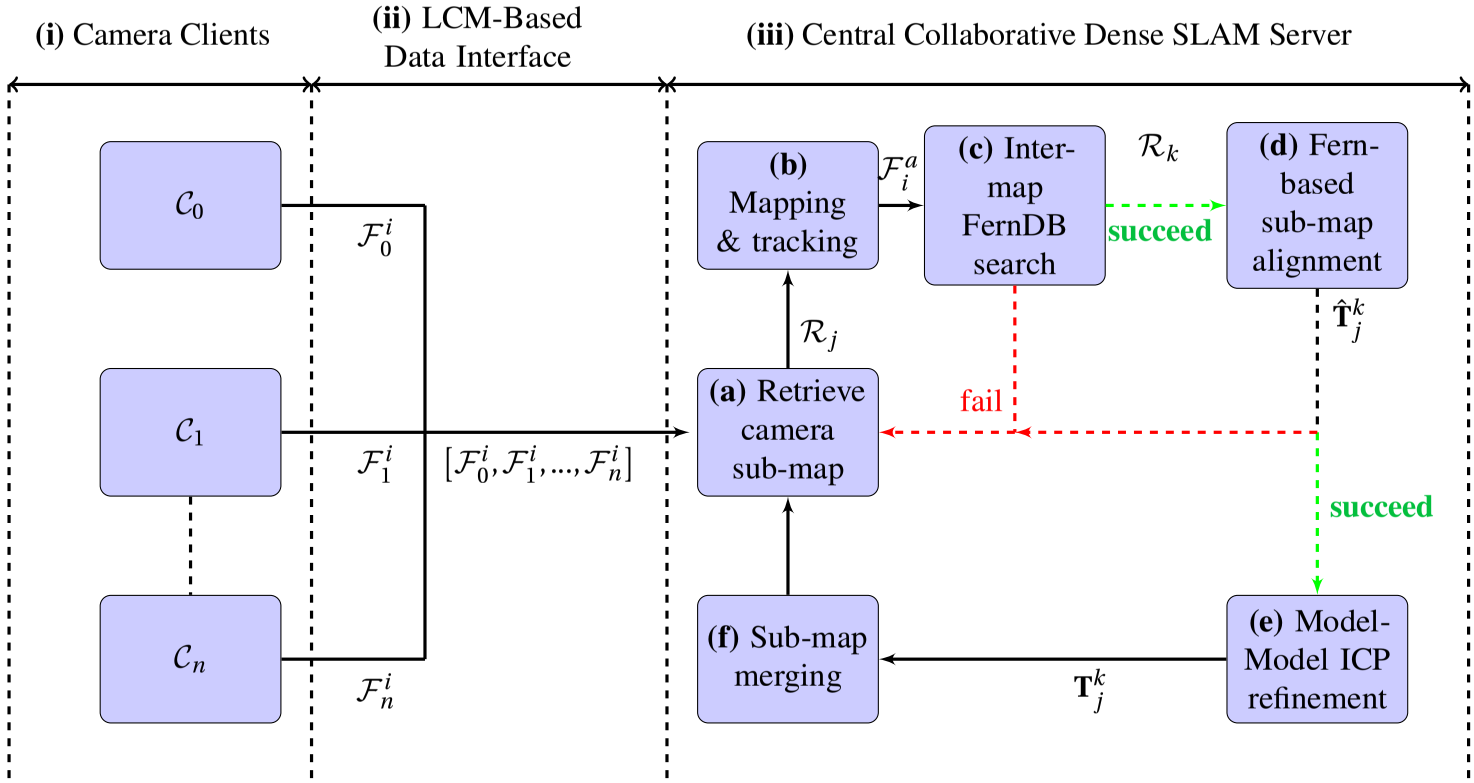



Figure 3.10: **Dense collaborative SLAM system architecture and data processing pipeline.** $(i)$ A number of camera nodes explore a space. $(ii)$ As they explore, cameras send frames over the network via the LCM-based data interface. $(iii)$ Every time-step, the latest frames, one from each camera, are processed sequentially as follows: $(a)$ The reference frame for the camera is retrieved from the reference frame DB. If this is the first frame from the camera, a new key-reference frame pair is initialised in the hash map, otherwise the existing key for the camera is used to retrieve the reference frame $\mathcal{R}_j$ to which it belongs. $(b)$ The camera is tracked, frame data is fused into the sub-map and, if possible, local and global loop closure corrections are applied. $(c)$ Inter-map loop closure candidates are sought. Model predicted active frame $\mathcal{F}_i^a$ is used to search the fern database of the other reference frames. $(d)$ If a match is found (green path), an initial estimate of the transformation aligning $\mathcal{R}_j$ to $\mathcal{R}_k$ is computed, denoted by $\hat{\mathbf{T}}_j^k$. $(e)$ $\hat{\mathbf{T}}_j^k$ is further validated and refined by bringing $\mathcal{F}_i^a$ into tight alignment with the matching sub-map. $(f)$ The two reference frames are merged using the refined transformation $\mathbf{T}_j^k$ and the camera-reference frame hash map is update to reflect the loop closure.

**Algorithm 1** Dense collaborative mapping.

**Input:** a set of $n$ cameras $c = \{\mathcal{C}_i\}$

**Output:** a set of $m$ maps $\{\mathcal{M}_j\}$ where $m \leq n$

HashMap $\mathcal{C}\_to\_\mathcal{R} \leftarrow \emptyset$
**for** $\mathcal{C}_i \in c$ **do** //initialise a new map for each camera.
    $\mathcal{C}\_to\_\mathcal{R}[\mathcal{C}_i] \leftarrow$ new ReferenceFrame()
**end for**
**loop** //main collaborative SLAM loop
    **for** $\langle \mathcal{C}_i, \mathcal{R}_j \rangle \in \mathcal{C}\_to\_\mathcal{R}$ **do**
        track_and_map($\mathcal{C}_i$, $\mathcal{R}_j$) //track camera and update map
    **end for**
    **for** $\mathcal{C}_i \in c$ **do** //search for inter-map loop closures
        Map $\mathcal{M}_j \leftarrow \mathcal{C}\_to\_\mathcal{R}[\mathcal{C}_i].map()$
        Mat4 × 4 $\mathcal{C}_i^j \leftarrow \mathcal{C}_i.currentPose()$
        Frame $\mathcal{F}_i^a \leftarrow$ predictView($\mathcal{C}_i^j$, $\mathcal{M}_j$)
        **for** $\mathcal{R}_k \in \mathcal{C}\_to\_\mathcal{R}$ **do**
            **if** $\mathcal{C}_i \notin \mathcal{R}_k$ **then**
                Mat4×4 $\hat{T}_k^k \leftarrow$ findFern($\mathcal{F}_i^a$, $\mathcal{R}_k$) //compute initial alignment
                Frame ${}_k\mathcal{F}_i^a \leftarrow$ predictView($\hat{T}_j^k$, $\mathcal{R}_k$)
                $T_j^k \leftarrow$rgbdOdometry(${}_k\mathcal{F}_i^a$, $\mathcal{F}_i^a$, $\hat{T}_j^k$) //refine initial alignment
                **if** rgbdOdometry was successful **then**
                    $\mathcal{M}_j \leftarrow$merge($\mathcal{R}_j$, $\mathcal{R}_k$, $T_k^j$) //merge reference frames
                    $\mathcal{C}\_to\_\mathcal{R}[\mathcal{C}_i] \leftarrow \mathcal{R}_j$
                **end if**
            **end if**
        **end for**
    **end for**
**end loop**

## 3.3 Experimental Results

In benchmarking our system we aim to do two things: (i) quantitatively measure the impact of multiple sensors on mapping and tracking performance in terms of both accuracy and processing time; (ii) qualitatively demonstrate the capability of our system. To achieve this we use two standard SLAM benchmark datasets, the ICL-NUIM dataset of Handa et al. (2014) and the RGB-D dataset of Sturm et al. (2012). All experiments were run on a machine equipped with an NVidia GeForce GTX 1080 Ti GPU, 11GB of GDDR5 VRAM, a 4 core Intel i7-7700K CPU running at 4.20GHz and 16GB of DDR4 system memory.

We performed a quantitative analysis of the performance of the system using the living room scene from the ICL-NUIM datasetf (Handa et al., 2014). There are four trajectories through this scene accompanied by both ground truth camera poses for each trajectory and a synthetic ground truth surface model. In our experiments, we used three of the four living room trajectories, $kt_0$, $kt_1$ and $kt_2$. We note that we have not included $kt_3$ in our results due to the fact that $kt_3$ exposes a failure mode of the system. $kt_3$ is one of the more challenging sequences in the ICL-NUIM dataset, as is reflected in the results of both Dai et al. (2017) and Whelan et al. (2016). We found that including $kt_3$ in any of the combinations in our experiments resulted in significantly reduced reconstruction quality.

With the remaining trajectories we looked at the surface reconstruction accuracy and camera trajectory estimation accuracy of the system. To measure the surface reconstruction accuracy we processed an exhaustive set of combinations of the living room trajectories. For each combination we then computed the average per-vertex error in metres using the reference ground truth model. Table 3.1 reports the results of these mapping sessions. To measure the estimated trajectory accuracy we computed the average trajectory error root mean square error (ATE RMSE) in metres between each of the estimated trajectories in each of the combinations and the corresponding reference ground truth trajectories. Table 3.2 reports the results of these experiments. Additionally, we have provided pose estimation accuracy results using the TUM-RGB-D datatset (Sturm et al., 2012). We used the ‘fr3/office’ sequence, as it is a long and challenging sequence. See Figure 6.7

| Sequences | Surface Reconstruction Accuracy |
|---|---|
| $kt_0$ | $0.007m$ |
| $kt_1$ | $0.008m$ |
| $kt_2$ | $0.009m$ |
| $kt_3$ | $0.058m$ |
| $kt_0$ & $kt_1$ | $0.007m$ |
| $kt_0$ & $kt_2$ | $0.009m$ |
| $kt_1$ & $kt_2$ | $0.01m$ |
| $kt_0$ & $kt_1$ & $kt_2$ | $0.009m$ |

Table 3.1: **Dense collaborative SLAM surface reconstruction accuracy on ICL-NUIM dataset.** Each row represents the results of an independent mapping session. The first column in each row gives the specific combination of living room trajectories used in that session. The second column gives the per collaboration surface reconstruction accuracy in metres. The reader will note that in the one camera sessions our results are not exactly as reported in Whelan et al. (2016). This is for a number of reasons; (i) the system is sensitive to the specific GPU of the machine it is deployed on; (ii) as fern encoding is a random process loop closures will not always occur at exactly the same point. In addition, while we have made an effort to tune EF's parameters appropriately we have intentionally tried to avoid overfitting, albeit at the cost of decreased performance in certain sequences. Indeed, the approach of parameter tuning becomes impractical as the number of parameters that must be tuned grows in proportion with the number of collaborators in a session.

and its caption for details of how we generated collaborative mapping sessions with this dataset. The models shown in Figure 3.12 demonstrate the quality of the maps produced by our system. For a visualisation of a dense collaborative mapping session please see the accompanying video (https://youtu.be/GUtHrKEM85M).

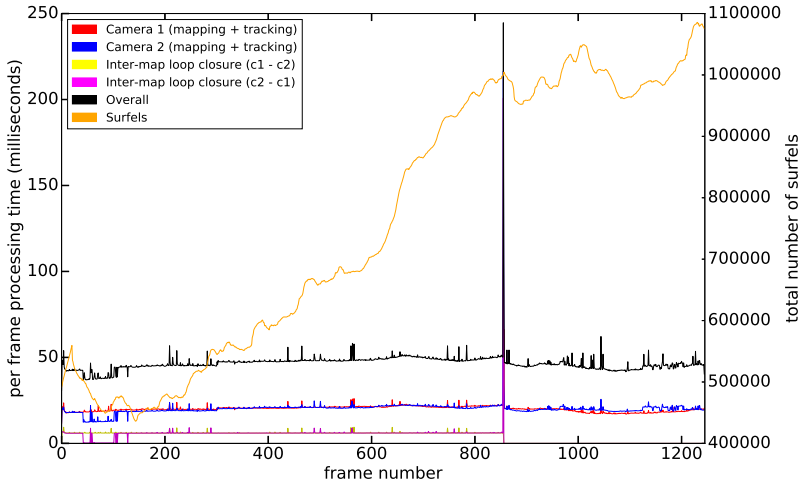
These quantitative and qualitative results demonstrate that our system is capable of dense mapping and tracking with up to 3 cameras with accuracy that is on par with the original single-sensor ElasticFusion algorithm. Surface reconstruction accuracy (Table 3.1) shows no significant variation between one, two and three camera sessions. While trajectory estimation accuracy shows some minor variation between one, two and three camera sessions on the synthetic ICL-NUIM dataset, this variation is not reflected by the real-world TUM-RGBD sequence, where no significant variation in accuracy has occurred (Table 3.2).

Next, we again used the 'fr3/office' sequence, this time to measure the processing time of the different components of the system during a 2 camera and then a 3 camera collaborative session. The results of this experiment are shown in Figure 3.11. From

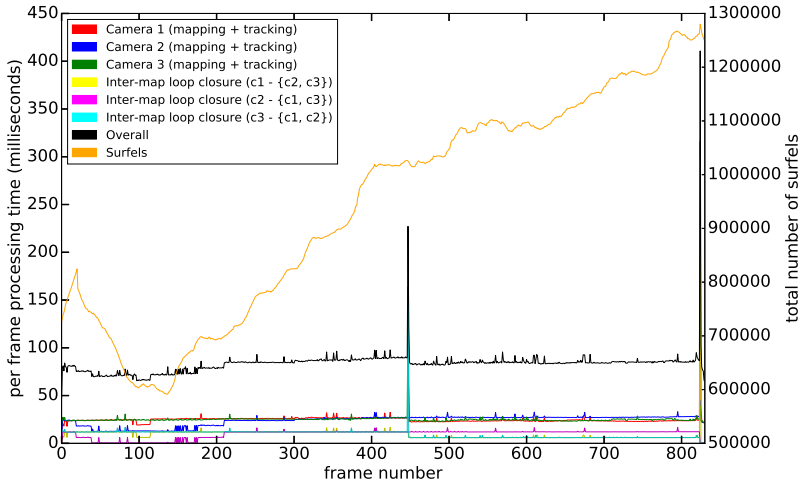
| Sequences | Individual Trajectory ATE RMSE |
|---|---|
| **ICL-NUIM** | |
| $kt_0$ | $0.0143m$ |
| $kt_1$ | $0.0127m$ |
| $kt_2$ | $0.0191m$ |
| $kt_3$ | $0.2219m$ |
| $kt_0$ & $kt_1$ | $0.0147m$ \| $0.5950m$ |
| $kt_0$ & $kt_2$ | $0.1112m$ \| $0.0191m$ |
| $kt_1$ & $kt_2$ | $0.0124m$ \| $0.7212m$ |
| $kt_0$ & $kt_1$ & $kt_2$ | $0.0340m$\|$0.5954m$\|$0.0043m$ |
| **TUM-RGB-D** | |
| $office_0$ | $0.018m$ |
| $office_0$ & $office_1$ | $0.019m$ \| $0.015m$ |
| $office_0$ & $office_1$ & $office_2$ | $0.014m$ \| $0.015m$ \| $0.014m$ |

Table 3.2: **Dense collaborative SLAM trajectory estimation accuracy on ICL-NUIM and TUM-RGB-D datasets.** Similar to Table 3.1, each row represents the results of an independent mapping session. The first column in each row gives the specific combination of trajectories used in that session. The second column gives the per collaboration absolute trajectory error (ATE).

these timings we can see that for 2 and 3 camera sessions the system maintains efficient performance, operating just outside hard real-time rates of $30hz$ (i.e. $\sim 33ms$ per frame) for the TUM-RGBD dataset (Sturm et al., 2012). However, we can also see that adding more cameras to the SLAM session leads to a corresponding decrease in frame-rate. This highlights what is not only a bottleneck in our system, but an open area of research in collaborative dense SLAM in general: in addition to scaling in proportion to the size of the space being mapped, and the resolution of the sensor being used, a multi-camera system must also scale in proportion to number of cameras that are mapping.

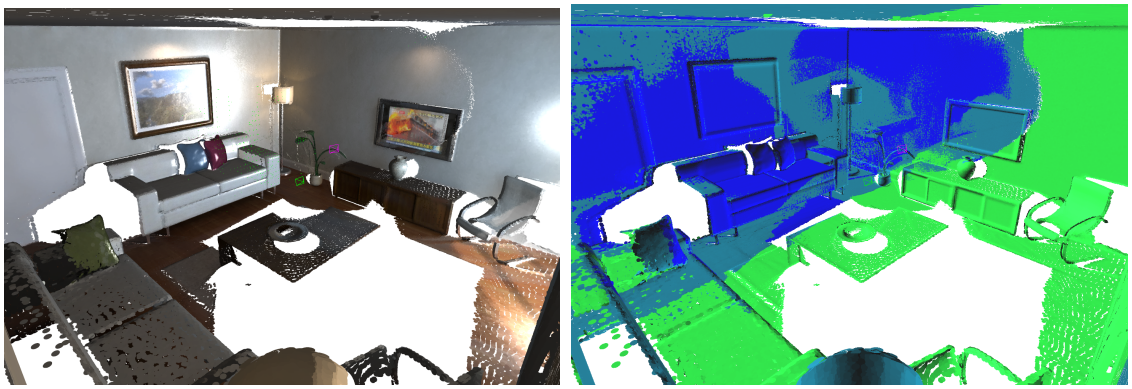



(a)

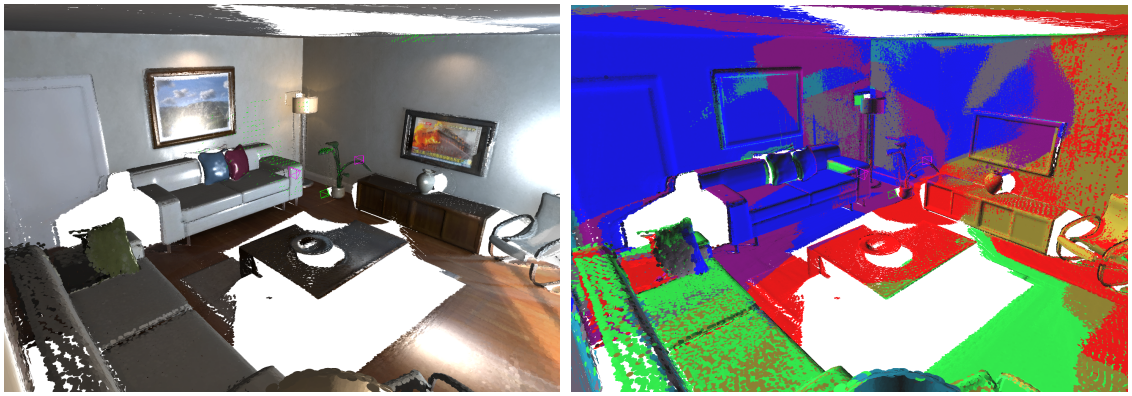



(b)

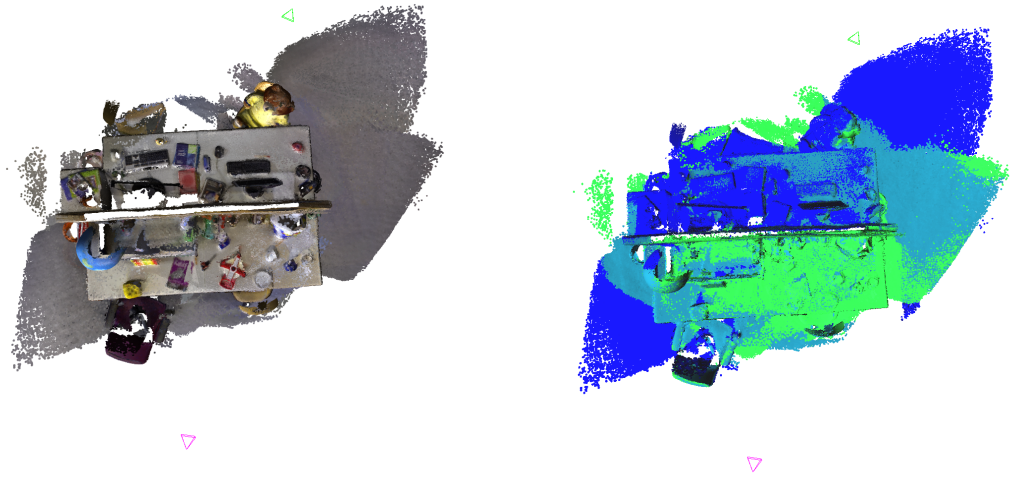
Figure 3.11: **System timings for (a) a 2 camera and (b) a 3 camera collaborative mapping session with the 'fr3/office' sequence.** The original sequence contained 2487 frames. For graph (a) we split these frames into two subsequences of equal length, similarly, for graph (b) we split them into three subsequences of equal length. The subsequences were then processed collaboratively. For example, in (a) Camera 1 corresponds to the first half of the frames from the original sequence while Camera 2 corresponds to the second half of the frames. The spike at frame 942 in graph (a) is caused by an inter-map loop closure between the two cameras. Similarly, the two spikes in graph (b) at frames 449 and 826 are both caused by inter-map loop closures.



(*a*)



(*b*)



(*c*)

Figure 3.12: **Examples of maps produced by the system.** The left column shows the final models produced by the system. The right column shows the contributions of each camera to the final model. (*a*) and (*b*) correspond to the ICL-NUIM $kt_0$ & $Kt_2$ and $kt_0$ & $kt_1$ & $kt_2$ sessions respectively. (*c*) corresponds to the 2 camera session using the TUM-RGBD dataset.

## 3.4 Summary

This chapter has described our system for collaborative dense SLAM that allows multiple independently moving RGB-D cameras to collaboratively build a shared map of space and simultaneously track against that shared map. Each camera is initialised in its own reference frame where it begins building its own local map and keyframe database. Inter-map loop closure candidates are found by matching a camera's current view of its local map to keyframes in the databases of the other cameras. Once a loop closure candidate has been identified, the two triggering submaps are aligned and merged together. In contrast to related multi-camera systems, our system is centralised and allows all cameras to perform full SLAM operation directly on the same model - once inter-map loop closures have been corrected.

The system described in this chapter is fundamentally limited to operating with slow hand-held camera motion and in indoors environments. These limitations are inherited from ElasticFusion, a single-sensor dense visual SLAM system, that acted as the foundation upon which we built our collaborative SLAM system. In Chapters 5 and 6 we describe systems that also build on ElasticFusion, but with the aim of permitting dense visual SLAM in automotive scenarios, and with pinhole and fisheye monocular cameras.

One of the main bottlenecks of the system, as indicated by our experiments, is the number of cameras in the collaborative mapping session. Going from a two camera session to a three camera session results in a drop in the frame-rate of the system from approximately $25ms$/frame to approximately $12.5ms$/frame (see Figure 3.11). Motivated by the observation that ElasticFusion treats every frame with equal importance, and therefore processes every frame in full, in Chapter 4 we consider the original single sensor ElasticFusion algorithm and present a novel information theoretic approach to keyframing with the goal of improving the system's computational efficiency. Our approach is to implement a mechanism that selects only those keyframes that carry novel information for updating and extending the map. Conversely, this allows the system to discard frames with redundant information, thereby avoiding unnecessary computation.

# CHAPTER 4

# Information Theoretic Keyframe Selection

Chapter 3 described our approach to collaborative dense SLAM, which built on ElasticFusion, a single sensor dense SLAM system (Whelan et al., 2015). Our experimental results (Section 3.3) indicated that the number of collaborating sensors participating in the mapping session represents a bottleneck of the system. In the context of centralised collaborative dense SLAM, where the central mapping server has finite computational resources, standard techniques for increasing a system's throughput such as parallelisation, while potentially helpful (e.g., by allowing the tracking of each camera to proceed in parallel), would not address a more fundamental bottleneck in the ElasticFusion algorithm itself; that as the number of surfels in the map increases, so too does the system's per-frame processing time.

In this chapter, we focus on this bottleneck within the ElasticFusion architecture. Specifically, we introduce a novel information theoretic approach to keyframing, that allows for selectively switching between performing full SLAM and a tracking only mode within ElasticFusion, based on how accurately the dense map models live camera data. Keyframing has proved to be an effective method for curbing growth of computational complexity in both sparse and dense SLAM, however previous dense keyframing methods do not make full use of the dense map when selecting keyframes. Using our approach to keyframing, ElasticFusion can track how well the map models live camera imagery

by measuring the normalised information distance (NID) between live frame data and a map prediction based on rendering map data into a virtual camera placed at the current pose estimate. We then adapt ElasticFusion such that when a region of the environment is well modelled by the current map, the system does not fuse live camera data into the map, avoiding computationally costly and unnecessary map updates. However, when the map's prediction accuracy deteriorates, either due to the camera exploring unseen regions of the scene or due to structural changes in the scene, the map is updated with live frame data, ensuring it remains up-to-date and accurate. Our approach exposes a tunable parameter that allows a user to balance the trade-off between the completeness of the map and the computational cost of maintaining a more dense map in a principled manner.

In Section 4.5, we provide both quantitative and qualitative analyses of our approach to keyframing again using the ICL-NUIM and Freiburg datasets. In these experiments, we focus on keyframing in the context single-session dense SLAM with ElasticFusion as it is conceptually simpler, but we note that the conclusions also apply in the context multi-camera collaborative mapping with the system described in Chapter 3. In collaborative SLAM, the effects of this bottleneck are amplified by the shared map, which, often times, will be more extensive than if each robot were to build its own individual map.

The work in this chapter was published in

- L. Gallagher and J. McDonald. Efficient Surfel Fusion Using Normalised Information Distance. In *Computer Vision and Pattern Recognition (CVPR) workshop on 3D Scene Understanding for Vision, Graphics, and Robotics*, 2019

## 4.1 Background

Dense reconstruction algorithms take advantage of multiple overlapping surface measurements to estimate a denoised surface model. BundleFusion (Dai et al., 2017), ElasticFusion (Whelan et al., 2016), Kintinuous (Whelan et al., 2014a) and KinectFusion (Newcombe et al., 2011a) are all exemplars of this class of algorithms. Common to each of these algorithms is a focus on maximising the accuracy of both the reconstructed models

and estimated 6-DOF motion of the camera without considering the relationship between the cost of processing and its impact on model quality. In this section we propose a general information theoretic approach to modelling this relationship and demonstrate its potential in significantly reducing the processing requirements of dense surfel fusion with limited impact on accuracy.

Although such considerations are not typically required in the context of a high-end GPU processing the output from a single camera, they become critical when applying these algorithms in both collaborative settings and in compute limited scenarios. For example, the work in Section 3.2.3 on the development of collaborative dense SLAM showed that increasing the number of sensors in the fusion process rapidly leads to a degradation in system performance, which drops from real-time processing rates of more than $25hz$ for 1 and 2 camera sessions to interactive rates of just $12hz$ for 3 camera sessions (see Figure 3.11). In compute limited scenarios common in much of robotics and augmented reality, additional costs such as battery usage and heat dissipation highlight the need for more efficient approaches.

Since the seminal KinectFusion algorithm was introduced there has been a profusion of methods proposed that use scalable data structures, such as hierarchical trees and hash tables, for efficiently storing the underlying TSDF volume (Niessner et al., 2013; Vespa et al., 2018). Geometric simplification methods that reduce the number of parameters used to represent the map, such as incremental meshing have also been proposed (Whelan et al., 2014b). These methods, however, must maintain complexity around surface discontinuities and regions of high frequency detail and require intermediate map representations from which they infer simpler models.

Methods that leverage efficient algorithms for camera tracking and fusion, compact map representations and geometric simplifications for optimising the whole SLAM pipeline have also been proposed (Kahler et al., 2015; Whelan et al., 2014a).

In comparison, relatively little attention has been paid to increasing the efficiency of surfel-based SLAM methods. In model-predictive frameworks, like ElasticFusion, as map size and complexity increase so too does the cost of updating the map with the current

frame. In the presence of sensor noise and drift in camera tracking simply fusing every frame can lead to spurious surfels and regions of the surface being over-represented by the map. Though the above approaches could potentially be helpful in improving the efficiency of surfel fusion it is the purpose of the work in this chapter to explore a mode of keyframing that is based on quantifying the novelty of the information in incoming frames. The method we propose more optimally balances the accuracy of dense surfel models against the computational cost of dense surfel fusion. To demonstrate the effectiveness of the method we place it in the pipeline of the ElasticFusion dense mapping system (Whelan et al., 2015) and report empirical evidence that this leads to good precision and scalability characteristics in comparison to the original algorithm. We report results of the technique's scalability and the accuracy of the resultant maps by applying it to both the ICL-NUIM (Handa et al., 2014) and TUM RGB-D (Sturm et al., 2012) datasets. These results demonstrate the capabilities of the approach in performing accurate surface reconstructions whilst utilising a fraction of the frames when compared to the original algorithm.

## 4.2 Normalised Information Distance

In this section we recount concepts from the field of information theory that are relevant to the discussion.The information content of the outcome of a random event $x$ is given by

$$h(x) = log_2 \frac{1}{P(x)} \tag{4.1}$$

$h(x)$ is a measure of how surprising the outcome was. Here information is given in units of *bits*, however, it can alternatively be given in other units, such as *nats* if natural $log$ is used instead of $log_2$. The *entropy* of a random variable $X$ is the average information content of an outcome

$$H(X) = \sum_{x \in A_X} P(x) \, log_2 \frac{1}{P(x)} \tag{4.2}$$

the *joint entropy* of two random variables $X$ and $Y$ is defined as

$$H(X,Y) = \sum_{x,y} P(x,y)\, log_2 P(x,y) \tag{4.3}$$

The joint entropy of $X$ and $Y$ is less than or equal to the sum of $X$ and $Y$'s individual entropies

$$H(X,Y) \leq H(X) + H(Y) \tag{4.4}$$

with equality iff $X$ and $Y$ are independent. Using these quantities we can specify the *mutual information* between two random variables $X$ and $Y$

$$I[X;Y] = H(X) + H(Y) - H(X,Y) \tag{4.5}$$

There are many ways of describing the nature of mutual information in a way that appeals to our intuitions. For instance, mutual information is a measure of the average reduction in uncertainty about $x$ that results from learning $y$ (Bishop, 2007). It can also be seen as a measure of the degree to which two random variables are independent of one another.

Perhaps, from the point of view of optimal model estimation, a more insightful way of thinking about mutual information is in terms of the Kullback-Leibler divergence of two probability distributions. Namely, if we have $P(X|\theta)$, a probability distribution conditioned on some parameter set $\theta$ (i.e the state estimated by the SLAM system), and $P(X,Y|\theta)$ the joint distribution of two random variables $X$ and $Y$ (which one can roughly think of as live and model predicted camera frames respectively) then the mutual information between $X$ and $Y$ is given by the divergence between their joint distribution and the product of their marginals, where $P(X,Y) = P(X)\,P(Y) \implies X \perp Y$ (Bishop, 2007)

$$I[X;Y] = KL(P(X|\theta,Y)\,||P(X|\theta)\,P(Y)) \tag{4.6}$$

The *Normalised Information Distance* (NID) between two random variables $X$ and $Y$ is defined as follows

$$NID(X,Y) = \frac{H(X,Y) - I[X;Y]}{H(X,Y)} \tag{4.7}$$

NID is a measure of the *relative similarity* of two random variables. In this work we propose using the NID between a live camera frame and a model predicted camera frame as a basis for deciding whether the current frame should be fused into the reconstruction.

*Normalised information distance* (NID) is a true metric over information space in the sense that it satisfies the four conditions of a metric (non-negativity, identity, symmetry and triangle inequality, Vinh et al., 2010). Information theoretic quantities such as *mutual information* (MI) and *variation of information* (VI) have been used to solve problems in computer vision involving heterogeneous sensor modalities, such as aligning medical imagery (Studholme et al., 1999). The benefit of such measures is that they are functions of the joint probabilities of the two signals rather than of the signals themselves. Both quantities, however, are problematic from a state estimation perspective. MI is not a true metric as it does not satisfy the triangle inequality condition. VI, on the other hand, is a true metric, but it exhibits an unfortunate ambiguity; if one were to decrease the joint entropy between two random variables and correspondingly decrease the cardinality of one of them one could maintain the same VI between the two random variables. The NID solves both these problems; it is a true metric, and it is not ambiguous.

NID has been used in the context of SLAM before. Stewart and Newman (2012) use NID to localise a moving camera against an *a priori* known $3D$ map. The geometry in view of the camera is predicted. The pose of the camera is then estimated by minimising the NID between the live camera frame and the camera frame predicted from the prior model (Stewart and Newman, 2012). In the same line of work, Pascoe et al. (2017) propose to use NID directly for camera odometry in a monocular SLAM system. NID provides a level of robustness to lighting, weather and structural changes in the scene not possible with standard ICP-based tracking algorithms. After long periods of neglect variations in the scene can make it impossible to update keyframes directly. The authors again use NID. This time the keyframe update is computed such that it minimises the NID between the reference frame and the current frame (Pascoe et al., 2017).

Our approach is inspired by two lines of research that use information theoretic measures extensively; *Active Vision* (Chli and Davison, 2008; Davison, 2005) and *Normalised*

*Information Distance-based SLAM*. In active vision the mutual information between image features is used to efficiently guide the search for correspondences. As features are matched the uncertainty in the location of other correlated features collapses, thus reducing the remaining search space.

*Active Vision* attempts to turn the problem of feature matching on its head. Traditional solutions to this problem follow a 'bottom up' process of extracting features from image intensities, then attempting to match descriptors across images. In contrast, the active vision approach is 'top down'. The process starts with matching and not with feature extraction, using the correlations between expected feature locations in the image to determine what features to search for next (Davison, 2005). What these works give us is a precedent for making decisions on what computations to perform based on information theoretic principles such as mutual information.

In *Normalised Information Distance-based SLAM* the normalised information distance between a model predicted reference image and a live camera image is used to construct an optimisable objective function for camera tracking and measurement fusion, that is robust to drastic changes in scene illumination (Pascoe et al., 2015, 2017; Stewart and Newman, 2012).What these works give us are the computational mechanics for calculating the normalised information distance between two frames in a *direct* and *dense* setting.

## 4.3 Reducing Redundancy in Dense SLAM

In model-predictive camera tracking, once the current pose of the camera, $\mathbf{P}_t \in SE(3)$, has been estimated it is used to render a synthetic view of the map, $\mathcal{F}_t^p$. For each frame, we compute the $NID$ between the appearance distribution of $\mathcal{F}_t^p$ and that of the live frame, $\mathcal{F}_t^l$,

$$NID\left(\mathcal{F}_t^p, \mathcal{F}_t^l\right) = \frac{H\left(\mathcal{F}_t^p, \mathcal{F}_t^l\right) - I\left[\mathcal{F}_t^p; \mathcal{F}_t^l\right]}{H\left(\mathcal{F}_t^p, \mathcal{F}_t^l\right)} \tag{4.8}$$

where, $H(X, Y)$ and $I[X; Y]$ are the joint entropy and mutual information, respectively, of two random variables $X$ and $Y$. When the NID is high this implies that the live camera frame is not well explained by the current model estimate, and hence the system should

fuse that frame.

For each frame we compute two NID scores, one for the RGB component of the frame, $NID_{rgb}$, and one for the depth component, $NID_d$. The final NID is the weighted sum

$$NID\left(\mathcal{F}_t^l, \mathcal{F}_t^p\right) = (1 - \lambda_d)\, NID_{rgb}\left(\mathcal{I}_t^l, \mathcal{I}_t^p\right) + \lambda_d NID_d\left(\mathcal{D}_t^l, \mathcal{D}_t^p\right) \tag{4.9}$$

where, $\lambda_d \in [0..1]$ denotes the relative weighting between the RGB and depth components of the NID, $\mathcal{I}$ denotes the intensity image derived from the colour channels of the frame, and $\mathcal{D}$ denotes the depth component.

It is, in general, quite difficult to achieve a low NID score in either photometric or depth space for a number of reasons. Firstly, we are comparing live intensities to those produced via splatted rendering, which is only a rough approximation of the image formation process. Live camera images include other effects such as white balancing, barrel distortion and rolling shutter. Even if all these effects could be modelled by the SLAM system, the integration of light onto the camera sensor plane during an exposure is an inherently random process. The depth measurements are also corrupted by noise introduced by the finite resolution of the camera, its motion and the influence of interfering natural light.

Recall Equation (2.11), the equation for surface fusion. In our system the fusion process is governed by the following equation

$$\mathcal{M}_{t+1} = \begin{cases} W\mathcal{M}_t^l + \mathcal{M}_t & \text{if } NID\left(\mathcal{F}_t^l, \mathcal{F}_t^p\right) > \tau \\ \mathcal{M}_t & \text{otherwise} \end{cases} \tag{4.10}$$

the map at time $t+1$, $\mathcal{M}_{t+1}$, is the fusion of the map at time $t$, $\mathcal{M}_t$, and the map estimate contained within the current live frame $\mathcal{F}_t^l$, $\mathcal{M}_t^l$, if the normalised information distance exceeds a threshold $\tau \in [0..1]$, otherwise the frame is skipped and the map is simply carried over. To emphasise this point; we fuse only those frames that are sufficiently *dissimilar* to our current model predictions. Figure 4.2 shows how the parameter $\tau$ effects the completeness of the final model.

To compute both NID scores we follow the approach outlined in Stewart and Newman (2012), except we drop the cubic spline weighting as we do not require a differentiable histogram. We compute a joint appearance intensity histogram and a joint $2.5D$ depth histogram. Each of the bins in the histograms corresponds to a range of values in $\mathcal{F}_t^l$ co-ocurring with a range of values in $\mathcal{F}_t^p$. The histogram $\mathcal{P}$ is computed in a SIMD fashion on the GPU using CUDA atomic instructions then the marginal entropies and, ultimately, the NID is derived on the CPU by marginalising over the histogram's rows and columns.

## 4.4 Accounting for Time

The map in our system is divided into two subsets; $\theta_t$, containing all active surfels that have been fused within a window of time $\delta_t$, and $\phi_t$ containing all the inactive surfels. Surfels in $\theta_t$ are used for tracking and fusion. As the camera loops back around into old regions of the map local loops are closed between $\theta_t$ and $\phi_t$ and the surfels in $\phi_t$ that are viewed from the triggering frame are reactivated.

To accommodate long sequences of frames without fusion it is important that we adjust $\delta_t$ in our system so that $\theta_t$ grows and shrinks, and local loop closures are triggered in much the same way as in the original EF algorithm. This is achieved by letting $\delta_t = \delta_t + \delta_n$ where $\delta_n$ is the number of frames since the last frame that was fused. Additionally, if a loop closure (local or global) is triggered then we proceed with fusion for that frame irrespective of the NID.

We take the current active frame, used for camera tracking and current inactive frame, used for closing local loops, as our reference frame for NID calculation. The depth maps for both frames are used to determine which values are visible from the current viewpoint. Doing so allows us to avoid a costly re-rendering of a single joint active/inactive frame, thereby making NID calculation a constant-time cost.

## 4.5 Experiments

In this section we provide experimental results of the application of the system to a number of datasets. Though the presented keyframing method is equally applicable in both collaborative and single-camera mapping, here we focus on the latter. The addition of multiple cameras adds significant complexity in the form of additional input streams, inter-map loop closures and multi-camera map updates. While analysing the behaviour of NID keyframing with these additional factors intact presents a valid experimental question, their presence does not help us quantify the behaviour of NID keyframing and its effect on the core mapping and tracking algorithm.

Figure 4.2 provides a qualitative example of how the NID threshold $\tau$ effects model completeness. The figure shows how setting $\tau$ to progressively higher values leads to increasingly sparser reconstructions. However, it is important to note that this sparsity is induced in a principled manner, by only discarding redundant frames, and can be easily tuned, using $\tau$, to trade-off map density for computational efficiency on a per application basis. Figure 4.1 shows different points during the reconstruction of the Freiburg 3 office sequence (Sturm et al., 2012). Here, the effectiveness of NID keyframing in striking a balance between reducing the number of redundant frames fused and maintaining an accurate and complete reconstruction can be seen on a real-world sequence.

To provide a quantitative evaluation of the system's performance we use the ICL-NUIM synthetic living room dataset (Handa et al., 2014) where we measure the impact of varying both the NID threshold $\tau$ and the relative weighting factor, $\lambda_d$. Specifically, for each setting of $\tau$ and $\lambda_d$ we processed each of the four living room sequences from the dataset and recorded a number of metrics, including: the average $NID$ score for the session, the number of frames fused during the session, the final number of surfels in the map, the ATE (as per Section 3.3), and the surface reconstruction accuracy (as per Section 3.3). The results are displayed in Figures 4.3 and 4.4. These results allow direct comparison to the original EF algorithm and show that our system can maintain comparable accuracy whilst using significantly fewer frames and producing significantly fewer surfels in the

final model. We can see that for $\tau \in [0.7, 0.9]$ there is a significant reduction in the number of frames being fused and the number surfels in the final map across all sequences. At the same time, there is no significant change in system accuracy for $Kt_0$, $Kt_1$ and $Kt_2$. While there is some decrease in accuracy for $Kt_3$, for some values of $\lambda_d$, e.g., $\lambda_d = 0.5$, the decrease in accuracy is not significant.

From the plots in Figures 4.3 and 4.4, we can see that weighting the $NID$ score towards either $NID_{rgb}$ (i.e. $\lambda_d = 0$ and $\lambda_d = 0.25$) or $NID_d$ (i.e. $\lambda_d = 1$ and $\lambda_d = 0.75$) leads to worse tracking and surface reconstruction accuracy than taking the average (i.e. $\lambda_d = 0.5$). Determining why this is the case will require further investigation, however, one possible explanation for this effect is the potentially different rates at which the photometric and geometric properties of surfels in the map converge to accurate values. In fact, from the plots, we can see that by weighting the $NID$ towards $NID_d$ leads, in general, towards fewer frames being fused and fewer surfels in the final maps than weighting towards $NID_{rgb}$ does. Note also how higher values of $\lambda_d$ result in lower average $NID$ scores. Together, these two trends indicate that, on average, geometric predictions are rendered more faithfully by ElasticFusion, using less input frames, than photometric predictions.

In general, whether a specific surfel is accurately reconstructed is a function of the scene's geometry, lighting, texture, the specific trajectory taken by the sensor, as well as its resolution and noise characteristics. For example, a relatively textureless but geometrically variable surface might require only a few frames, taken from oblique viewing angles, in order for photometrically accurate renderings of the surface to be made, but many frames, taken from a variety of view points, in order to accurately capture the surface's geometry. In this example, setting $\lambda_d$ to a low value (e.g., 0.25), could lead to many frames being discarded that carry a large amount of novel information about the scene's geometry, but relatively little novel information about the scene's colour, resulting in poor geometry estimates. Therefore, taking the average of $NID_d$ and $NID_{rgb}$ would balance the need for a reconstruction that is both geometrically and photometrically accurate more optimally than simply focusing on either geometric or photometric properties of the scene. And this is indeed what we see in the surface reconstruction accuracy and ATE plots in Figures 4.3

and 4.4.

Finally, to measure the scalability of our approach we used the Dyson lab sequence (Whelan et al., 2016) which consists of over 6500 frames captured during a scan of an office environment. The sequence contains a mix of segments where the camera explores new regions and where the camera closes local loops and re-visits already scanned regions of the office. These results are shown in Figure 4.5. Note that in this plot the NID fusion time includes the time for computing the NID. Furthermore, whilst the EF fusion time increases steadily over the log, the rate of increase for the fusion component of the NID fusion approach is negligible. This is both due to the NID based frame selection, and the consequent reduction in surfels. It can be seen that the rates of growth are very different.

## 4.6 Summary

In this chapter we presented a novel approach to keyframe selection, an important feature of any SLAM system, that allows cameras to selectively switch to a tracking only mode when in well mapped regions of the model. We integrated our keyframe selection method into ElasticFusion, where it leverages the dense map to decide which incoming frames contain novel information about the scene, allowing the system to avoid processing overly redundant frames. Our experiments showed that our approach to keyframe selection allows the system to reconstruct accurate dense maps with a fraction of the frames, when compared to the original ElasticFusion algorithm.

By requiring that frames exceed a minimum threshold in terms of the quantity of novel information they contain before they can be fused into the map, the presented keyframe selection method can be used to control the completeness of ElasticFusion's surfel map. Our experiments showed that increasing the minimum information threshold resulted in fewer frames being fused and, ultimately, fewer surfels in the final map. At the same time, map accuracy, when compared to the original EF algorithm, remained largely unaffected. Our experiments also showed that, over longer mapping sessions, our keyframe selection method can lead to shorter frame processing times. This effect can be attributed to the slower growth in map size resulting in slower growth in frame prediction time, one

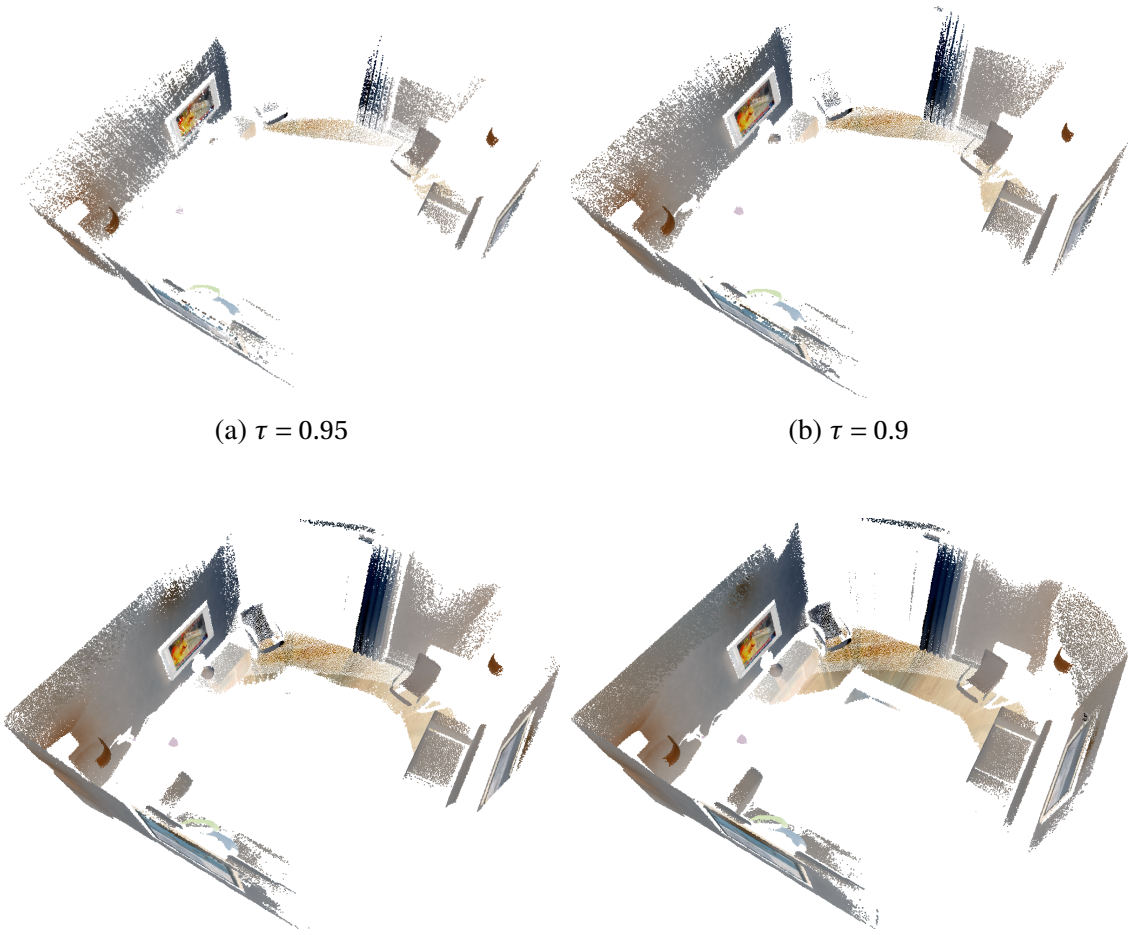
of EF's main bottlenecks.

As discussed in Section 1.2, it is the goal of this thesis to address certain key limitations of dense visual SLAM so that dense systems and methods can be applied in a wider range of less constrained scenarios. In this chapter, and in Chapter 3, our discussion has focused on extensions to ElasticFusion, a dense visual SLAM system built for mapping indoor spaces with a single slow moving hand-held, RGBD camera. First, in Chapter 3, we addressed ElasticFusion's single sensor limitation, opening the door to multi-camera collaborative dense SLAM applications. Our experiments showed that one of the main bottlenecks in collaborative dense SLAM is the number of cameras in the session. This prompted an investigation into the keyframe selection method that was the focus of the current chapter. In the next chapter, we describe the design and implementation of a hybrid SLAM system that combines elements of sparse and dense visual SLAM with the goal of moving beyond indoor mapping with slow moving hand-held RGBD cameras. In particular, in Chapter 5, we focus on the case of dense mapping in outdoor automotive scenarios with monocular cameras as a representative example of a less constrained environment. Following this, Chapter 6 describes how we integrated a wide-angle fisheye camera model, more suited to applications where safety is critical than the standard pinhole camera model used initially by the hybrid system.

(a) (b)

(c)

Figure 4.1: A hand-held camera explores a scene taken from the TUM RGB-D dataset (Sturm et al., 2012). Green frustums represent the pose of the camera for fused frames. (a) Initially the NID between frames predicted from the incomplete model and live frames is quite high, so all frames are fused. (b) The camera continues exploring but selectively decides not to fuse frames that the NID deems are well explained by the model. (c) The camera closes a loop and transitions fluidly to purely tracking against reactivated portions of the model as there is now no need for fusion.



(a) $\tau = 0.95$ (b) $\tau = 0.9$

(c) $\tau = 0.8$ (d) $\tau = 0.7$

Figure 4.2: **NID qualitative results for $\mathbf{Kt_1}$ sequence from ICL-NUIM (Handa et al., 2014)**. For these experiments we set $\lambda_d = 1 - \alpha = 0.75$. A restrictive setting of $\tau$ (a) means only a few highly informative frames are fused into the model, leading to an accurate but less complete map. (b) - (d) by steadily decreasing $\tau$ more frames are fused into the model leading to more complete models but at a higher computational cost.

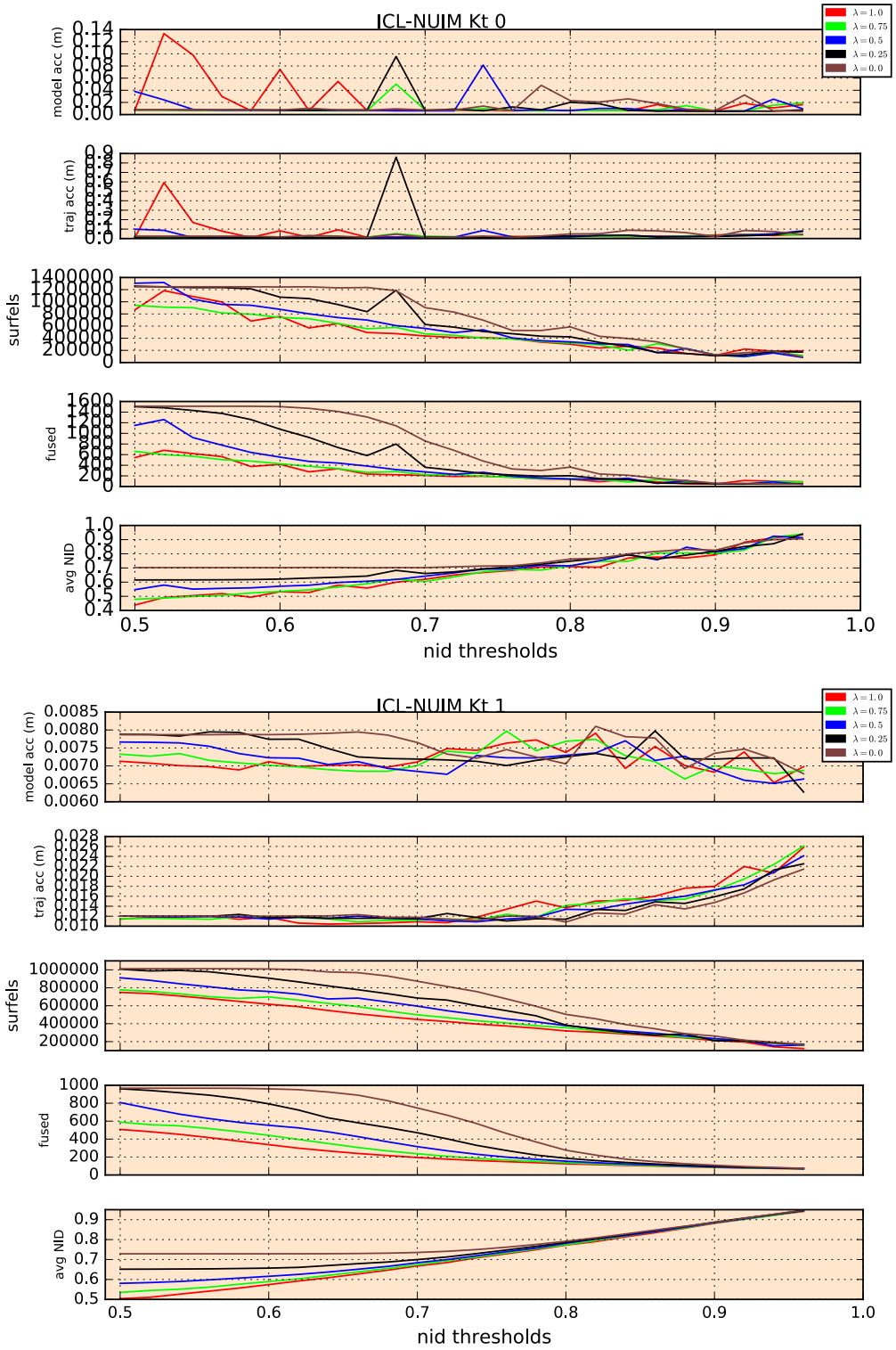



Figure 4.3: **NID keyframing results for ICL-NUIM sequences $\mathbf{Kt_0}$ and $\mathbf{Kt_1}$.** We measured the model accuracy, trajectory accuracy, number of surfels in the final model, number of frames fused and the average NID score for each sequence as we varied the NID threshold $\tau$ and the relative weighting $\lambda_d$ between $NID_d$ and $NID_{rgb}$.

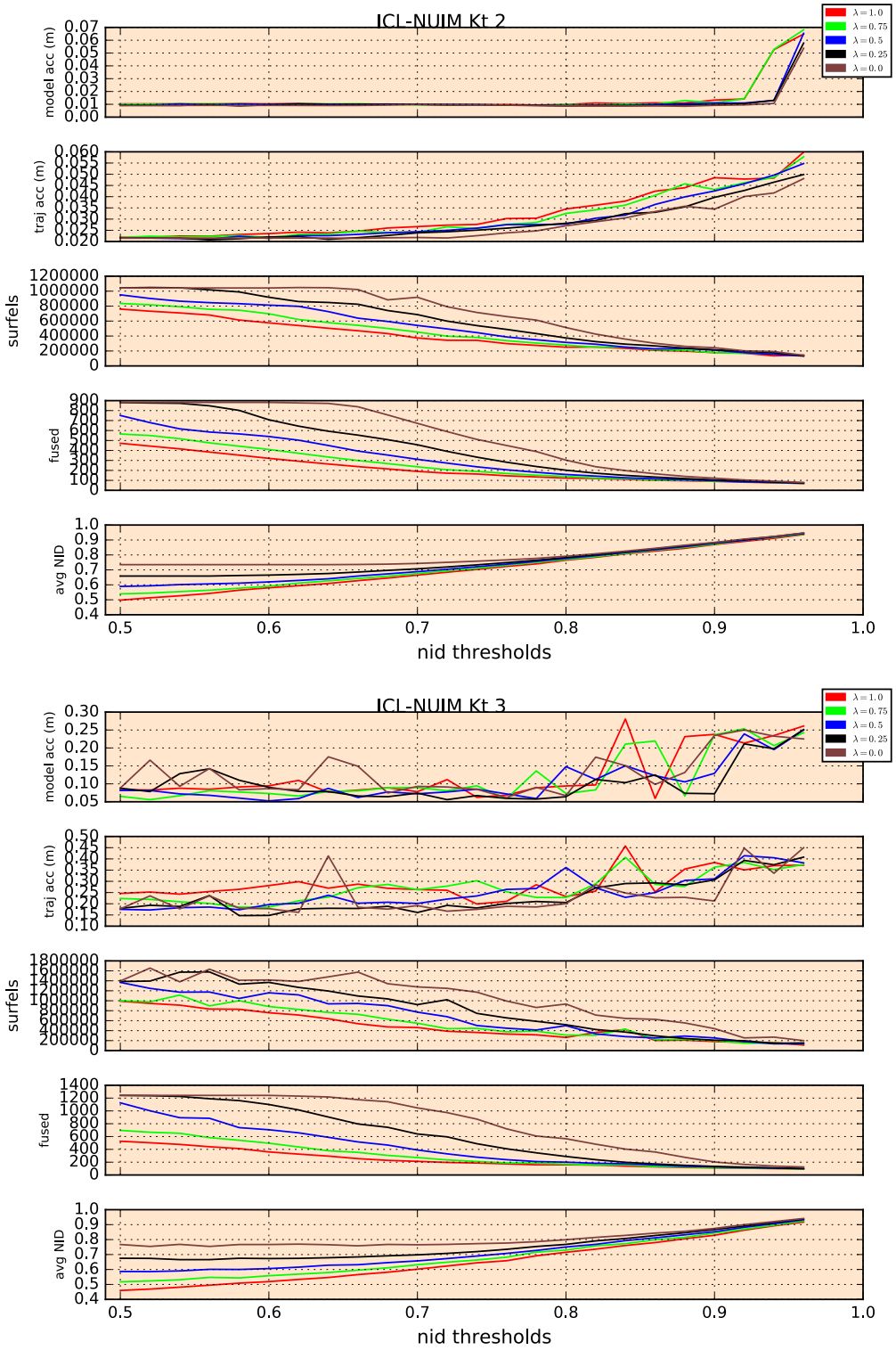
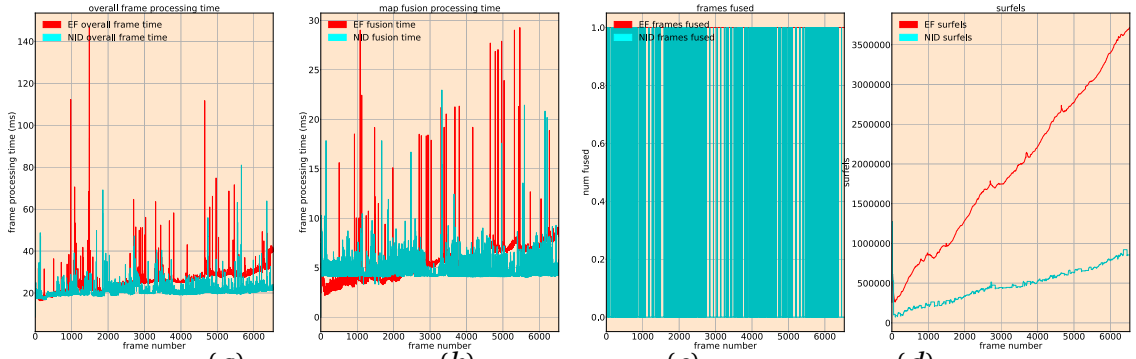



Figure 4.4: **NID keyframing results for ICL-NUIM sequences $\mathbf{Kt_2}$ $\mathbf{Kt_3}$**. We measured the model accuracy, trajectory accuracy, number of surfels in the final model, number of frames fused and the average NID score for each sequence as we varied the NID threshold $\tau$ and the relative weighting $\lambda_d$ between $NID_d$ and $NID_{rgb}$.




(*a*) (*b*) (*c*) (*d*)

Figure 4.5: Break down of frame processing time for the single session Dyson robotics lab sequence. Red curves correspond to a NID threshold $\tau = 0$ (i.e. NID keyframing turned off). Blue curves correspond to an NID threshold of $\tau = 0.86$. (*a*) Frame processing time as a function of frame number. As mapping proceeds, the per-frame processing time of the session running without NID keyframing steadily increases. The per-frame processing time of the session running NID keyframing remains relatively flat. (*b*) While there is an overhead to computing the NID this cost is amortised over the course of the exploration of the office. (*c*) Whether a specific frame was fused in each of the two sessions. (*d*) The total number of surfels in the map at each point in the sequence.

# CHAPTER 5

# A Hybrid Sparse-Dense Monocular SLAM System

The use of active depth sensors and direct camera tracking methods limits, for the most part, dense mapping systems, such as ElasticFusion, upon which the developments of Chapters 3 and 4 were based, to indoor environments and relatively slow camera movement. However, dense mapping systems, and their resultant high fidelity $3D$ reconstructions, could potentially confer many benefits to outdoors robotics applications, such as autonomous driving, facilitating higher-level scene understanding, perception, and planning. However, before they can be used effectively in outdoors environments, the limitations induced by the sensors and algorithms employed in dense mapping need to be addressed.

In this chapter we present a system for incrementally reconstructing a dense $3D$ model of the geometry of an outdoor environment using a single monocular camera attached to a moving vehicle. Our system employs neural network-based dense depth prediction with a hybrid mapping architecture combining state-of-the-art sparse features and dense fusion-based visual SLAM algorithms within an integrated framework. Our novel contributions include the design of hybrid sparse-dense camera tracking and loop closure mechanisms. We use the motion estimates from the sparse method to overcome the large and variable inter-frame displacement typical of outdoor vehicle scenarios. Our system then registers the live image with the dense model using whole-image alignment. This enables the fusion of the live frame and dense depth prediction into the model. Global

consistency and alignment between the sparse and dense models are achieved by applying pose constraints from the sparse method directly within the deformation of the dense model.

We use outdoor autonomous driving as a representative example of a setting that is less constrained than the indoor setting targetted by ElasticFusion and other contemporary dense visual SLAM systems. Results of qualitative and quantitative experiments, for both trajectory estimation and surface reconstruction accuracy, demonstrate that our system achieves competitive performance on the KITTI autonomous driving dataset. The work in this chapter was published in

- L. Gallagher, V. Kumar, S. Yogamani, and J. McDonald. A Hybrid Sparse-Dense Monocular SLAM System for Autonomous Driving. In *2021 European Conference on Mobile Robots (ECMR)*, pages 1–8, 2021. doi: 10.1109/ECMR50962.2021.9568797

## 5.1 Background

Over the past decade, approaches to fusion-based dense visual SLAM have demonstrated the ability to build high fidelity dense $3D$ models of an environment facilitating higher-level scene understanding, perception and planning (Dai et al., 2017; Newcombe et al., 2011a; Whelan et al., 2014a, 2015). However, given that these techniques typically require active RGB-D sensors, their applicability in outdoor scenarios has been limited. At the same time, sparse and semi-dense monocular systems have found a large degree of success in these environments but are limited in their resulting sparse or partially dense representations (Campos et al., 2021; Engel et al., 2014, 2017; Forster et al., 2014, 2017; Mur-Artal and Tardós, 2017a,b; Mur-Artal et al., 2015). In this chapter, we investigate the potential of recent results in monocular dense depth prediction in combining these approaches' strengths. In particular, we employ dense depth prediction in developing a monocular SLAM system that achieves comparable accuracy to sparse camera tracking algorithms while allowing dense fusion-based modelling of the environment. We demonstrate the effectiveness of

this approach by tracking and mapping an environment from a single monocular camera attached to a moving vehicle. In this section, we discuss the considerations that motivate the design of our system for outdoor dense monocular SLAM.

Visual SLAM systems based around the concepts of dense alternation, every pixel fusion, and direct whole-image alignment-based camera tracking, have shown that it is possible to build dense, high-fidelity models of indoor scenes in real time with active depth sensors such as the Microsoft Kinect. These systems and the models they produce have many benefits in downstream machine perception and spatial reasoning tasks. They can produce 'watertight' maps that model the continuous surface that underlies the scene. For example, this allows them to synthesize accurate novel views and realistically fuse augmentations into the scene for AR effects (Dai et al., 2017; Newcombe et al., 2011a; Schöps et al., 2019; Whelan et al., 2014a, 2015). High-resolution dense models may become more necessary for robot planning in the future to accommodate a broader range of scenarios (e.g., analyzing surface geometry in unevenly paved roads Fallon et al. (2015), obstacle detection and avoidance Wang et al. (2019b), etc.).

In this chapter we focus on ElasticFusion (see Section 3.1), a dense visual SLAM system built around the idea that decoupling the processing of each frame into two alternating steps - direct every pixel camera tracking and fusion of the current frame into the map - can lead to accurate dense visual SLAM. Though contrary to the well established theory that accuracy in visual SLAM necessitates joint estimation of both camera trajectory and scene geometry, the alternation approach taken by ElasticFusion, and many other dense SLAM systems, has proven to be necessary in order to cope with the large amount of data required for by dense mapping. It has also proven effective within the relatively constrained setting of indoor, hand-held, RGBD visual SLAM.

Different principles for active RGBD sensing have been proposed, with structured light and time-of flight (ToF) principles being two of the most common. In both cases, light is actively projected into the scene in order to measure the distance of the surface from the sensor. During outdoor operation, external light from the sun interferes with light projected by the sensor leading to noisy depth images (El-laithy et al., 2012; Fankhauser et al., 2015).

In addition, fast motion of the camera and dynamic scene elements, which one would expect in autonomous driving (Geiger et al., 2012a, 2013), leads to rolling shutter and motion blur artefacts in the depth maps produced by active depth sensors (Guo et al., 2018; Tourani et al., 2016). Since ElasticFusion relies on an RGBD camera to estimate scene geometry, its effectiveness in outdoors settings is limited.

Using passive vision sensors over active sensors, such as RGBD cameras and LiDAR, in autonomous driving SLAM systems has many advantages. A calibrated monocular camera is inexpensive, lightweight, energy efficient, can be deployed in a diverse range of environments, such as in underwater (Kim and Eustice, 2013) and space applications (Matthies et al., 2007), and can be used as a direction sensor, providing rich photometric and geometric measurements of the scene at every frame. In contrast, LiDAR, for example, although highly accurate, is expensive, heavy, and produces a relatively sparse signal compared to the dense measurements provided by a vision sensor. Indeed, current research on monocular depth estimation and object detection indicates the potential for cameras to either match or exceed the performance of LiDAR on these tasks (Wang et al., 2019b; You et al., 2020).

Multi-sensor platforms attempt to leverage the combined advantages of monocular, inertial, stereo, LiDAR, and GPS sensors. However, such platforms need careful internal and external calibration, which can be done online to maintain synchronisation and geometric alignment between the sensors and, ultimately, achieve accurate performance. In the autonomous driving domain, even the most carefully calibrated rig will move and warp in real time due to the effects of heat, wear and tear of mechanical parts like tires, variations in loading of the vehicle, and the knocks and bumps inherent in driving on roads (Dang et al., 2009; Mueller and Wuensche, 2016, 2017). GPS comes with its complexities, requiring line-of-sight with at least four satellites to infer position, which can prove challenging in urban environments (Abel and Chaffee, 1991). Physical challenges aside, fusing sensor signals from different modalities is often non-trivial (e.g., visual-inertial Leutenegger et al., 2015).

In addition to the limitations imposed by active sensors, the direct approach to

camera tracking, favoured by many dense SLAM systems, typically assumes that inter-frame displacement of the camera is relatively small. When it comes to dense mapping for autonomous driving, large inter-frame displacement of the camera, caused by the fast motion of the vehicle, violates this important assumption. For example, in the KITTI autonomous driving dataset (Geiger et al., 2012a), the vehicle, at times, exceeds speeds of 80km/h. Since the cameras attached to the vehicle operate at 10hz this means that the inter-frame displacement of the cameras can exceed 2.22m. For accurate direct camera tracking, the optimisation procedure must be initialised with a camera pose that lies within the basin of convergence of the globally optimal solution. If it is not, convergence to a non-optimal local minimum is possible. In practical terms, without a good initialization pose, camera tracking performance deteriorates. The closer inter-frame displacement is to zero the better tracking performance is going to be. While increasing the camera frame-rate can help bring the displacement closer to zero, doing so only helps up to a point. Increasing the frame-rate decreases the exposure time wherein the camera sensor integrates light. Irrespective of the quality of camera hardware, the less exposure time a sensor has the more susceptible it is to photon noise (Hasinoff, 2014). In a poorly lit scene, decreasing exposure time hits the point of diminishing returns much quicker. At some point, the negative effect of image noise is going to outweigh the positive effect a small baseline has on camera tracking performance (Handa et al., 2012).

Highly variable outdoor lighting, due to changing weather and lighting conditions, breaks the fundamental constraint of direct photometric tracking, the *photo-constancy constraint*. This fundamental constraint says that a surface point, when imaged with a moving camera, will result in the same pixel value at corresponding pixel sites. This constraint is at the best of times a gross simplification of the reality of imaging a scene. However, if it is *approximately* true, deviations in the colours of corresponding pixels, due to systematic and non-systematic (random) sources of error, can be modelled and accounted for or annealed by robust estimation methods. If such sources of error cannot be accounted for, or the light in the scene behaves in an inherently non-Lambertian way (i.e. the brightness of points on the surface are dependent on the point of view of the observer) then

direct camera tracking optimisation procedures based on the photo-constancy constraint will fail.

In contrast to dense systems, sparse visual SLAM systems, based on joint estimation of scene structure and camera motion, have proven to be effective in outdoors settings and during fast camera motion. Recently, several systems have emerged that represent the state-of-the-art in traditional monocular SLAM, and visual-odometry performance (Campos et al., 2021; Engel et al., 2014, 2017; Forster et al., 2014, 2017; Gao et al., 2018; Mur-Artal and Tardós, 2017b). The ORB-SLAM open-source SLAM frameworks (Campos et al., 2021; Mur-Artal and Tardós, 2017b; Mur-Artal et al., 2015) in particular represent the state-of-the-art in monocular SLAM. Similar to PTAM, ORB-SLAM (Mur-Artal and Tardós, 2017b) splits the SLAM problem into sub-problems that can be solved in different threads; (i) a feature-based camera tracking thread that runs in real time; (ii) a local mapping thread that takes keyframes extracted by the first thread and optimizes a local area of the map around this keyframe using bundle-adjustment (BA); and (iii) a global mapping thread that integrates loop closure constraints and maintains global consistency.

However, in the absence of prior information, monocular vision systems suffer from ambiguity and drift in the estimated scale of the reconstruction (Strasdat et al., 2010a), and are not as robust as stereo and multi-sensor systems in texture-less regions and during fast camera motion (Campos et al., 2021; Mur-Artal and Tardós, 2017a). Furthermore, with the exception of a few techniques for dense and semi-dense map representations (Engel et al., 2013, 2014, 2017; Newcombe et al., 2011b), these systems produce sparse reconstructions which only offer a coarse-grained description of the scene's geometry (Campos et al., 2021; Davison et al., 2007; Eade and Drummond, 2008; Klein and Murray, 2007; Mur-Artal and Tardós, 2017b; Mur-Artal et al., 2015).

More and more deep learning is being used to improve the performance of SLAM systems. Broadly speaking, neural networks have been applied in two critical areas within the SLAM problem; in the representation of geometric and visual information (Bloesch et al., 2018; Costante et al., 2016; Sucar et al., 2021), and in overcoming the ill-posedness of many of SLAM's sub-problems, for example, depth estimation from monocular images

(Eigen et al., 2014; Godard et al., 2017; Ravi Kumar et al., 2020; Zhou et al., 2017). Recent results on neural network-based depth estimation has shown that it is possible to train a network to predict accurate metric depth from a single image, with only weak supervision on the predicted depth's scale (e.g., Guizilini et al., 2020; Ravi Kumar et al., 2020), which, as we show in this chapter, and in Chapter 6, paves the way for metric-scale dense monocular SLAM.

Several systems have been presented that demonstrate hybrid approaches to monocular visual SLAM and VO (Bloesch et al., 2018; Czarnowski et al., 2020; Tateno et al., 2017; Yang et al., 2020), while other approaches implement a fully learned SLAM pipeline (Wang et al., 2017; Zhou et al., 2018). In comparison to this work, these systems have either not been designed with large-scale fast, outdoor camera motion in mind (Bloesch et al., 2018; Czarnowski et al., 2020; Tateno et al., 2017; Zhou et al., 2018), or do not produce dense models (Wang et al., 2017; Yang et al., 2020).

The work most similar to our approach is the dense surfel-based fusion system of Wang et al. (2019a), combining camera tracking from a sparse SLAM system (ORB-SLAM2 (Mur-Artal and Tardós, 2017b)) with a dense fusion-based back-end. The substantial difference with the current work is the degree to which they couple the sparse and dense maps. In Wang et al. (2019a), a sub-mapping approach is employed where each surfel is anchored to a reference keyframe from the sparse system. Local and global optimisations of the keyframe poses by the sparse system are reflected in the dense map by ensuring the pose of a surfel relative to its anchor keyframe remains constant. In contrast, our system maintains a much looser coupling between the sparse and dense representations. The sparse pose is used to initialise a dense alignment that registers the camera to the dense map allowing for fusion of the current frame. Loop closures from the sparse system are used to constrain a deformation graph that non-rigidly deforms the surfels themselves. In this way, the benefits of each system are leveraged, while the systems themselves are only indirectly connected through 6-DOF transformations and $3D$ constraints. Furthermore, we quantitatively demonstrate the effectiveness of our approach in outdoor automotive environments using deep learning-based depth estimation.

The main contribution described in this chapter is the development of a system for live dense metric $3D$ reconstruction in outdoor environments using a single RGB camera attached to a moving vehicle. Although other researchers have reported dense and semi-dense SLAM or dense VO approaches, to the best of our knowledge, this is the first fully dense SLAM system *quantitatively* evaluated on automotive scenarios. The novelty of our approach lies in the use of a dense depth prediction network (Ravi Kumar et al., 2020) within a hybrid architecture, combining state-of-the-art sparse feature tracking and dense fusion-based visual mapping algorithms in a loosely coupled fashion.

Using the motion estimates from the sparse method to overcome the large and variable inter-frame displacement typical of outdoor vehicle scenarios, our system then registers the live image with the dense model using whole-image alignment. This allows dense depth prediction and fusion of the live frame into the model. Global consistency and alignment between the sparse and dense models are achieved by applying pose constraints from the sparse method directly within the deformation of the dense model. We evaluate the method over the KITTI benchmark dataset, providing qualitative and quantitative results for both the trajectory and surface reconstruction accuracy.

## 5.2 A Hybrid Approach

We take a hybrid approach to dense monocular tracking and mapping. A feature-based SLAM system, ORB-SLAM3 (Campos et al., 2021), is used to provide an initial estimate of the camera's pose for each frame. The pipeline then follows a dense alternation architecture, extending the ElasticFusion (EF) SLAM system (Whelan et al., 2015), in which the map is first held fixed. At the same time, the camera is tracked against it, using the initial pose estimate from the previous step. Once the camera pose has been estimated, it can fuse the current frame into the map. We densely predict per pixel metric depth estimates using a SoTA self-supervised convolutional neural network, UnRectDepthNet (Ravi Kumar et al., 2020). The various sub-systems predominantly operate on different processors; the GPU is used extensively by the dense alternation and depth prediction network, while ORB-SLAM operates on the CPU. Our hybrid architecture is summarised as follows:

1. A scale-aware depth prediction network is used to estimate a metric depth map for each frame. An initial estimate of the camera motion is made using ORB-SLAM's feature-based camera tracker, which is suited to the fast motion of the vehicle.

2. The initial pose estimate is further refined by aligning it to the current active model in view of the camera.

3. Live RGB images and corresponding predicted metric depth maps are fused into a global dense surfel model of the scene. The surfel model is divided into an active and an inactive portion as per the original EF algorithm.

4. When ORB-SLAM identifies a loop closure, we use the resulting loop closure constraint within the EF deformation graph to correct the geometry of the dense surfaces. This brings the previously visited inactive portion of the map back into alignment with the current active portion. Importantly, this also keeps the different map and camera trajectories of ORB-SLAM and EF consistent.

Figure 5.1 shows the architecture of our system. In the remainder of this section, we will describe how these key elements are combined into one system that builds consistent dense surfel models from a monocular camera attached to a moving vehicle. The section provides complete coverage of the system where the main system components are covered in Section 5.2.1, Section 5.2.4, and Section 5.2.5. We then present and discuss experimental results that demonstrate the efficacy of our system in Section 5.3.

### 5.2.1 Scale-aware depth estimation

The system employs UnRectDepthNet to make per-pixel depth estimates at known scale (Ravi Kumar et al., 2020). While the core depth estimation algorithm used here is the same as reported in Ravi Kumar et al. (2020), several novel innovations to the base method were introduced in Gallagher et al. (2021). However, these novel innovations are not contributions of this thesis. A description of the depth estimation module is included here for completeness.

UnRectDepthNet establishes a *structure-from-motion (SfM)* framework for self-supervised depth estimation. In particular, separate depth and pose estimation networks are jointly trained with a novel view synthesis loss function. Training examples consist of an unlabelled reference image along with several source images taken from overlapping points of view. The depth network takes as input the target image and predicts a depth estimate for each pixel site. The pose network takes as input the target image and the source images and, for each source image, predicts the relative transformation from the target image to the source image.

Given known camera intrinsics, per-pixel depth estimates from the depth network, and relative poses between cameras, each pixel in the source image can be warped to each of the target images. The basis for self-supervised depth estimation supervision is the photometric similarity between each target pixel and the source pixels it warps to. Additional loss terms on depth smoothness, feature space representations of the input images (Shu et al., 2020) and cross-sequence depth consistency help to regularize the network. Scale ambiguity is a challenging problem in monocular depth estimation (Garg et al., 2016; Zhou et al., 2017). As a result, an absolute value is needed to serve as an anchor point, provided through a measurement from another dedicated sensor to obtain actual depth estimation. In this work, UnRectDepthNet was trained with weak supervision on a combination of Velodyne point cloud and calibration information to improve the estimation of scale. Velodyne Lidar serves as proxy for ground truth and the scale factor of predicted depth maps is estimated by associating the calculation with the correct pixel in the image plane, minimizing $\ell_2$ loss.

UnRectDepthNet supports a number of camera models for standard pinhole and wide FOV fisheye cameras. For this system, we carried out the view synthesis by employing the pinhole camera projection model. In Chapter 6 we go into more detail on self-supervised depth estimation where we introduce a novel reformulation of a self-supervised depth network in terms of the Kannala-Brandt camera model. For the interested reader the seminal work of (Zhou et al., 2017) and references therein provide more details.

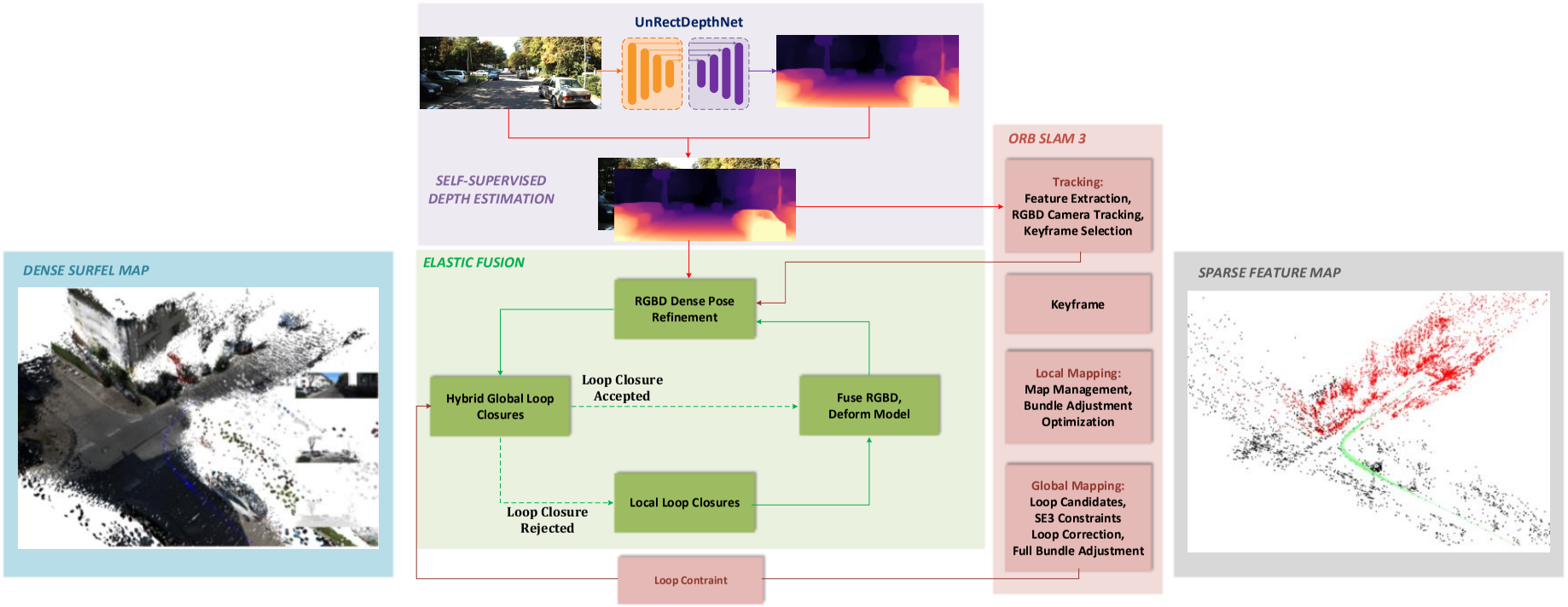



Figure 5.1: **An overview of our system's architecture.** UnRectDepthNet is used to estimate a depth map for the current live camera image. ORB-SLAM then tracks the motion of the camera since the last frame. Internally ORB-SLAM continues to extract keyframes and pass them to its local and global mapping threads. The sparse feature map on the right is used internally by ORB-SLAM. Once the camera motion has been estimated, the pose is used to synthesize a dense view of the current active surfel model. The pose estimate is further refined by performing a direct joint photometric and geometric alignment between the synthetic and live RGB-D frames, taking the pose estimate from ORB-SLAM as an initialization point. Assuming no local or global loop closures have occurred during this time-step, the live RGB image and predicted depth are fused into the surfel model. If ORB-SLAM has detected a global loop closure, then a global deformation is attempted. Surface-to-surface constraints for optimisation of the deformation graph are generated from the pre-corrected and corrected (post loop closure) poses of the camera $\mathbf{P}_{kf}^{t}$ and $\hat{\mathbf{P}}_{kf}^{t}$ respectively. If deformation graph optimisation fails or no global loop is detected by ORB-SLAM, a local loop closure is attempted as per the original EF algorithm.

### 5.2.2 ORB-SLAM3 - Feature-based RGBD Tracking

ORB-SLAM3 (Campos et al., 2021) builds a sparse map of a scene, represented using a *covisibility* graph where each node in the graph corresponds to a keyframe consisting of a pose and a set of $3D$ points. When two keyframes view common map points, an edge between them is added to the graph. The system has 3 main threads. The camera tracking thread receives RGB-D frames from the camera, extracts ORB features, and computes an initial pose estimate via motion only BA with the previous frame. The estimate is further refined by aligning the current frame to a local map of covisible keyframes. New keyframes are detected and sent to a background thread. Here the local map in the vicinity of the new keyframe is optimized with a full bundle adjustment. Loops between the latest keyframe and historic keyframes are detected with DBoW (Galvez-López and Tardós, 2012). If geometric alignment between the new keyframe and the matched keyframe succeeds, then a loop is closed, and an edge between the two keyframes is added to the graph. The local map in the vicinity of the keyframe is rigidly transformed into place. The rest of the keyframe graph is corrected using pose graph optimisation. A final full BA is performed to recover the MAP estimate of all keyframe poses and structure.

### 5.2.3 Dense Surfel Fusion

The dense map representation and estimation pipeline used by our system builds on ElasticFusion (Whelan et al., 2015), which we described in detail in Section 3.1. Our system estimates a dense surfel map $\mathcal{M}$ of a scene as viewed from a monocular RGB camera. The map $\mathcal{M}$ is a flat list structure containing a set of surfels $\mathcal{M}^s$. As before, each $\mathcal{M}^s$ contains a $3D$ position, a normal, a radius $r$ indicating its extent, a confidence value indicating the quality of its estimation, an RGB colour, a timestamp $t$ when it was last fused, and a timestamp $t_o$ when it was inserted into the map. During operation, $\mathcal{M}$ is divided into two disjoint subsets; an active portion containing surfels that have recently been measured by the live camera data and an inactive portion containing those surfels that have not been measured in a threshold amount of time. in Section 5.2.4, we detail the

systems hybrid camera tracking procedure, which combines direct whole image alignment with feature-based camera tracking. RGB frames, and their corresponding dense depth estimates, are fused into the global model by back projecting each pixel to a $3D$ point using the depth map and the camera's intrinsic matrix. ElasticFusion also has a visual place recognition loop closure mechanism based on fern encoding; however, we do not utilize this module. As with ElasticFusion, local loops are identified by aligning views of the active and inactive portions of $\mathcal{M}^s$ in the current camera view, allowing inactive surfels to be reactivated. Both global and local loop closures provide surface-to-surface constraints to optimize a space deforming graph, which is then applied to $\mathcal{M}^s$. In Section 5.2.5, we describe how we bootstrap the dense deformation graph from loop closure constraints coming from the sparse system.

### 5.2.4 Hybrid Camera Tracking

In our system, at each time-step, $t$, the current live camera frame $\mathcal{F}_t$, consisting of colour image $I_t$ and predicted depth map $\mathcal{D}_t^p$, is passed to ORB-SLAM to compute an initial estimate of the camera pose $\mathbf{P}_t$. Note that this is the camera pose before any refinement by the local mapping thread in the case that $\mathcal{F}_t$ is extracted as a keyframe. Though this initial pose is accurate (as can be seen in Table 5.1), it cannot be used directly for the fusion of $\mathcal{F}_t$ into $\mathcal{M}$. ORB-SLAM continues to improve its local map, while the dense surfel map does not undergo any significant correction until a local or global loop closure is triggered. This can lead to $\mathbf{P}_t$, and therefore the live frame, becoming misaligned from the dense map. Our goal is to track the camera *and* to build an accurate model, and so we need to balance the accuracy of $\mathbf{P}_t$ against the need to stay aligned with the dense model in order to fuse $\mathcal{F}_t$ accurately.

To bring the camera back into alignment with the model, we perform a frame-to-model refinement of $\mathbf{P}_t$. The active map around $\mathbf{P}_t$ is rendered into a virtual camera positioned at $\mathbf{P}_t$. A 6DOF transformation $\mathbf{T} \in SE(3)$ that aligns the live frame to the virtual one is then estimated in an iterative non-linear least-squares joint photometric and geometric optimisation embedded in a 3 level image pyramid. Composing $\mathbf{T}$ with $\mathbf{P}_t$ yields

a refined pose estimate for the current frame that can now be used to fusion as per the original EF algorithm (Whelan et al., 2015).

### 5.2.5 Hybrid Loop Closures

When ORB-SLAM identifies a loop closure between the latest keyframe $Kf_i$ and a historic keyframe in the map, $Kf_h$, a transformation $\mathbf{T}_i^h$ that aligns $Kf_i$ with $Kf_h$ is computed. This yields a pose pair; the uncorrected pose of the new keyframe $\mathbf{P}_{kf}^i$ and the pose of the keyframe after loop correction $\hat{\mathbf{P}}_{kf}^i$. ORB-SLAM's loop closure mechanism proceeds to apply the loop closure and optimize its sparse map as per the original algorithm (Campos et al., 2021).

To update the dense surface to reflect the loop closure $\mathbf{P}_{kf}^i \rightarrow \hat{\mathbf{P}}_{kf}^i$, we non-rigidly deform each surfel using a deformation graph $\mathcal{G}$ in the same way as EF (Whelan et al., 2015). $\mathcal{G}$ consists of a set of nodes $\mathcal{G}^n$ sampled from the surfels in $\mathcal{M}$. Each node consists of a position (i.e. the sampled surfel's position), a timestamp (also sampled surfel's timestamp), an affine transformation $\mathcal{G}_{\boldsymbol{T}}^{\boldsymbol{n}}$, and a set of temporally nearby neighbour nodes. The temporal connectivity of the nodes prevents interference between multiple passes of the same surface at different points in time. To apply $\mathcal{G}$ to $\mathcal{M}$, a set of temporally and spatially nearby influencing nodes is computed for each $\mathcal{M}^s \in \mathcal{M}$. A weighted sum of the $\mathcal{G}_{\boldsymbol{T}}^{\boldsymbol{n}}$ in these nodes are applied to the surfel to deform it into place.

To optimize the $\mathcal{G}_{\boldsymbol{T}}^{\boldsymbol{n}}$, a set of surface-to-surface correspondences are generated from a loop closure, mapping surface points from the active region to the inactive region of the model. Cost function terms on the distance between correspondences, as well as terms that maximize the rigidity and smoothness of the deformation, ensure that, when applied to the surface, it brings the correspondences together. A pinning term holds the inactive map in place, ensuring that it is the active map that is deformed into the inactive map. Further constraints on the relative position of historical correspondences from previous loop closures prevent new deformations from breaking the old loop closures. The final cost is optimized using Gauss-Newton gradient descent to retrieve the set of deforming affine transforms $\mathcal{G}_{\boldsymbol{T}}^{\boldsymbol{n}}$. See Whelan et al. (2015) for more details.

| Sequence | **O** $t_{rel}(\%)$ | **H** $t_{rel}(\%)$ | **H+L** $t_{rel}(\%)$ | **D3** $t_{rel}(\%)$ | **ORB** $t_{rel}(\%)$ |
|---|---|---|---|---|---|
| 01 | 31.17 | 24.49 | 25.34 | 1.07 | 1.39 |
| 02 | 2.97 | 3.11 | 2.38 | 0.80 | 0.76 |
| 06 | 1.16 | 1.42 | 1.34 | 0.67 | 0.51 |
| 08 | 1.90 | 2.14 | 2.11 | 1.00 | 1.05 |
| 09 | 1.85 | 2.02 | 1.62 | 0.78 | 0.87 |
| 10 | 1.24 | 1.50 | 1.41 | 0.62 | 0.60 |

Table 5.1: **Results on test set of KITTI odometry**. Results from ORB-SLAM stereo (Mur-Artal and Tardós, 2017b) and a SOTA VO system D3VO (Yang et al., 2020) are shown for comparison. Though not SOTA, these results show that our system's camera tracking achieves a level of accuracy necessary for building a fused $3D$ surfel model. It can be seen that using predicted depth leads to accurate RGBD tracking with ORB-SLAM (**O**). We show the effect of introducing pre-fusion alignment (**H**) and applying global loop closures to the dense model (**H**+**L**).

In order to reflect ORB-SLAM loop closures in the dense model in our system using the deformation graph, we first generate a set of surface-to-surface correspondences from $\mathbf{P}^i_{kf} \rightarrow \hat{\mathbf{P}}^i_{kf}$. To do so, the active surfels $\mathcal{M}^a \in \mathcal{M}$ in view of $\mathbf{P}^i_{kf}$ are computed, as are the timestamps of the inactive surfels in view of $\hat{\mathbf{P}}^i_{kf}$. These timestamps are used during optimisation of $\mathcal{G}^{\boldsymbol{n}}_{\boldsymbol{T}}$ to pin the inactive surfels of $\mathcal{M}$ in place. Each of the active surfels $\mathcal{M}^a_s$ generates a constraint consisting of a source point, computed by composing $\mathcal{M}^a_s$ with $\mathbf{P}^i_{kf}$, and a destination point computed by composing $\mathcal{M}^a_s$ with $\hat{\mathbf{P}}^i_{kf}$. These constraints are then used to optimise the deformation in the same way as Whelan et al. (2015).

Note that $Kf_i$ is not necessarily the current live camera frame. It could therefore fall outside the current active region of the map. If it is outside the active region of the map, then this will hinder the system's ability to generate surface-to-surface constraints. We found that by setting the active time window appropriately ($\delta t = 200 frames$), the system could find sufficient constraints during loop closures.

Closing loops in this way achieves two goals; adjusting the dense surface geometry to stay consistent with the real world; and keeping the dense map consistent with ORB-SLAM's sparse map and camera pose estimates. It also balances the need for this consistency against the computational intensity of correcting dense geometry. Figure 5.2 illustrates each of the steps involved in the hybrid loop closure process.

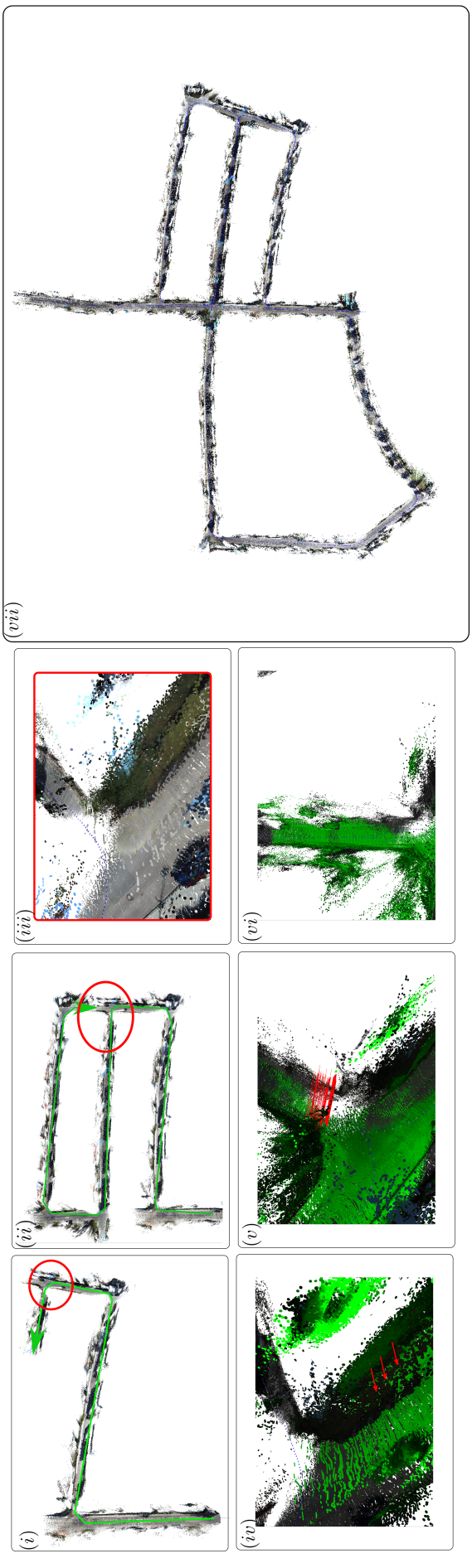



Figure 5.2: **A loop closure example**; (*i*) a car begins exploring (green arrow); (*ii*) after a period of time a loop is closed between an area previously mapped by the car and its current location; (*iii*) before a loop closure is applied, drift is evident in the model; (*iv*) the active portion of the model (green) has drifted from the inactive portion of the model (grey). Red arrows highlight corresponding points between the active and inactive map; (*v*) a global surface deformation has been applied bringing the active portion of the map back into alignment with the inactive portion. Red lines are surface-to-surface constraints generated from the loop closing pair $\mathbf{P}_{kf}^{t} \rightarrow \hat{\mathbf{P}}_{kf}^{t}$; (*vi*) as the car moves back into the region of the road already mapped in the first pass, local loop closures are triggered, reactivating portions of the map for use in mapping and tracking; (*vii*) the final global model.

## 5.3 Evaluation

We show quantitative and qualitative results of our system tested on the KITTI odometry benchmark dataset (Geiger et al., 2012a). We have evaluated the accuracy of trajectory estimation and surface reconstruction and provide a breakdown down of the computational performance of the system. All processing was done on a machine equipped with an Intel Core $i7-7700K$ CPU, 16GB of RAM, and an NVIDIA GTX 1080 Ti GPU. Qualitative results of the proposed approach are illustrated in https://youtu.be/Pn2uaVqjskY. Source code for the project is publicly available at the following repository https://github.com/robotvisionmu/DenseMonoSLAM. A qualitative example of our system's outputs is shown in Figure 5.3.

### 5.3.1 KITTI - Tracking

The KITTI odometry benchmark dataset provides 11 sequences with ground truth poses. We show results for sequences $01, 02, 06, 08, 09$ and 10. The remainder of the sequences are used during training of the depth prediction network and hence omitted from our evaluation. For each test sequence we use the relative translational error $t_{rel}$ averaged over sequences ranging in length from $100m$ to $800m$ (Geiger et al., 2012a). Table 5.1 shows the results alongside a SOTA SLAM system, ORB-SLAM2 (Mur-Artal and Tardós, 2017b), and D3VO (Yang et al., 2020), a SOTA VO system. Interestingly, our experiments show that using the dense depth predictions and the RGBD operating mode of ORB-SLAM (**O**) results in accurate metric-scale camera tracking with a monocular camera. While introducing hybrid tracking (**H**) increases error w.r.t the ground truth, it allows the current frame to be fused into the model. Introducing hybrid loop closures (**H**+**L**) helps bring the sparse and dense models back into alignment and reduces global error in the model and trajectory. Sequence 01 is challenging for our system. Scenes with little structure lead to a degradation in depth prediction and camera tracking. Figure 5.4 shows qualitative results of this sequence.

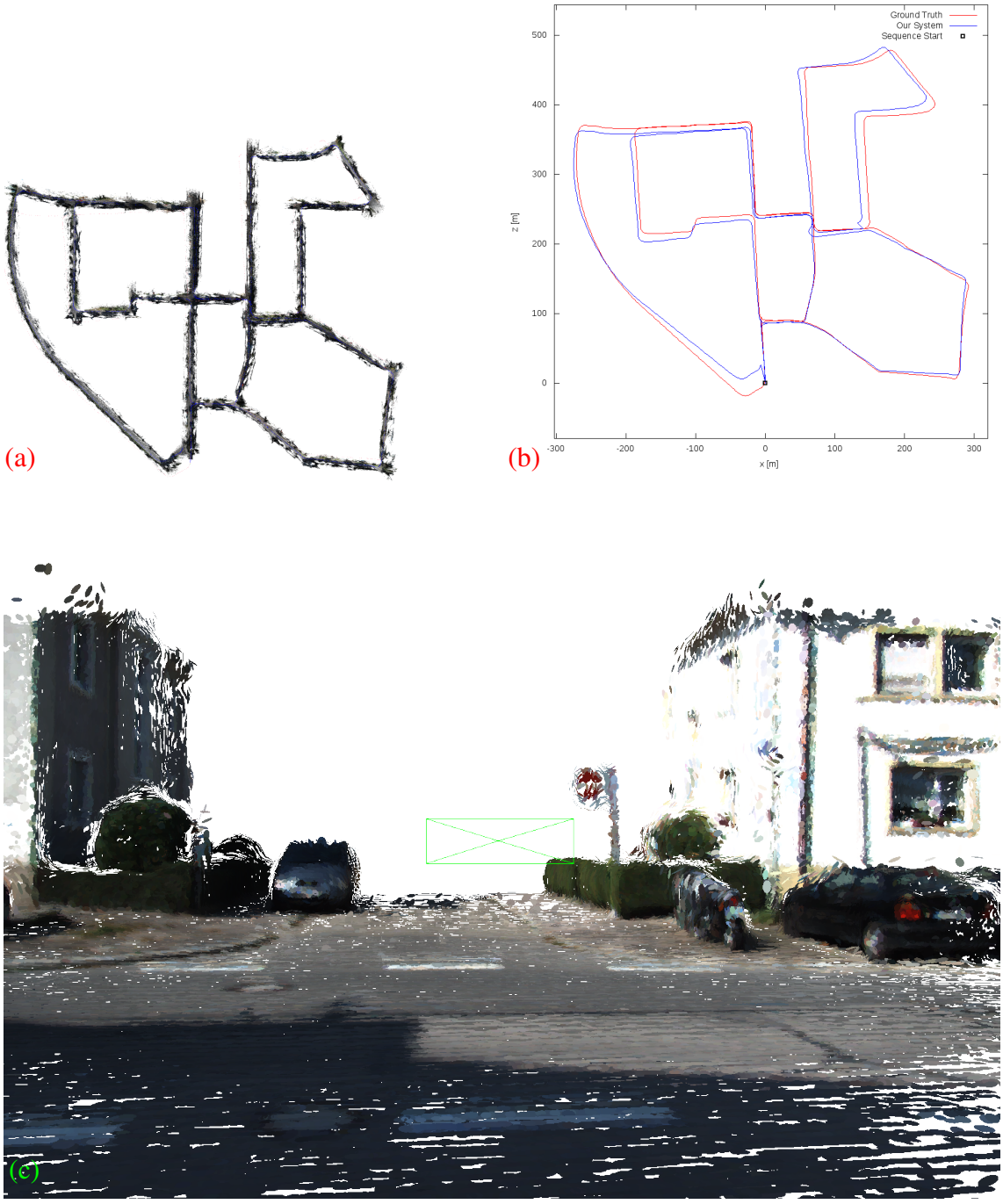



Figure 5.3: **Trajectory and dense reconstruction of an odometry sequence from Geiger et al. (2012a, 2013).** (a) shows the final surfel model produced by our system. The model contains approximately $16m$ surfels. (b) shows the estimated trajectory alongside the ground truth trajectory. (c) shows an up-close view of the reconstruction as the vehicle passes through an intersection contained in (a).

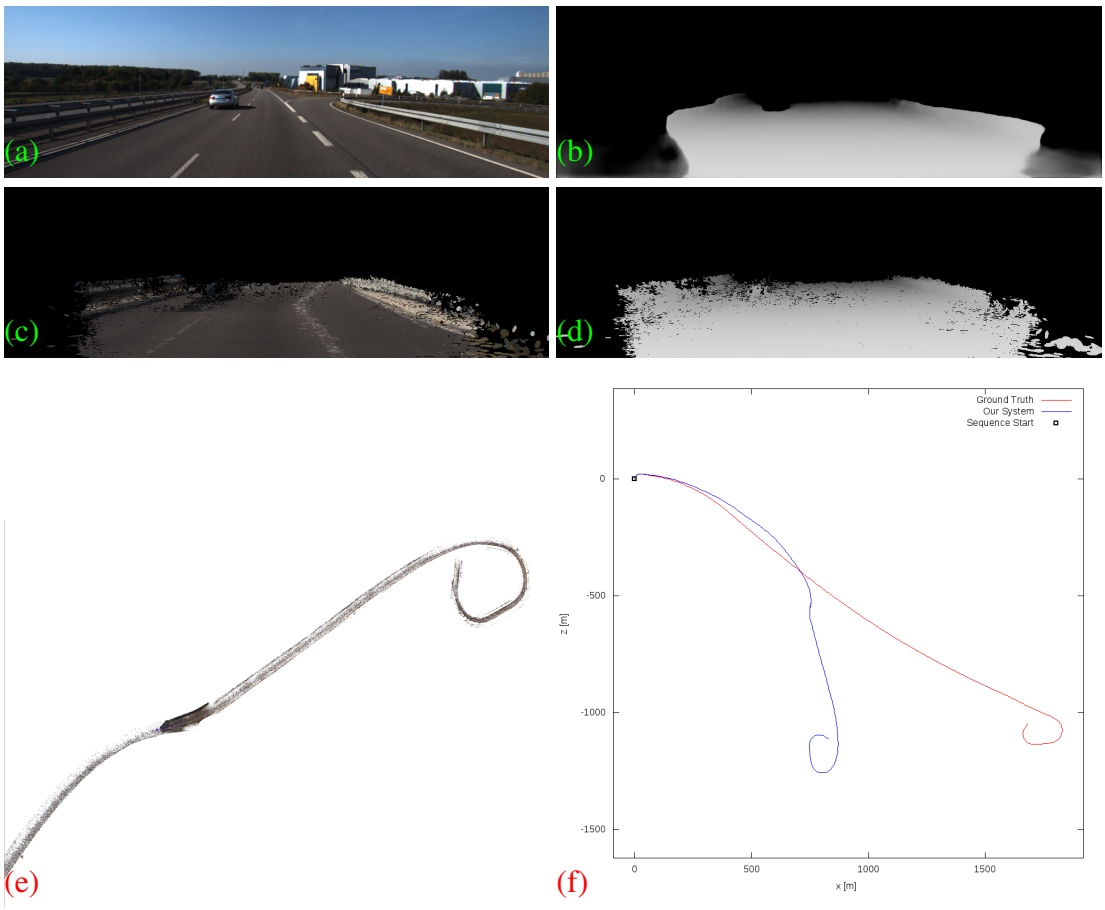



Figure 5.4: **Sequence** 01 **from the KITTI odometry dataset is challenging for monocular systems**. (a) and (b) show the live colour image from the camera and UnRectDepthNet's depth prediction respectively. (c) and (d) show ElasticFusion's corresponding model predictions. (e) shows a portion of the final global model. (f) shows a comparison between the estimated and ground-truth trajectories.

### 5.3.2 KITTI - Surface Reconstruction

We evaluate surface reconstruction accuracy on several sequences from the KITTI odometry benchmark. As KITTI does not include ground-truth surface models, we compare against models constructed from the Velodyne point clouds that accompany each sequence. To do so, we transform each point cloud in the sequence into a global coordinate system using the corresponding ground truth poses. We then apply a voxel filter to down-sample the resultant point cloud. In Table 5.2, we show the surface-to-surface mean distance between points in the estimated model and the nearest point in the Velodyne point cloud model. Before computing the score, the two models are rigidly aligned. We use the labels from Behley et al. (2019) to filter out dynamic objects from Velodyne point clouds. Our

| Sequence | O (m) | H (m) | H+L (m) |
|---|---|---|---|
| 02 | 5.95 | 6.40 | **4.03** |
| 06 | 1.23 | 0.97 | **0.64** |
| 08 | 2.70 | **2.46** | 2.72 |
| 09 | 1.23 | 1.18 | **0.71** |
| 10 | 0.89 | **0.79** | 0.82 |

Table 5.2: **Surface accuracy of our system on KITTI.** Entries represent the mean surface-surface distance between points in the estimated map and the closest point in the aligned ground truth surface model. Sequence 01 has been omitted due to poor tracking performance (see Table 5.1)

experiments show that, in most cases, hybrid tracking (**H**) improves surface reconstruction accuracy over ORB only tracking (**O**) by bringing the current frame into tight alignment with the dense model before surface fusion. Introducing hybrid loop closures (**H**+**L**) further improves surface accuracy across a number of sequences by correcting global error in the dense surface.

Our system does not achieve optimum tracking performance, with ORB-SLAM stereo and D3VO achieving better relative translational error (last two columns in Table 5.1). However, our surface reconstruction accuracy results demonstrate that tracking accuracy is sufficient for building a fused dense surface model, the main benefit of our system in comparison to ORB-SLAM stereo and D3VO. Although ORB-SLAM could be used to construct a dense point cloud, i.e. by back-projecting the full depth map estimated by UnRectDepthNet for each keyframe (as has been demonstrated for depth maps captured by a real RGBD sensor, Mur-Artal and Tardós, 2017b), it does not include a mechanism for incrementally denoising the reconstruction by fusing overlapping measurements of surface points, as our system does. Similarly, while D3VO leverages a monocular depth estimation network capable of estimating dense depth maps, the backend bundle adjustment optimisation itself is sparse. Importantly, D3VO, is a VO system and not a SLAM system, as such, its focus is on maintaining accurate tracking over a local sliding window of keyframes, and not on solving the full SLAM problem. By including the hybrid loop closure mechanism (Section 5.2.5), our system can estimate a persistent, globally consistent dense map.

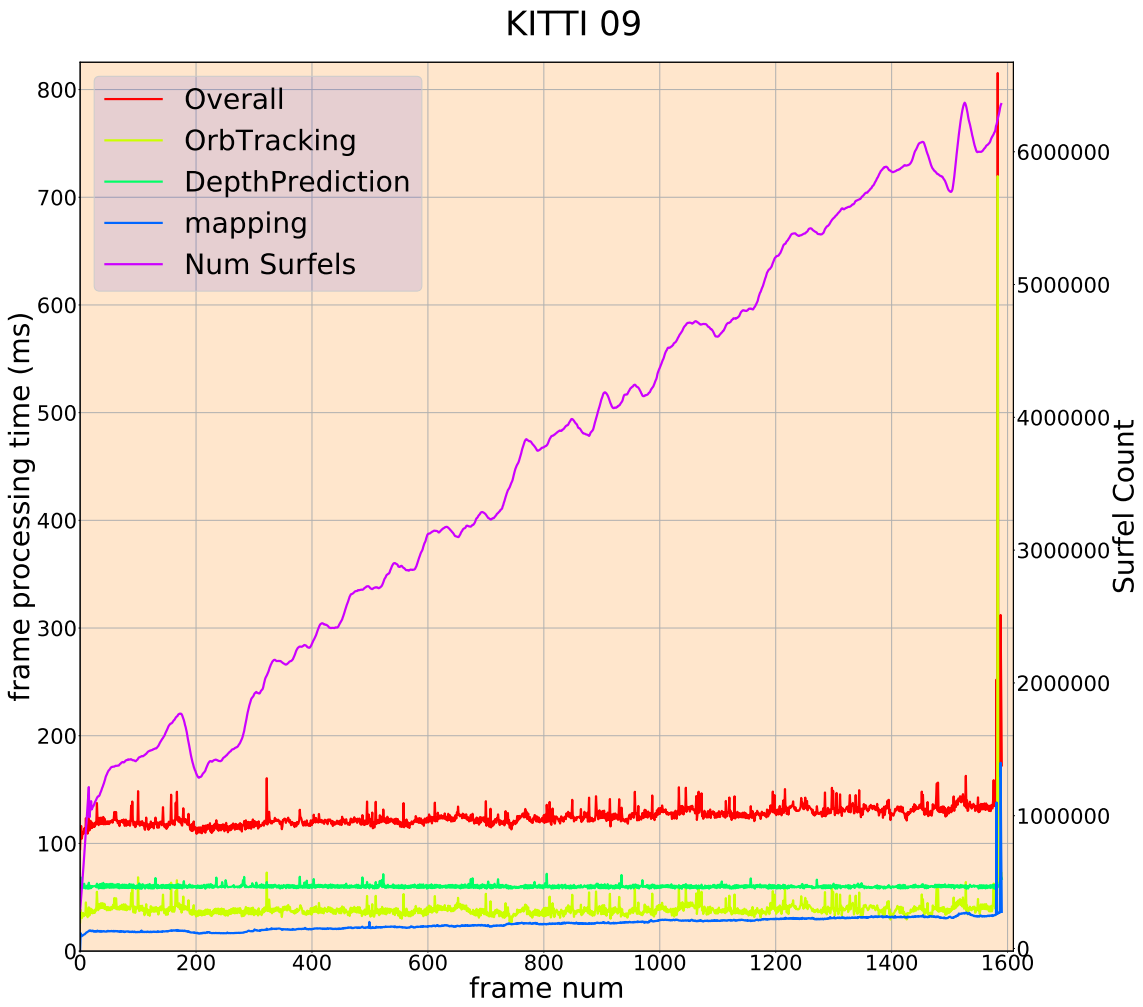



Figure 5.5: **Breakdown of time taken by our system** to process each frame in sequence 09 the KITTI odometry benchmark Geiger et al. (2012a). Our system falls just outside of true real-time rates of $10hz$ for KITTI. In future work we hope to utilize hardware with half precision support to improve system run-time.

### 5.3.3 System Resource Usage

In Figure 5.5 we show a breakdown of the frame processing time for sequence 09 in the KITTI odometry benchmark (Geiger et al., 2012a). Mapping time (blue) goes up in accordance with the number of surfels (purple) as reported in Whelan et al. (2015). The system operates between $8-9hz$. The spike at the end of the sequence is due to a global loop closure. Table 5.3 shows the distribution of run-time performance over the test set of the Eigen split of the KITTI odometry dataset.

| Operation | Median($ms$) | Mean($ms$) | std. dev.($ms$) |
|---|---|---|---|
| ORB-SLAM Tracking | 40.04 | 41.93 | 25.13 |
| Dense Mapping | 30.97 | 35.61 | 15.82 |
| Depth | 60.32 | 61.14 | 3.84 |
| Overall | 136.48 | 140.29 | 31.76 |

Table 5.3: **Mean, median and standard deviation of system run-time over test sequences from the Eigen split of the KITTI odometry dataset**

## 5.4 Summary

In this chapter we presented a hybrid SLAM system that combines dense depth prediction with sparse feature tracking and dense surfel fusion techniques. The system permits live-dense metric reconstructions of outdoor scenes using a monocular camera in automotive scenarios. Sparse tracking provides camera pose estimation capable of operating robustly at vehicle speeds. The resulting poses are used within the dense fusion tracking step to initialise a whole image alignment refinement process. Global consistency in the model is maintained through visual place recognition and pose to pose constraints from the sparse system, which again are passed to the dense fusion algorithm where they are integrated with a deformation graph-based map correction step. Our results show competitive performance with SOTA techniques while providing dense fused surface models. We believe our system to be the first dense monocular fusion-based visual SLAM system quantitatively evaluated on automotive scenarios. Although the focus of this chapter is automotive, the system could easily be adapted to other scenarios by retraining the depth prediction network.

The SLAM system presented in this chapter assumes that the input sensor can be modelled by the pinhole camera model. This limits the system to cameras with a narrow field of view (FOV). However, in many domains, including the automotive domain, wide FOV fisheye cameras are becoming increasingly popular due to their ability to keep large portions of the scene in frame at all times, making them suitable for safety critical systems. Although there are several SLAM systems that support fisheye cameras, these existing systems either don't provide dense reconstructions, don't support loop closures or use multi-camera rigs. In Chapter 6 we take the system presented in this chapter and extend it

to operate with large FOV fisheye cameras, reformulating the depth prediction and dense surface fusion stages of the pipeline in terms of the Kannala-Brandt fisheye camera model. The extended system supports operation with a single monocular fisheye camera, requiring less calibration effort than a multi-camera rig. Furthermore, the system supports loop closures for globally consistent dense mapping. As with the systems of Chapters 3 to 5 of this thesis, the goal of the system to be presented in the next chapter is to broaden the scope of dense visual SLAM so that it can be used in applications beyond the typical constrained setting of indoor mapping with a slow moving hand-held RGBD camera. To that end, we again evaluate the system's performance in the automotive domain, which acts as both a representative example of a less constrained setting, and as an example of a setting where a fisheye SLAM system has many potential benefits beyond what a SLAM system that supports only the pinhole camera model can offer.

# CHAPTER 6

# Dense Fisheye SLAM

In the previous three chapters, we have presented several extensions and adaptations to ElasticFusion, a state-of-the-art dense visual SLAM system. Although these extensions have enabled ElasticFusion's dense mapping capabilities to be brought to bear in a number of new environments and situations beyond what was previously possible, its utility is still limited by its requirement that the input sensor must be well modelled by the pinhole camera model.

In this chapter, we introduce the final contribution of this thesis; a novel dense mapping system that uses a single wide field of view (FOV) monocular fisheye camera as the sole input sensor and produces a dense surfel representation of the scene's 3D geometry. In Chapter 1, we discussed the benefits of monocular fisheye cameras in the context of safety critical systems in the automotive domain. In this chapter, We extend the hybrid sparse-dense monocular SLAM system of Chapter 5 to use large FOV fisheye imagery. This is achieved by reformulating the mapping pipeline in terms of the Kannala-Brandt fisheye camera model. Each frame is processed in its original undistorted fisheye form, with no attempt to remove distortion. To estimate depth, we introduce a new version of the PackNet depth estimation neural network (Guizilini et al., 2020) adapted for fisheye inputs. We reformulate PackNet's multi-view stereo self-supervised loss in terms of the Kannala-Brandt fisheye camera model. To encourage the network to learn metric depth during training, the pose network is weakly supervised with the camera's ground-truth inter-frame velocity. To improve overall performance, we additionally provide sparse

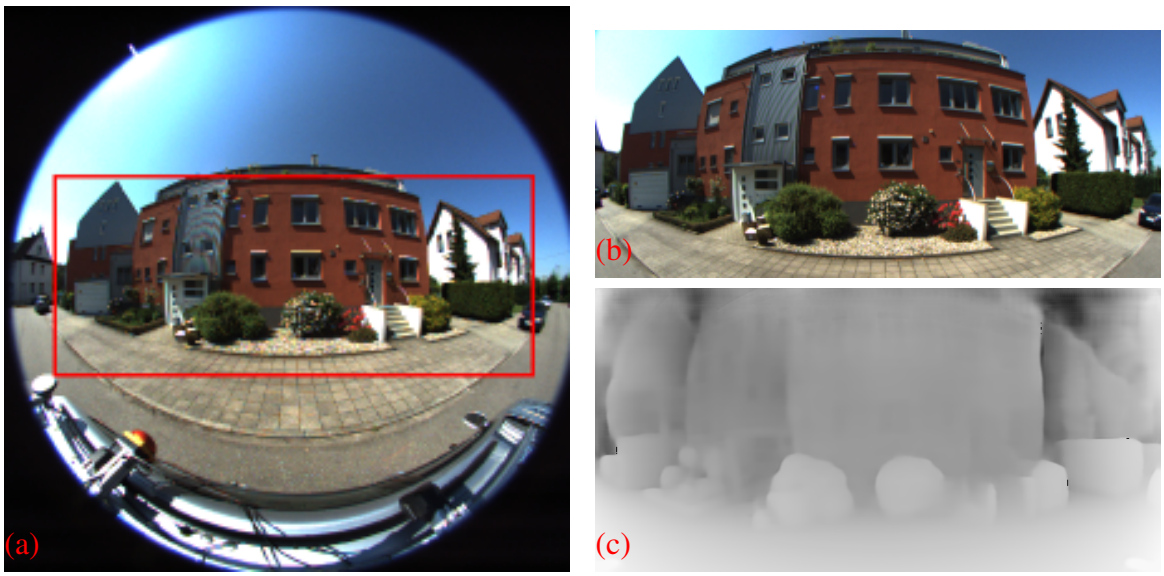



Figure 6.1: **Fisheye inputs to our system.** Images from KITTI-360 (Liao et al., 2021). (a) shows the original uncropped fisheye image (1400 × 1400). (b) shows the cropped region used by our system. We crop the image to remove the car, camera housing and the sky (c) shows the live depth estimate of (b).

depth supervision from dataset LiDAR and SICK laser scanners during training. We demonstrate our system's performance on the real-world KITTI-360 benchmark dataset. Our experimental results show that our system is capable of accurate, metric camera tracking and dense surface reconstruction within broad but local windows on the order of $160m$. Our system operates within real-time processing rates and in challenging conditions. The work in this chapter was published in

- L. Gallagher, G. Sistu, J. Horgan, and J. B. McDonald. A system for dense monocular mapping with a fisheye camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 6478–6486, June 2023

## 6.1 Background

Several SLAM systems have been presented that leverage alternate camera models to allow the use of fisheye and omnidirectional camera imagery. Caruso et al. (2015) reformulate the direct whole image alignment camera tracker and depth map filtering of LSD-SLAM (Engel et al., 2014) in terms of the unified omnidirectional camera model (Geyer and

Daniilidis, 2000). However, the system only estimates a semi-dense map. Similarly, Matsuki et al. (2018) implement an omnidirectional extension to DSO (Engel et al., 2017), reformulating the underlying optimisation in terms of the unified camera. The system presented by Matsuki et al. (2018) is a visual odometry system, not a SLAM system, and so it does not perform loop closure detection and correction. Schöps et al. (2017) use motion stereo to predict depth frames which are then integrated into a truncated signed distance function. Aggressive filtering of the depth estimates and low input resolution limits the accuracy and completeness of the resultant models. Furthermore, the system does not include a loop closure mechanism which is vital for large-scale long-term mapping and tracking. Cui et al. (2019) use a multiple-view sweeping planes-based algorithm to estimate a dense depth map for a reference camera in a multi-camera rig. Dynamic objects are detected and removed to prevent trailing artefacts from being integrated into the map. Only a local volume of space around the car is held in the map at any one point in time and no loop closure mechanism is included. Won et al. (2020) present an omnidirectional dense SLAM system where a sensor platform consisting of four 220° FOV cameras is used. The system uses a light weight omnidirectional MVS depth estimation neural network, based on the architecture proposed by Won et al. (2019), to estimate a 360° depth map for each frame. Each frame is then tracked using ROVO (Seok and Lim, 2019). Platform poses and depth maps are then fused into a dense TSDF map. Loop closures are identified using a feature-based vocabulary tree where a geometric consistency check verifies proposed matches. Once identified, loops are closed via pose graph optimisation. ORB-SLAM3 is a sparse visual SLAM framework that supports operation with fisheye cameras (Campos et al., 2021). The feature-based approach of ORB-SLAM3 lends itself to robust camera tracking during fast camera motion and in challenging conditions, such as variable lighting. We use ORB-SLAM3's feature-based camera tracking capabilities to estimate the frame-to-frame motion of the camera in our system. We also use ORB-SLAM3's loop closure mechanism to bootstrap a deformation-graph-based correction of the dense surfel map during loop closure. However, as we are interested in building dense maps in real time we go beyond ORB-SLAM3 and its low resolution reconstruction of the scene's geometry.

In contrast to these works, we focus on large-scale monocular dense mapping with loop closures. Our system uses a single fisheye camera where the incoming camera frames are integrated into a dense global surfel map. We avoid undistorting the incoming imagery and instead operate directly on the distorted fisheye images. One of the main benefits of fisheye images is their increased FOV which inevitably gets diminished during undistortion procedures. For every frame, our system detects and, if necessary, closes, loops in the vehicle's trajectory. To summarise, our contributions include:

- **Fisheye PackNet**: Reformulated PackNet (Guizilini et al., 2020) training procedure for Fisheye images, where we include velocity supervision, sparse LiDAR supervision, and the Kannala-Brandt fisheye model for multi-view stereo loss function.
- **Generic Camera Model**: Redesigned ElasticFusion's mapping pipeline to be generic (in the software engineering sense of the word) with respect to camera model. Within this genericised framework, we implemented the Kannala-Brandt camera model, allowing fusion of fisheye depth maps.
- **A Hybrid Sparse-Dense Fisheye SLAM system**: Combining ORB-SLAM3 feature-based tracking with ElasticFusion dense mapping with fisheye cameras.
- **Kitti-360 Analysis**: We provide quantitative analysis of our system's depth estimation and local surface reconstruction accuracy, using the KITTI-360 benchmark dataset. Additionally, we provide qualitative results that demonstrate the overall performance of our proposed fisheye SLAM system.

An overview of our system's inputs and outputs can be seen in Figure 6.1 and Figure 6.3, respectively. In this work we emphasise local map accuracy over global consistency. Though they are not mutually exclusive, we argue that local surface reconstruction accuracy is, in many robotics applications, just as important as achieving global consistency, one of the main objectives of SLAM. Dense monocular mapping with a fisheye camera is a challenging problem, however, as our experimental results show locally accurate reconstructions are possible.

## 6.2 Dense Monocular Fisheye SLAM

The hybrid system described in Chapter 5 consists of 6 main elements: $(i)$ neural network-based metric depth prediction; $(ii)$ sparse feature-based tracking with ORBSLAM-3 providing estimates of the initial camera pose; $(iii)$ refinement of the initial sparse pose estimate through direct whole image alignment against the dense map ; $(iv)$ loop closure detections from ORB-SLAM's pipeline to bootstrap a deformation graph which is applied to reflect the loop closure in the dense map; $(v)$ local, model-to-model loop closures in the dense map reactivate inactive surfels used for data-association and camera tracking; and $(vi)$ fusion of the current live camera frame and depth prediction into the dense model. At the foundation of this SLAM system is the pinhole camera model limited to narrow FOV cameras. In this chapter, we go beyond this initial system, developing a novel system that takes advantage of large FOV fisheye imagery. The architecture of the proposed system is summarised as follows:

1. **Monocular Depth Estimation**: PackNet (Guizilini et al., 2020) is trained to predict metrically-scaled depth maps for fisheye cameras. Metric scaling is achieved by training the network with a weak velocity supervision on the camera pose. At inference time the velocity is not required. The network is used to infer metric scale depth maps using only the current live RGB image as input.

2. **Sparse Camera Tracking**: ORB-SLAM3 is used to provide the camera pose in each frame where we use the live RGB for monocular camera tracking. Internally ORB-SLAM uses the Kannala-Brandt fisheye model to process fisheye images without undistorting them.

3. **Dense Surface Fusion**: The ORB-SLAM3 estimated camera pose is used to fuse the current RGB and depth information into a dense surfel-based map, similar to the map representation of Whelan et al. (2015). Our fusion operation incorporates the Kannala-Brandt fisheye camera model.

4. **Loop Closure Correction**: Global and local loop closures are identified and used to

correct the dense surface's geometry, thereby reducing the effects of accumulated drift, and allowing the system to reuse old parts of the map for mapping.

Our full pipeline is visualised in Figure 6.2

### 6.2.1 Kannala-Brandt Camera Model

The model of image formation that has been most widely used in SLAM has been the pinhole camera model based on a perspective projection and minor distortion correction. However, in order to induce such a large receptive field, and project all the light within it onto a flat sensor plane, fisheye lenses introduce distortions which are not well captured by this model. This distortion of the image is relative to a hypothesised rectilinear image in which straight lines in the world are projected to straight lines in the image. To understand the limits of the pinhole camera model, note that pinhole projection can be written as $r = f \tan\theta$, where $r$ is the distance of the projected point from the camera's principal point and $\theta$ is the angle of incidence of the incoming ray with respect to the camera's principal axis. It is the case that $\lim_{\theta \to (\pi/2)+\pi k} \tan\theta$ does not exist. In simple terms, it is impossible to project points with large $\theta$ onto a finite image plane. As noted above, a large number of camera models more suited to capturing the distorting effects of fisheye lenses have been proposed (Kumar et al., 2023). Deciding which of the available fisheye models is optimum for dense fisheye SLAM, although an important open research question, is beyond the scope of this thesis. Instead, in this work we opt to use the Kannala-Brandt model. The Kannala-Brandt camera model is used by both ORB-SLAM3, which is a dependency of our system, and OpenCV, which provides an accurate fisheye calibration procedure.

We use the 8-parameter Kannala-Brandt camera similar to OpenCV's implementation and the implementation used in ORB-SLAM3. The first 4 parameters of the model are the focal lengths in the $x$ and $y$ direction $f_x$ and $f_y$ and the principal point $(c_x, c_y)$. The final 4 parameters are the coefficients of a $5^{th}$ degree polynomial for correcting the point of projection as a function of angle of incidence $(\kappa_0, ..., \kappa_3)$. In particular, the projection

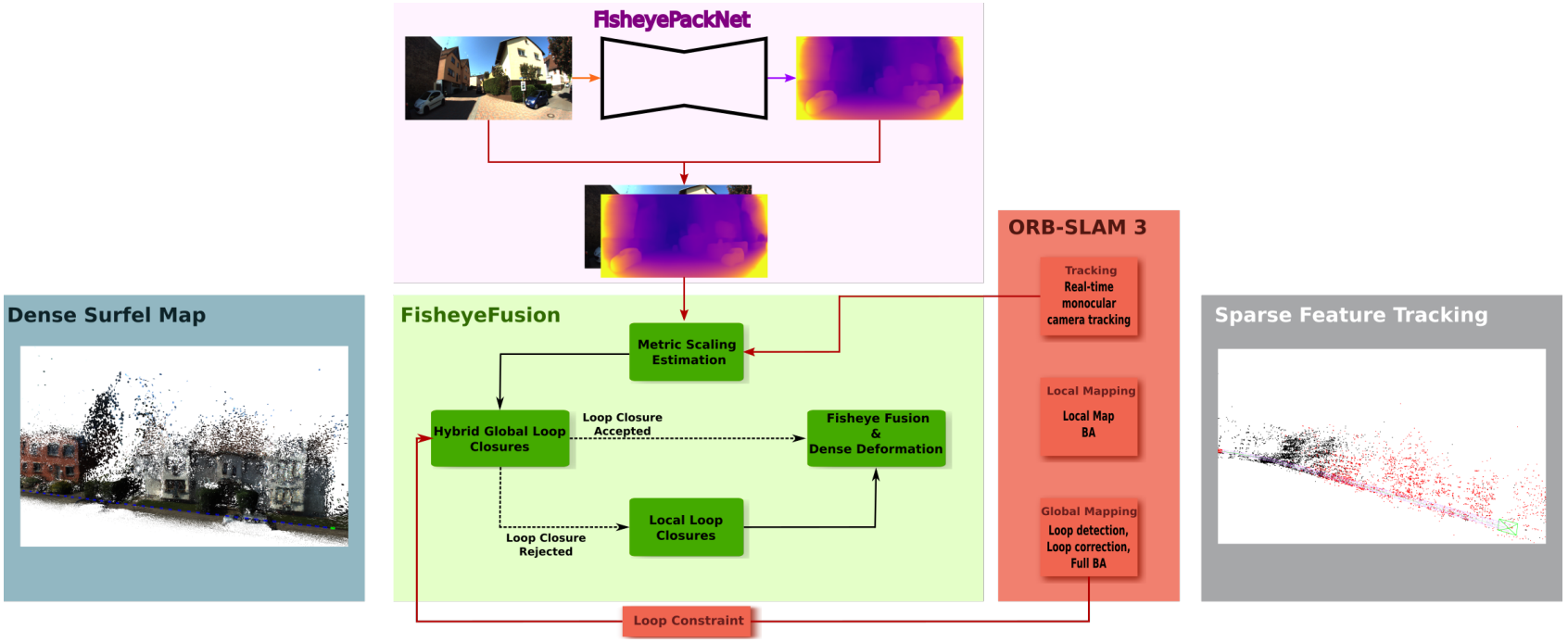



Figure 6.2: **An overview of our system's architecture.** PackNet is used to estimate a depth map for the current live camera image. ORB-SLAM then tracks the motion of the camera. ORB-SLAM continues to extract keyframes and pass them to its local and global mapping threads. The sparse feature map on the right is used internally by ORB-SLAM. Assuming no local or global loop closures have occurred in this time-step, the live RGB image and predicted depth are fused into the surfel model. If ORB-SLAM has detected a global loop closure, then a global deformation is attempted. Surface-surface constraints for optimisation of the deformation graph are generated from the pre-corrected and corrected (post loop closure) pose of the camera $\mathbf{P}_{kf}^{t}$ and $\hat{\mathbf{P}}_{kf}^{t}$ respectively. If deformation graph optimisation fails or no global loop is detected by ORB-SLAM a local loop closure is attempted as per the original EF algorithm.

function is given by,

$$\mathbf{x}_i = \pi\left(\mathbf{x}_w\right) = \begin{bmatrix} f_x \cdot r\left(\theta\right) \cdot cos\left(\psi\right) + c_x \\ f_y \cdot r(\theta) \cdot sin(\psi) + c_y \end{bmatrix} \tag{6.1}$$

where,

$$\theta = atan2\left(\sqrt{X^2 + Y^2}, Z\right)$$

$$\psi = atan2\left(Y, X\right)$$

$$r(\theta) = \theta + \kappa_0\theta^3 + \kappa_1\theta^5 + \kappa_2\theta^7 + \kappa_3\theta^9$$

here, $\mathbf{x}_w = (X, Y, Z) \in \mathbb{R}^3$ denotes a $3D$ point in camera coordinates, with the corresponding image point given by $\mathbf{x}_i \in \mathbb{R}^2 \subset \phi$, where $\phi$ is the image domain. To re-project an image point to world coordinates, the function $\pi$ can be inverted. The inverse of the mapping from Cartesian coordinates to pixel coordinates (i.e the application of $f_x$, $f_y$, $c_x$, $c_y$ in Equation (6.1) above) and from ray view direction to Cartesian coordinates ($\left(cos\left(\psi\right), sin\left(\psi\right)\right)$ in Equation (6.1) above) is straightforward to compute. The radial distortion component can be inverted by solving for the roots of $r(\theta)$ using Newton's method. See Kannala and Brandt (2006) for more details.

In this chapter, we aim to adapt the hybrid system from Chapter 5 to support fisheye cameras. As noted above ORB-SLAM3 already supports the Kannala-Brandt fisheye camera model, ElasticFusion, however, does not. In this chapter we are interested in ElasticFusion's map building functionality, i.e. we do not make use of, and therefore do not adapt, ElasticFusion's camera tracking functionality (see Section 6.2.3).

ElasticFusion's mapping functionality is implemented using OpenGL's shader pipeline. Since a core assumption of ElasticFusion is that the input camera can be modelled by the pinhole camera model, pinhole projection and unprojection code is diffuse, with reimplementations at multiple sites throughout the mapping shaders, resulting in a tight

coupling between ElasticFusion and the pinhole camera model. Our approach to adapting ElasticFusion to support fisheye cameras is to first make the map building pipeline generic with respect to the specific camera model being used, and then, within this genericised pipeline, implement the Kannala-Brandt camera model. To do this, we first centralised all pinhole projection/unprojection operations. We then abstracted the specific camera model implementation in the fusion process by introducing an interface that exposes camera model agnostic projection and unprojection functions. The fusion process then depends functionally on this interface and not on the specific camera model being used. The requirements of the projection function are, $proj : \mathbf{X}_w \rightarrow \mathbf{x}_i$ for homogeneous $3D$ world point $\mathbf{X}_w \in \mathbb{R}^3$ and homogeneous image point $\mathbf{x}_i \in \Omega \subset \mathbb{R}^2$, and an unprojection function $unproj : \mathbf{x}_i \rightarrow \mathbf{X}_w$. Note that $\mathbf{X}_w$ is specified in the camera's coordinate frame. With this interface, it is possible to easily extend the dense map building pipeline to support additional camera models without having to adapt the mapping shaders themselves. For example, the system now supports switching between mapping with the pinhole camera model and mapping with the Kannala-Brandt camera model. In the remainder of this chapter we focus on using the Kannala-Brandt camera model.

### 6.2.2 Scale-Aware Fisheye Depth Estimation

For depth estimation, we propose a variant of PackNet (Guizilini et al., 2020). The network is composed of two sub networks; a depth estimation network $\mathbf{F}_d : I \rightarrow \frac{1}{d(I)}$ (that is, the network outputs an inverse depth map) and a pose estimation network $\mathbf{F}_P : (I_t, I_{\mathcal{N}}) \rightarrow \mathbf{P}_{t \rightarrow n}$ that predicts the pose of each frame $I_n$ in the frame of reference of $I_t$. We use a lightweight version of the depth network, containing fewer packing and unpacking layers, that is more suited to real-time operation. Both networks are trained simultaneously in a self-supervised manner by establishing a structure-from-motion loss function over a small temporal window around the current training frame. To do so, the pixels of each context frame $I_n \in \mathcal{N}(I_t)$ are warped into the target frame $I_t$, producing a set of warped frames $\hat{I}_n$. The basis of the self-supervised loss function is photometric error induced between the target frame and each of the warped frames. We reformulate the multi-view warping

function of PackNet to employ the Kannala-Brandt camera model, allowing the network to estimate dense depth maps for fisheye camera imagery. The function to warp an image point $\mathbf{x}_n$ in a context frame to its corresponding location $\mathbf{x}_t$ in the target frame is given by

$$\mathbf{x}_t = \pi\left(\mathbf{P}_n^t \cdot \left[d_n \cdot \pi^{-1}\left(\mathbf{x}_n\right)\right]\right) \tag{6.2}$$

where, $\pi$ is the Kannala-Brandt projection model from Section 6.2.1, $\mathbf{P}_n^t$ is the rigid body pose from context frame $I_n$ to target frame $I_t$ and $d_n^{-1}$ is the inverse depth estimate for image point $n$. Note that for simplicity we have left out homogenisation and de-homogenisation operations.

We take additional measures to enforce metric scale depth estimates and to improve overall training and inference performance. Firstly, we apply weak velocity supervision loss on the pose network. Secondly, we perform sparse LiDAR supervision. For example, when using the KITTI-360 dataset, we render the LiDAR and SICK laser scanner point clouds into each fisheye frame using the ground-truth camera poses. This produces a set of sparse ground-truth depth maps which we use to supervise the training. The full loss function is

$$\mathcal{L} = \mathcal{L}_p + \lambda_1 \mathcal{L}_{reg} + \lambda_2 \mathcal{L}_{vel} + \lambda_3 \mathcal{L}_{depth} \tag{6.3}$$

where, $\mathcal{L}_p$, $\mathcal{L}_{reg}$ and $\mathcal{L}_{vel}$ are, respectively, the photometric loss, depth smoothness loss and velocity supervision loss as per the original PackNet paper (Guizilini et al., 2020). $\mathcal{L}_{depth}$ is an $L_1$ loss over ground-truth and predicted depth estimates (see Guizilini et al. (2019) for more details).

### 6.2.3 Hybrid Sparse-Dense Mapping With Kannala-Brandt Camera Model

To compute a dense model of the environment, we extend the hybrid sparse-dense SLAM system of Gallagher et al. (2021). ORB-SLAM3 is used for camera tracking and global

loop closure detection (Campos et al., 2021). To achieve dense mapping, Gallagher et al. (2021) combines ORB-SLAM3 with ElasticFusion (Whelan et al., 2015). As in ElasticFusion, the dense map is represented as an unordered list of surfels $\mathcal{M}$ where each surfel consists of a position $\mathbf{p} \in \mathbb{R}^3$, a normal $\mathbf{n} \in \mathbb{R}^3$, a radius $r \in \mathbb{R}$, a confidence $c \in \mathbb{R}$ and a colour. As before, at each point in time, the map is divided into two non-overlapping subsets of surfels; an active map $\Theta$ representing surfels that have been fused with live data within the last $\delta_t$ iterations and; an inactive map $\Phi$ representing surfels that have not been fused with live camera data within the last $\delta_t$ iterations. For each live colour image $I_t$ our system runs a number of steps detailed below.

**Depth estimation**. The system computes a depth map $\mathcal{D}_t = F_d(I_t)$ using the depth estimation network described in Section 6.2.2.

**Sparse camera tracking.** $I_t$ is passed to ORBSLAM-3 which performs feature-based camera tracking as per the original ORB-SLAM algorithm (Campos et al., 2021). Note that we run the full ORB-SLAM pipeline in the background including camera tracking, local mapping and loop closure detection and global optimisation. The output of this phase is the pose of the camera for the current time-step $\mathbf{P}_t \in SE(3)$.

**Global Loop Closures** We close global loops in a two-stage hybrid process similar to Gallagher et al. (2021). ORB-SLAM3 extracts novel and informative frames as keyframes, using a background thread to search for loop closures between keyframes. Once a loop closure candidate is found, it is geometrically verified and a constraint $\mathbf{T}_{K_i}^{K_j} \in SE(3)$ between the two triggering keyframes is calculated. ORB-SLAM corrects its sparse map with a pose graph optimisation followed by a bundle adjustment. To correct the dense map, our system renders the dense model into two virtual frames located at the same poses as the two triggering keyframes. The pose of the newly added keyframe $K_i$ is used to render the active portion of the model $\Theta$, while the pose of the matched keyframe $K_j$, transformed by the loop closing constraint $\mathbf{T}_{K_i}^{K_j}$, is used to render the inactive portion of the model $\Phi$. This ensures that it is the newer, active portion of the model that is brought into registration with the old part of the model during loop correction. From these two frames, a set of $3D \rightarrow 3D$ surface-surface correspondences are calculated. The surface correspondences

are used to optimise a deformation graph which, when applied to the dense model, corrects its geometry to reflect the loop closure.

**Local Loop Closures**. Assuming no global loop closure has occurred during the current iteration, the system looks for a local loop closure within the dense model. A local loop closure seeks to align the active and inactive portions of the model in view of the current camera pose. Local loop closures help keep the camera aligned with the dense map during locally loopy camera motion. To do so, we render two views of the scene, one each for the active and inactive map regions, using $\mathbf{P}_t$. The loop closure constraint is calculated via a robust ICP non-linear optimisation, as per the original EF algorithm. The result is a 6-DOF rigid body transformation $\mathbf{T}_a^i$ aligning the active frame with the inactive frame. Since local loops are purely model-to-model, instead of using $\pi$ to project $\Theta$ and $\Phi$ into $\mathbf{P}_t$, we render instead to virtual pinhole images. This allows us to take advantage of the EF's RGB-D alignment. To correct the dense geometry, $\mathbf{T}_a^i$ is used to compute a set of $3D \rightarrow 3D$ surface-surface point correspondences, in the same way as during a global loop closure. The loop is then corrected with a deformation graph in the same manner as a global loop closure.

**Fusion**. Once the camera has been tracked and all loop closures have been identified, the current live camera frame $I_t$ and estimated depth map $\mathcal{D}_t$ are fused into the active region of the map $\Theta$. To do so, data-associations between $\Theta$ and $I_t$ are found by rendering a high resolution index map of $\Theta$ using $\mathbf{P}_t$ and $\pi$. Once the current frame has been associated, fusion proceeds in the same way as in Whelan et al. (2015).

### 6.2.4 Accounting for Scale

The overall scale of a scene is unobservable by a single monocular camera with no prior information. Therefore, a monocular mapping system can only estimate the motion of the camera and the geometry of the scene up to a scale factor. What's more, the scale of the reconstruction tends to drift over time. For long and loopy mapping sessions, scale consistency can be enforced by observing and correcting scale drift during loop closures (Strasdat et al., 2010a). In our system, before we can use $\mathbf{P}_t$ to fuse the current frame and

depth map, we must recover the scale ratio $s \in \mathbb{R}$ between the sparse reconstruction and the dense reconstruction. We estimate the scale ratio between the metric dense depth estimate at the current time step $\mathcal{D}_t$ and the sparse ORB-SLAM map points $\mathcal{M}_{ORB}$ in view of the current pose $\mathbf{P}_t$. First, we render a sparse depth map $\mathcal{D}_{ORB}$ from $\mathcal{M}_{ORB}$ using $\mathbf{P}_t$. The scale difference at time step $t$ is given by median ratio $s_t = \frac{med(\mathcal{D}_t)}{med(\mathcal{D}_{orb})}$. To ameliorate noise and to account for the fact that scale drifts over time, we keep a rolling buffer of scale estimates over the last $n$ frames $\mathbf{S}$. The final scale ratio $s \in \mathbb{R}$ estimate is then taken as the average over $\{s_i \in \mathbf{S} | s_i \in \left[\mu_\mathbf{S} \pm 2\sigma_\mathbf{S}\right]\}$. Once the scale ratio $s$ has been recovered we use it to scale the translational component of the current pose, lifting it into metric scale and allowing fusion of the current frame into the metric scale dense model.

## 6.3 Experimental Results

To investigate the suitability of our proposed approach to dense monocular fisheye SLAM, we performed a qualitative and quantitative analysis on the KITTI-360 benchmark dataset (Liao et al., 2021). KITTI-360 is a follow-on to the well known KITTI benchmark dataset (Geiger et al., 2013). The dataset consists of 9 public access sequences of a car driving through a sub-urban environment where the car is equipped with two lateral facing fisheye cameras pointing in opposite directions. The sequences also include 3D LiDAR and a 2D SICK laser scanner. Dataset annotations include ground-truth poses, per-frame LiDAR and SICK point-clouds, accumulated point-clouds and calibration parameters. Semantic labels, GPS/IMU data, and bounding boxes are also included, however we do not make use of these additional annotations in this work.

Note that KITTI-360 provides a different camera model for the fisheye cameras to the one used in our SLAM system. To utilise KITTI-360 with our system we calibrate the fisheye cameras using the chessboard detector of Geiger et al. (2012b) and the fisheye calibration optimisation of OpenCV (Bradski, 2000). An overview of our system's inputs and outputs can be seen in Figure 6.1 and Figure 6.3, respectively.

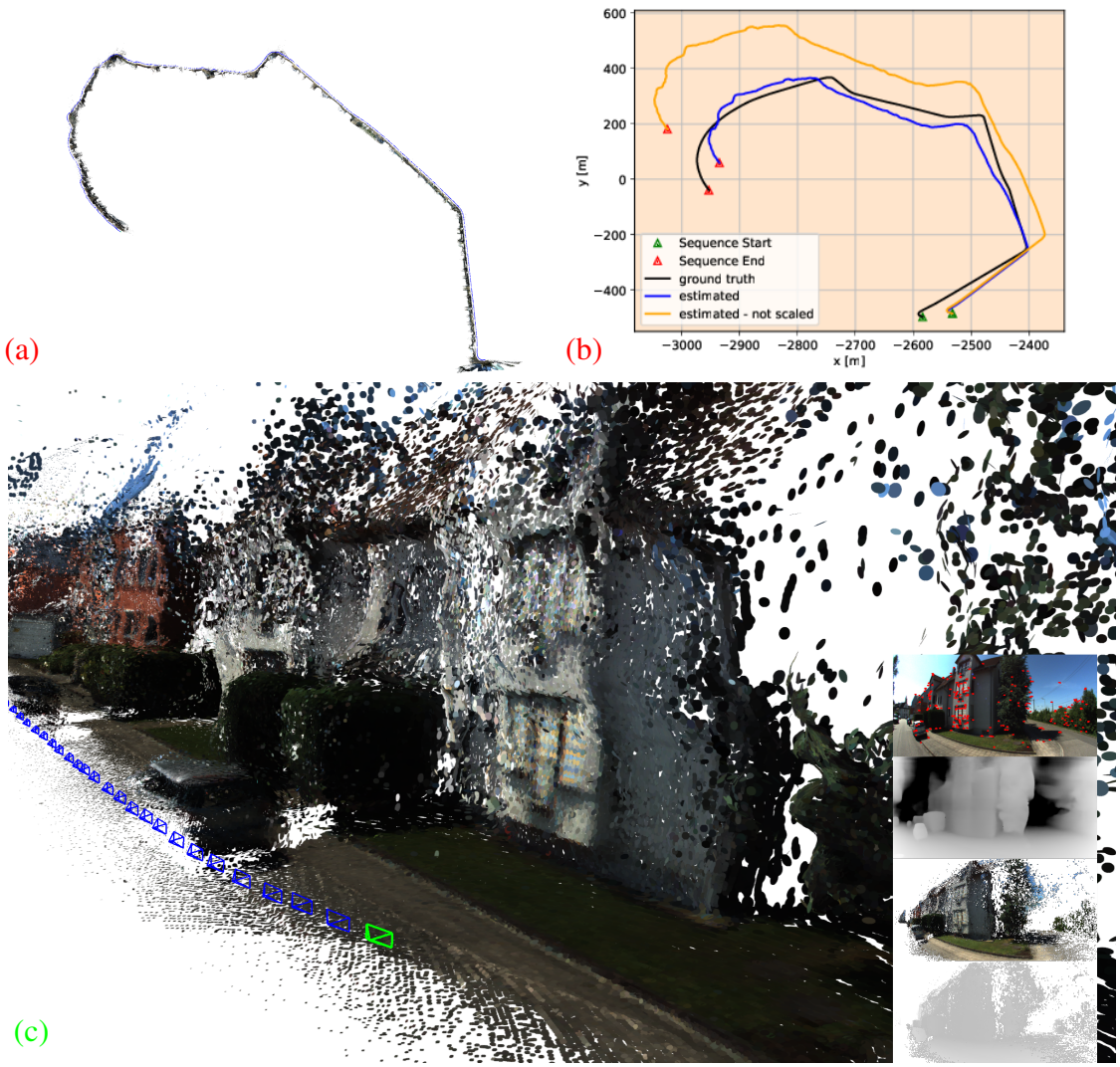



Figure 6.3: **Trajectory and dense reconstruction of a sequence from Liao et al. (2021).** (a) shows the final surfel model produced by our system. The model contains approximately $7.5m$ surfels. (b) shows the estimated trajectory (projected onto the $xy$ plane) alongside the ground truth trajectory. (c) shows an up-close view of the reconstruction as the vehicle passes through an intersection contained in (a). Inset in (c), from top to bottom: the current live image, the corresponding estimated depth map from the network, the model predicted RGB image and the model predicted depth map. Note that these results were captured using an accurate high resolution version of our network that allows our SLAM system to operate at approximately $2hz$. We trained a slim version of the network for real-time operation.

### 6.3.1 Depth Estimation Training Results

We trained Fisheye PackNet described in Section 6.2.2 on the fisheye images of the KITTI-360 dataset. Each of the 9 KITTI-360 sequences vary in length, ranging from $\sim 1k$ frames to $\sim 20k$ frames. We use both left and right images during training but treat them as

| **Method** | $\text{Abs}_{rel}$ | $\text{Sq}_{rel}$ | RMSE | $\text{RMSE}_{log}$ |
|---|---|---|---|---|
| PackNet Fisheye | 0.127 | 0.605 | 2.422 | 0.204 |

Table 6.1: **Evaluation of depth estimation on the proposed KITTI-360 test split.** Results are computed over depth estimates up to $80m$.

separate samples. For training, we end up with $\sim 45k$ training images, $\sim 45k$ testing images and $\sim 20k$ validation images. Sparse depth map training labels are produced in a pre-processing step by rendering the LiDAR point clouds and SICK point clouds into the ground truth camera frame. We train the network on a GPU server with 4 Nvidia RTX-3090 cards for 66 epochs. Each epoch takes approximately 1hrs40mins. In Table 6.1 we establish a first baseline of depth estimation results on KITTI-360 fisheye, reporting the accuracy of the trained depth estimation on the KITTI-360 test split with respect to the ground truth LiDAR scans. In Figure 6.4 we report qualitative results of our network's performance on the test split of KITTI-360.

### 6.3.2 Dense Fisheye Mapping Results

In this section, we present the results of experiments that investigate the effectiveness, at a local level, of our hybrid approach to dense monocular fisheye SLAM in estimating de-noised surface reconstructions. We also provide a number of qualitative examples of the system's overall performance, including side-by-side comparison of estimated and ground truth trajectories (e.g., Figures 6.5 and 6.6).

All Mapping experiments are run on an i7 CPU using an NVIDIA 1080ti using the test splits outlined in Section 6.3.1. Within the testing split of each sequence, we use chunks of contiguous frames for benchmarking our system. In our experiments, we demonstrate that, while global consistency can be difficult to achieve, our system still consistently achieves accurate local mapping over windows up to $> 300m$ in size. In Figure 6.5 we report surface reconstruction accuracy results. In particular, for a given test sequence, we retrieve nearby ground truth LiDAR scans to construct a local model. We then align this local model to the model estimated by our system. We take the mean distance between each point in our model and its corresponding closest point in the aligned

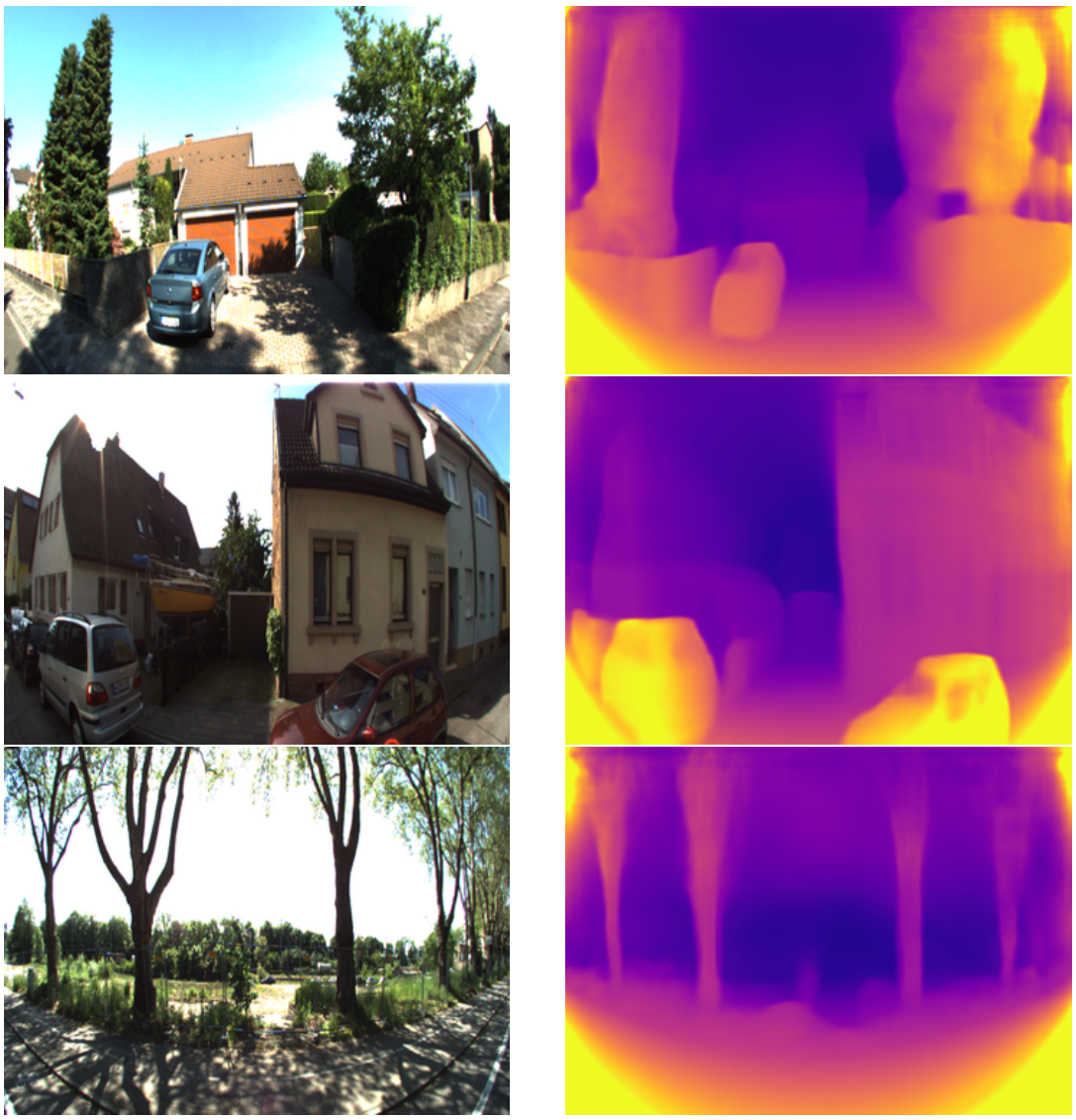
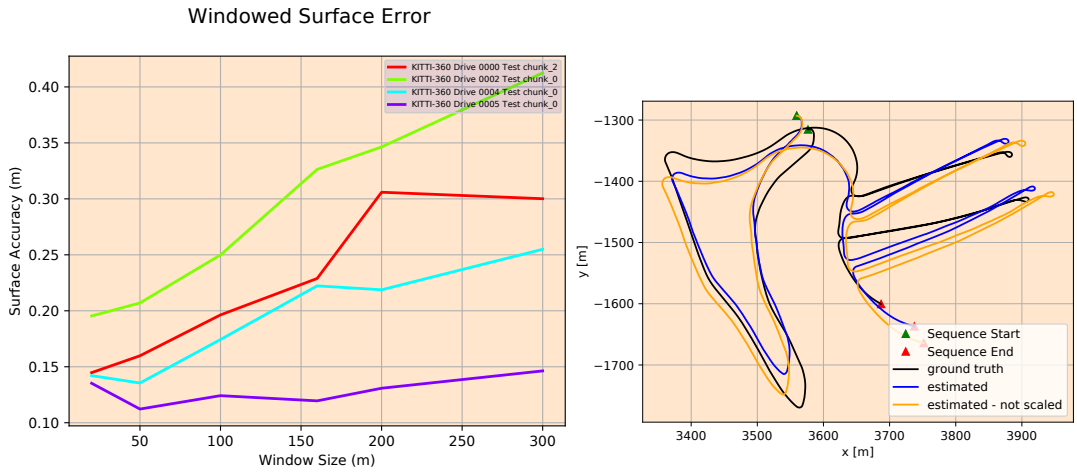


Figure 6.4: **Qualitative results of Fisheye PackNet on test split of KITTI-360.** The left column shows the input RGB fisheye image, the right column shows the estimated depth maps. Top row; note that while the gross geometry of the scene is well estimated, finer grained details, on the garage door and on the rear of the car, are smoothed out. Middle row; the prominent building corner highlights instabilities in depth estimation of sharp details in the scene. Bottom row; foliage represents a particularly challenging input for the network.

local model as a measure of the local mapping accuracy of our system. Figure 6.5(a) shows how incrementally increasing the neighbourhood region from $10-300m$ within which LiDAR scans are gathered results in local mapping accuracy of $\sim 13cm - \sim 42cm$. In Figure 6.5(b) we show how estimating scale is challenging for our system given the use of monocular ORBSLAM and the scale ratio (see Section 6.2.4). Figure 6.5(b) shows the trajectory estimated by our system (lifted into metric scale during mapping) alongside




(a) (b)

Figure 6.5: **Windowed dense surface reconstruction accuracy from test splits from Liao et al. (2021).** Figure 6.5(a) shows the mean surface-surface error as a function of local window size. For each of the sequences in figure 6.5(a), we average the results across 5 uniformly selected locations from trajectory. Figure 6.5(b) shows our system's estimated trajectory alongside the ground truth trajectory for KITTI-360 sequence 4. The blue curve shows the estimated trajectory aligned and scaled to the ground truth. Note how, even though tracking exhibits significant scale drift, within relative large local areas ($\sim 300m$) mapping is accurate.

the ground truth trajectory. We note that despite the exhibited scale drift, the overall morphology of the trajectory is captured in the output, and as shown in Figure 6.5(a), the resulting map remains highly accurate over ranges of $150m$. In Figure 6.7 we report a breakdown of our system's run-time performance. The system runs at $\sim 5hz$. Note that as the number of surfels increases, the time taken to fuse new frames into the dense model also increases. Depth estimation is the main bottleneck in the system. Figure 6.6 qualitatively demonstrates loop closures in our system.

## 6.4 Summary

We presented a monocular fisheye dense mapping system that combines dense depth prediction with sparse feature tracking and dense surfel fusion techniques. Fisheye depth prediction is achieved by integrating the Kannala-Brandt camera model into PackNet. With a combination of weak velocity supervision and sparse LiDAR-based depth supervision, we trained PackNet to predict dense depth maps on the fisheye images of the KITTI-360

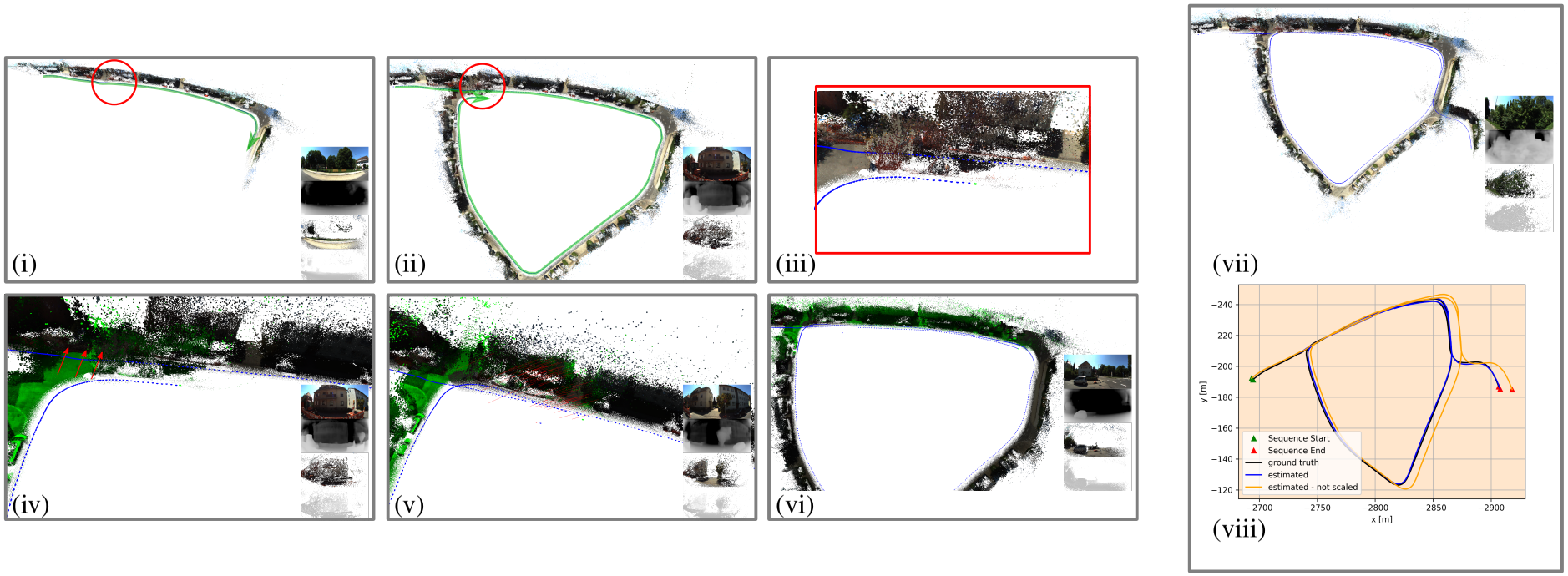



Figure 6.6: **A qualitative example of our system's performance**. $(i)$ The system begins building a dense map as the vehicle explores (green arrow). $(ii)$ The vehicle loops back on itself (red circle). $(iii)$ Error accumulated during the loop results in drift. $(iv)$ The active portion of the model (green) needs to be brought back into alignment with the old portion of the model (greyed out) from the vehicle's first pass. $(v)$ A loop closure is detected and a deformation is applied to the dense surface, correcting for the effects of drift. Red lines denote surface-surface deformation constraints. $(vi)$ Mapping continues; local loop closures reactivate the old part of the map from the first pass which can now be used for mapping. $(vii)$ The final global model. $(viii)$ The estimated and ground truth trajectories. The sequence used in this example corresponds to a sub-sequence from KITTI-360 sequence 9 Liao et al. (2021).

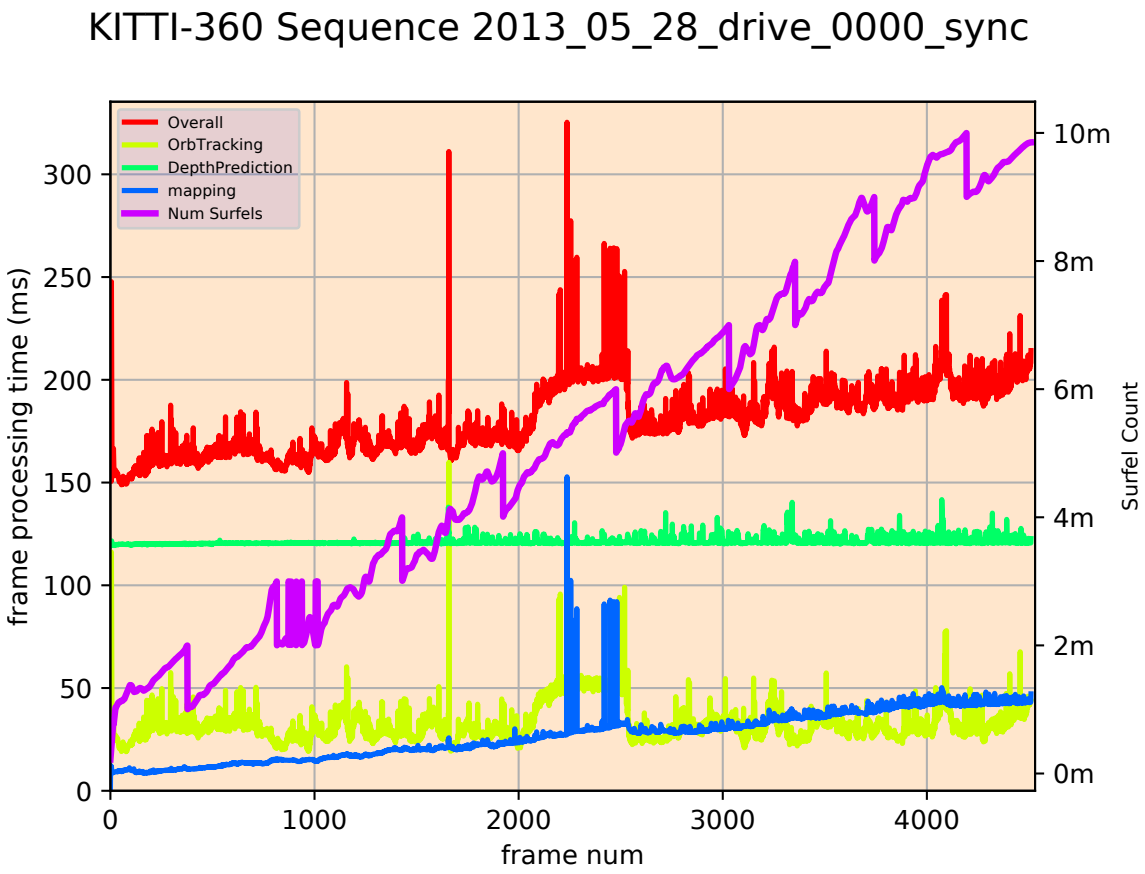



Figure 6.7: **A breakdown of our system's run-time performance.** We used a test split subsequence of KITTI-360 Sequence 00 for this experiment. The subsequence consists of ~ 4500 frames.

dataset. The proposed SLAM system permits live-dense reconstruction of outdoor scenes using a fisheye camera in automotive scenarios. Sparse tracking provides camera pose estimation capable of operating robustly at vehicle speeds and in outdoor environments with variable lighting. Global loop closures are identified by ORB-SLAM3's appearance-based place recognition module with the resultant constraints passed to the dense fusion algorithm where they are integrated with a deformation graph-based map correction step

Our results demonstrate dense, metric visual mapping with a single fisheye camera over local windows of up to $300m$, with accuracy on the order of $13-42$cm, depending on window size. The integration of loop closure constraints within a SLAM based approach permit leveraging of repeated passes of regions of the environment within a globally consistent framework. Although our system exhibits accurate mapping over extended scales, globally accurate mapping remains a challenge. Within the context of KITTI-360, the lateral, sideways facing positioning of the fisheye cameras, combined with the fast

motion of the vehicle, represents a difficult input for ORB-SLAM3. Furthermore, while we have incorporated measures to address scale ambiguity inherent in monocular SLAM, it remains a significant challenge over long sequences where the scale of the sparse map is bound to drift

In KITTI-360, the sideways facing orientation of the fisheye cameras on the vehicle means that the focus of expansion is to the image edge. This is in contrast to the forward facing pinhole cameras, where the focus of expansion is towards the image centre. The effect of the orientation of the camera, in relation to its direction of travel, on camera tracking and map fusion, especially in the context of wide FOV fisheye cameras, is an interesting question for future work.

Supporting a wide range of camera models within a single mapping system is a challenging problem. Projection and re-projection functions can vary drastically between different camera models. Often the Jacobian of the camera model is required for integrating the model into non-linear camera tracking and mapping optimisations. Furthermore, each camera model requires calibration, which itself can involve a complicated procedure that may fail or produce a poor calibration. This limits a SLAM systems suitability as a research and consumer grade tool. In future work, we aim to investigate Neural Ray Surfaces (Vasiljevic et al., 2020) and their ability to recover a ray surface for a camera without known calibration parameters, as the basis for a unified approach to dense, CNN-based mapping.

Another aspect of future work is to redesign the fisheye tracking function to leverage the dense model, similar to how it is leveraged by the hybrid tracking procedure of Section 5.2.4. This would involve reformulating the Jacobians of the direct tracking cost function in terms of the Kannala-Brandt camera model similar to (Matsuki et al., 2018).

# CHAPTER 7

# Conclusion

## 7.1 Discussion

The aim of this thesis was to bridge the gap between the state-of-the art in real-time dense $3D$ reconstruction and what is required for operation in the challenging conditions of outdoor environments and multi-camera collaborative mapping. Application to the automotive domain acted as representative context for our work, where several limitations have prevented SoTA dense VSLAM systems from being applied. In Chapter 3 we introduced a novel dense collaborative SLAM system that allowed multiple independently moving hand-held RGBD cameras to collaboratively map an indoor space. We identified two distinct phases of operation during collaborative mapping; an initial phase where cameras maintain local maps independently of one another, and a multi-camera fusion phase where cameras whose local maps overlap have been merged. We used fern-based visual recognition to identify overlap between camera specific local maps and ICP to find the transformations between the coordinate frames of the matched local maps, allowing processing to enter the multi-camera fusion phase of operation (for cameras in these two maps at least). The process of matching views and merging maps continues until either processing ends or there is only one global map left.

A significant limiting factor of collaborative mapping is the number of cameras that can be processed simultaneously. To address this issue, in Chapter 4, we explored the potential for switching between tracking and full scale SLAM processing using an

information theoretic keyframing approach. The approach estimates the quantity of novel information contained in live sensor data with respect to the dense map. Frames whose novelty exceeds a threshold are used for both mapping and tracking, while frames considered redundant are used to track the camera and are then discarded. By putting a threshold on how novel a frame needs to be before it is fused, this method exposes a tunable parameter that allows the complexity of the underlying surfel map to be controlled on a per-application basis. This level of control is an important aspect of next generation robust and versatile SLAM systems (Cadena et al., 2016).

Dense mapping is challenging in outdoor environments, especially when the camera is attached to a moving vehicle. The fast motion of the vehicle results in large displacement between successive frames which is antithetical to what is required for direct camera tracking. In general, what is required for direct camera tracking is a good initial estimate of the camera pose in the current frame. Initializing the optimisation at this estimated pose allows the tracking procedure to avoid local minima in the non-convex error surface defined by the tracking cost function.

Furthermore, environmental factors, like changing weather, lead to highly variable light conditions, making the estimation of dense scene geometry using classical multi-view reconstruction techniques, such as those used by DTAM (Newcombe et al., 2011b) and by active RGB-D sensors like the Kinect, extremely challenging. Underlying many approaches to direct camera tracking is the *brightness constancy constraint*. This constraint captures the assumption that the same point (with Lambertian reflectance properties) imaged by two cameras from two different perspectives, results in the same pixel brightness value in both cameras (albeit at different pixel sites). Of course, this constraint takes an overly simplistic view of things, even when applied to the most idealistic real-world scenes. In outdoor SLAM, all it takes is for a cloud to pass in front of the sun during the time between the capture of successive frames to alter the intensity of projected points. However, when combined with robust cost functions, it has been shown that the constraint only needs to be approximately correct in order to provide accurate camera tracking performance.

In the latter half of this thesis, we described a dense mapping system that overcomes

these challenging conditions to reconstruct outdoor spaces from a moving camera, in real time. ElasticFusion formed the foundation of this system, providing direct camera tracking and dense surface fusion capabilities. To overcome the challenges of large inter-frame displacement induced by the fast motion of the car we incorporate elements of ORB-SLAM, a sparse SLAM system. As opposed to ElasticFusion, which performs direct camera tracking, ORB-SLAM finds correspondences between highly salient feature points in successive frames. This feature-based approach to tracking makes ORB-SLAM far less susceptible to large inter-frame displacement and transient changes in scene illumination. The hybrid design of this system brings into question a potentially false dichotomy in SLAM research. Platinsky et al. (2017) ask the following question with the title of their paper - “Sparse joint optimisation or dense alternation?" However, the SLAM system of Chapter 5 represents a third option - that the strengths of both sparse and dense SLAM can be combined within one system.

In Chapter 6 we extended our hybrid system to operate on large FOV fisheye cameras. The highly distorted nature of fisheye images makes them challenging inputs for a SLAM system to process and cannot be modelled by the pinhole camera model that underlies the hybrid SLAM system. We reformulated the depth prediction and dense surface fusion stages of the pipeline in terms of the Kannala-Brandt camera model, allowing the system to process images in their original distorted fisheye form.

Taken as a whole, the contributions of this thesis have addressed key limitations of the state-of-the-art in dense VSLAM. Prior to this work, these limitations prevented techniques from this field from being applied to multi-camera collaborative mapping and outdoors environments, where they may be of potentially enormous benefit. Table 7.1 lists artefacts and video demonstrations created during the production of this thesis.

## 7.2 Future Work

The main task of future research in this area is to fully integrate the collaborative and outdoor SLAM systems of this thesis. In Appendix A, we demonstrate, with a simulated urban canyon dataset, that the collaborative SLAM system presented in Chapter 3 could, in

| Artefact | Link |
|---|---|
| Open source code for Chapters 3 and 5 | https://github.com/robotvisionmu/DenseMonoSLAM |
| Chapter 3 demo | https://youtu.be/GUtHrKEM85M |
| Chapter 3 - urban canyon demo | https://youtu.be/Uy7wYHbdPV8 |
| Chapter 5 demo | https://youtu.be/Pn2uaVqjskY |
| Chapter 6 demo | https://youtu.be/EU7WYRuklJo |

Table 7.1: **A list of artefacts and demonstrations created during the research of this thesis.**

principle, facilitate multi-camera mapping with outdoors autonomous vehicles. However, before the technique can be applied in the real world, the hybrid sparse-dense SLAM architectures, presented in Chapters 5 and 6, would need to be integrated, in order for the system to overcome the challenges of outdoor automotive scenarios. One of the main technical challenges that needs to be addressed in order to achieve this integration is that ORBSLAM-3, which is a dependency of both the hybrid sparse-dense system of Chapter 5 and the fisheye system of Chapter 6, is itself not a collaborative SLAM system. Extending it in this way would require careful engineering.

While this thesis, with the above-mentioned integration, would go some way to bringing the potential of collaborative dense SLAM to autonomous vehicles, there remains a wide chasm between the capabilities these SLAM systems provide and what will ultimately be required to build truly autonomous multi-robot systems. Traversing that chasm requires addressing other key problems.

The question of relaxing the static scene assumption is one such key problem and a promising direction for future research. As discussed earlier in the thesis, this assumption forms part of the limit of what many SLAM systems can handle. Some works try to identify dynamic scene elements in the input stream so they can be down weighted or removed from the optimisation (Keller et al., 2013). More interesting though is how they can be captured and included in the model. Newcombe et al. (2015) uses a surface warp-field to warp the current (dynamically deformed) frame to its canonical form as it appeared in the

first frame. While this system is both dense and real-time it has not demonstrated dynamic mapping of large complex scenes.

Are the models dense SLAM systems produce truly dense, or at least as dense as they could be? In many instances the resultant models contain many 'holes'; areas that through occlusion, noise or lack of input data have not been mapped. However, across many categories of objects, and even in general scenes, there are potentially strong priors that can be leveraged to fill the holes with plausible guesses of what form the unseen surface might take. There are many works on in-painting $3D$ models of objects to fill in missing data. However, exactly how these networks and their predictions can be placed in the loop of a SLAM system is a largely unexplored question.

Neural networks are a flexible computing tool with potentially many benefits for SLAM. As predictors, they can be used to estimate useful quantities for a SLAM system such as visual odometry, depth frames, uncertainty, motion segmentation, semantic segmentation and much more besides. As encoders, they can be used to represent both whole scenes and individual keyframes. The breakthrough NeRF system (Mildenhall et al., 2020) achieved state-of-the-art novel view synthesis performance. Since then several follow-on works have extended the system to operate like a full Structure From Motion system (Lin et al., 2021; Wang et al., 2021), and even a full SLAM system (Sucar et al., 2021; Zhu et al., 2022). The issue of scalability surrounding NeRFs has begun to be addressed. Frameworks that allow neural-network based representations, like NeRF, to be trained in a matter of seconds have been proposed (Müller et al., 2022). Other works extend classic NeRF architectures to cover multiple city blocks (Tancik et al., 2022).

NeRF represents the scene implicitly using a multilayer perceptron (MLP). The MLP maps input $3D$ query points to the emitted radiance at each point in space. Colour and depth predictions from specific viewpoints can be made by querying the NeRF at points sampled from along the camera's lines of sight (i.e. the direction rays emanating from each pixel site in the image array). The resulting radiances from each ray can then be combined in a volumetric rendering step. This representation is extremely flexible. When it comes to SLAM, NeRF style representations offer a route to truly dense $3D$ models, without the

discretisation artefacts of signed distance functions and occupancy grids. During mapping one can operate at potentially any scale or resolution. The MLP representation allows for plausible interpolation of unseen portions of the scene (Sucar et al., 2021).

Systems such as Kimera (Rosinol et al., 2021), SemanticFusion (McCormac et al., 2016) and Fusion++ (McCormac et al., 2018) estimate semantic labels for each patch of the surface. Many depth estimation networks can be augmented and trained to estimate semantics for each pixel site in the input images. These observations can then be integrated into the map to produce dense $3D$ semantic map as is done by SemanticFusion. However, the question of how to leverage such a semantic map is a particularly promising avenue for future research that has not been fully explored. It seems that, in principle, the relationship between model predicted semantic labels and the semantic labels of the input image can be leveraged in the underlying SLAM optimisation, as a form of regularisation or bias, to guide the optimsation to the correct structure and motion parameters.

It is our hope that the work of this thesis can act as a strong foundation for computing a geometric base layer upon which semantic and truly dense neural representations of challenging, dynamic scenes can be built.

# Appendices

# APPENDIX A

# Urban Canyon Simulation

In this appendix, we provide qualitative results of our approach to collaborative SLAM applied to the domain of autonomous driving within a simulated urban environment. ElasticFusion, and therefore our system, was designed with a hand-held camera exploring an indoor environment in mind. Neither system can be directly applied to outdoors environments and scenarios that involve fast camera motion due to two primary limitations: (i) ElasticFusion requires that the input device is an RGB-D camera. While this is not a limitation in indoors environments, when operating outdoors, natural light causes and increase in sensor noise beyond what ElasticFusion is capable of tolerating; (ii) ElasticFusion's tracking procedure assumes that inter-frame displacement is relatively small and that scene lighting is consistent. This assumption is violated in the domain of outdoor, automotive applications where the camera will be moving at speed and lighting is highly variable. For the experiments in this appendix, we extend the Urban Canyon dataset (Zhang et al., 2016), creating a series of overlapping camera trajectories, where each trajectory consists of a sequence of RGB-D frames rendered from a simulation of a vehicle traversing synthetic model of a city. During rendering, sensor noise, scene lighting and inter-frame displacement are tightly controlled allowing us to demonstrate that our collaborative SLAM system can be applied to automotive applications if ElasticFusion's underlying limitations can be addressed. In Chapters 5 and 6 we revisited ElasticFusion's limitations, and presented dense SLAM systems that can operate outdoors and with cameras attached to moving vehicles.

For this experiment, we establish a simulated multi-vehicle automotive scenario in an urban environment based on the Urban Canyon dataset (Zhang et al., 2016). Urban Canyon consists of a neighbourhood-sized section of city and includes a number of roads meeting at intersections and surrounded by buildings. The scene is generated within Blender using the SceneCity add-on. The original dataset simulates a single forward-looking camera attached to a moving vehicle. For our experiments, we wish to simulate multiple vehicles moving through the city. To do so, we split the original trajectory of the vehicle, defined using splines in Blender, into two overlapping sub-trajectories. We further reverse the direction of one of the sub-trajectories. This ensures that there is ample opportunity for inter-map loop closures, but that computing the inter-map constraint is non-trivial as the vehicles have significantly different viewing directions. We then render depth and colour images for each sub-trajectory. During rendering, we ensure that frames are sampled close enough so that inter-frame displacement is suitably small and that scene lighting remains consistent for the duration of the render. We further note that depth maps are not corrupted by noise due to natural light. The camera is modelled as a pinhole camera and all sub-sequences are rendered with Blender's Cycle ray-tracing engine as per the original Urban Canyon dataset (Zhang et al., 2016). The collaborative Urban Canyon dataset can be seen in Figure A.1

To demonstrate our system's applicability to automotive scenarios, we process the collaborative Urban Canyon dataset in a two-vehicle collaborative mapping session. The results of this collaborative mapping session are detailed in Figure A.2 and its caption. The system allows two vehicles, starting in separate local sub-maps, to successfully find and correct an inter-map loop closure and build a single shared model of a neighbourhood-sized section of a city. These results demonstrate that if scene lighting and inter-frame displacement can be accounted for, and accurate depth maps can be computed then our system can be used in automotive scenarios.

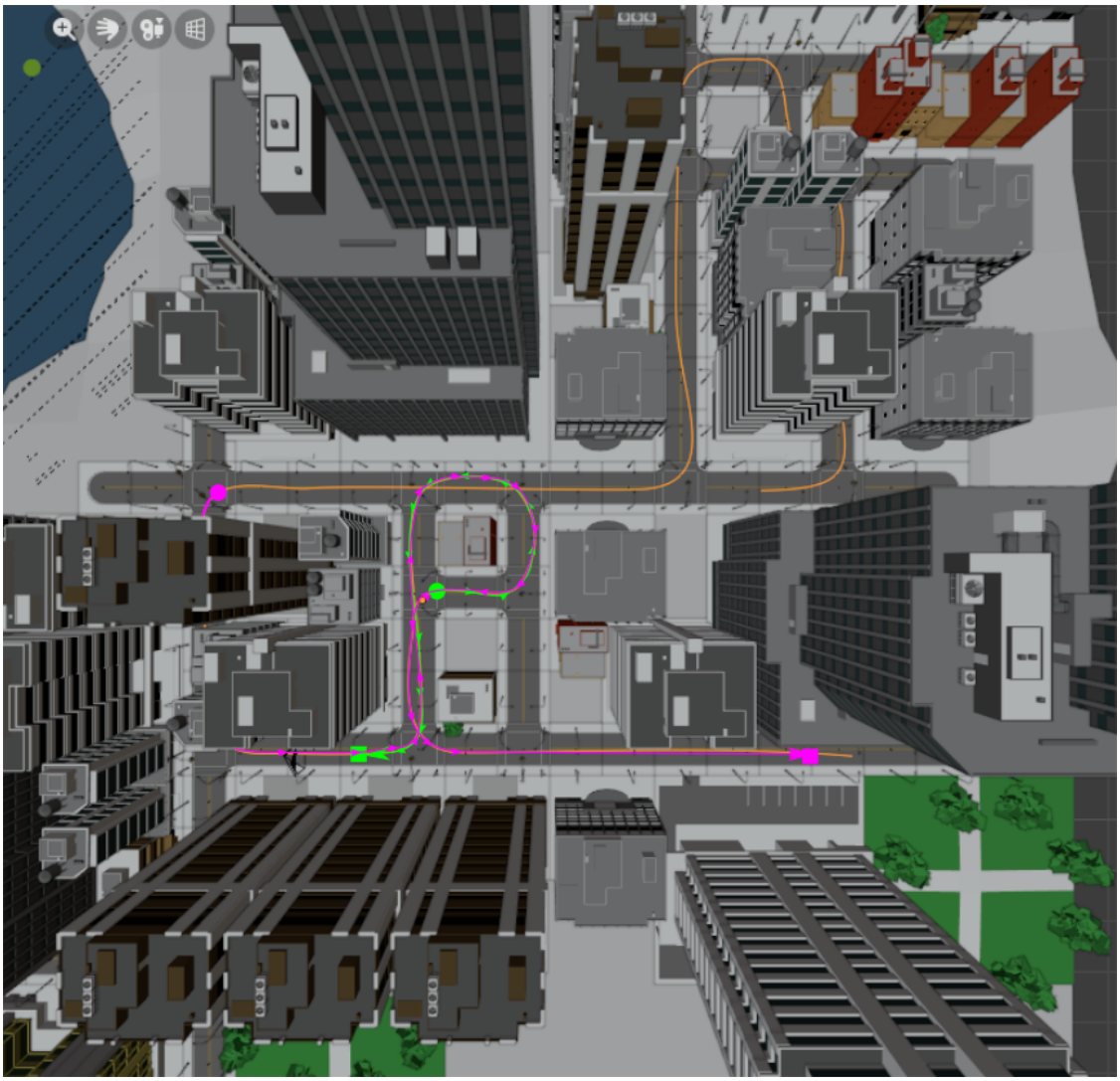


Figure A.1: **Collaborative urban canyon dataset.** The orange curve describes full trajectory of the car through the synthetic city scene as described in Zhang et al. (2016). For collaborative experiments, we divide this trajectory into 2 main segments overlaid in purple and green. Arrows along the curves point in the direction of travel of the car. Note that the two sequences overlap substantially but that the car travels in opposite directions in each of them.

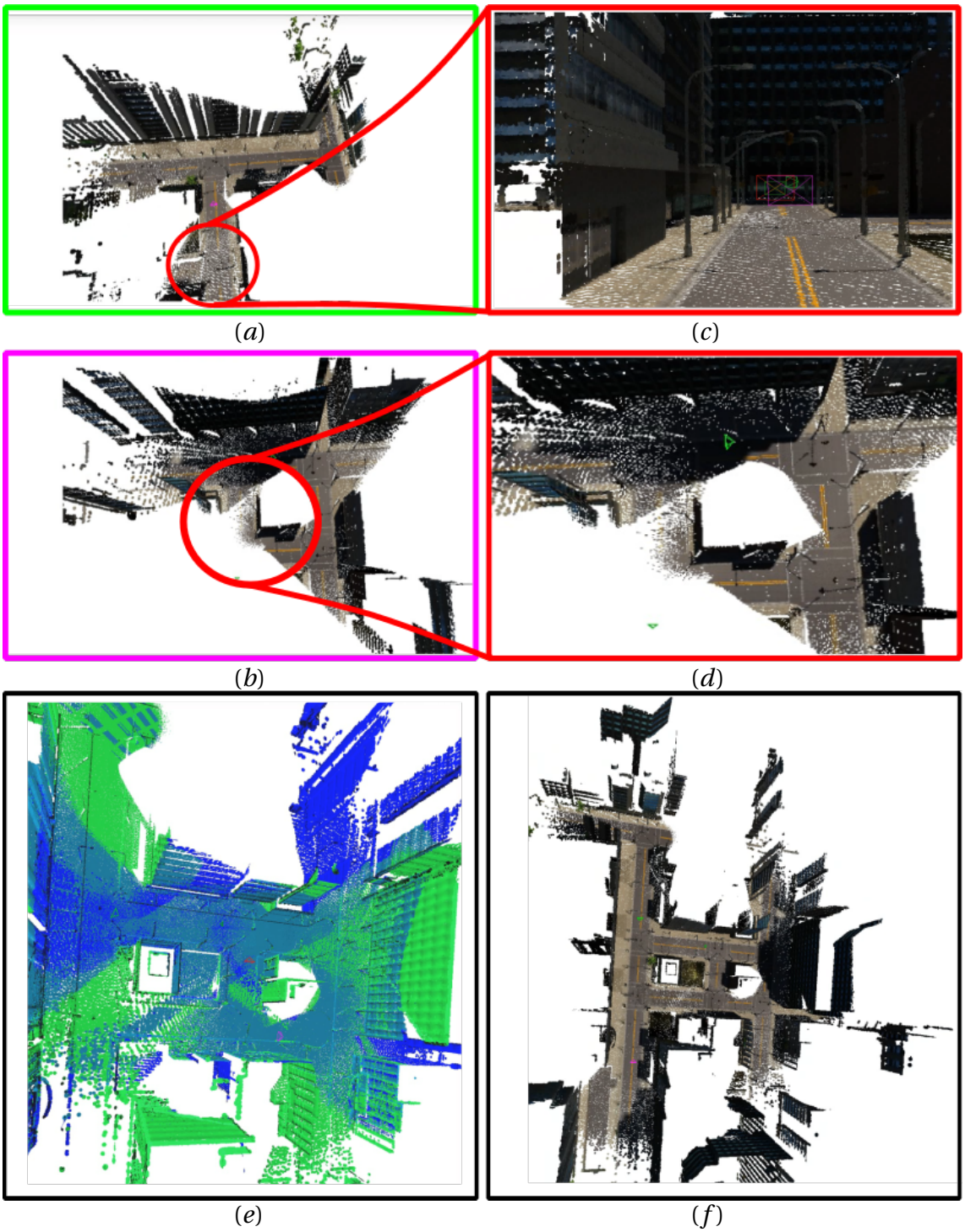



Figure A.2: **Collaborative dense SLAM with synthetic urban canyon dataset.** (*a*) and (*b*) two cameras, corresponding to the two sequences outlined in Chapter A, begin mapping within their own relative local maps. (*c*) and (*d*) visual place recognition across the two sub-maps triggers an inter-map loop closure, bringing the two sub-maps into alignment. (*e*) the two maps are merged into a single global coordinate frame. Each camera can now use the map data produced by the other map. Primary coloured green and blue surfels correspond to surfels that have been measured by a single camera, while the teal coloured surfels have been fused with surfels from both cameras, signifying map sharing. (*f*) final global model produced collaboratively in the urban canyon environment.

# Bibliography

J. Abel and J. Chaffee. Existence and uniqueness of GPS solutions. *IEEE Transactions on Aerospace and Electronic Systems*, 27(6):952–956, 1991. doi: 10.1109/7.104271. 126

J. Bares and D. Wettergreen. Dante II: Technical Description, Results, and Lessons Learned. *The International Journal of Robotics Research*, 18(7):621–649, 1999. doi: 10.1177/02783649922066475. 9

J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 141

C. Bishop. *Pattern Recognition and Machine Learning, 5th Edition*. Information science and statistics. Springer, 2007. ISBN 9780387310732. 109

G. Blais and M. D. Levine. Registering Multiview Range Data to Create 3D Computer Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8): 820–824, August 1995. ISSN 0162-8828. doi: 10.1109/34.400574. 82

M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. Davison. CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 55, 128, 129

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 158

G. Bresson, Z. Alsayed, L. Yu, and S. Glaser. Simultaneous Localization and Mapping: A

Survey of Current Trends in Autonomous Driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017. doi: 10.1109/TIV.2017.2749181. 16

A. Bruss and B. Horn. Passive Navigation. *Computer Vision, Graphics, and Image Processing*, 21(1):3–20, 1983. ISSN 0734-189X. doi: 10.1016/S0734-189X(83)80026-7. 30

M. Buehler, K. Iagnemma, and S. Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 3642039901. 15

C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. doi: 10.1109/TRO.2016.2624754. 3, 6, 19, 23, 46, 71, 167

C. Campos, R. Elvira, J. Rodríguez, J. Montiel, and J. Tardós. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. doi: 10.1109/TRO.2021.3075644. 12, 29, 41, 69, 73, 124, 128, 130, 134, 136, 148, 156

D. Caruso, J. Engel, and D. Cremers. Large-Scale Direct SLAM for Omnidirectional Cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 141–148, 2015. doi: 10.1109/IROS.2015.7353366. 147

R. Castle, G. Klein, and D. Murray. Video-Rate Localization in Multiple Maps for Wearable Augmented Reality. In *2008 12th IEEE International Symposium on Wearable Computers*, pages 15–22, 2008. doi: 10.1109/ISWC.2008.4911577. 65, 66

J. Cheng, L. Zhang, Q. Chen, X. Hu, and J. Cai. A Review of Visual SLAM Methods for Autonomous Driving Vehicles. *Engineering Applications of Artificial Intelligence*, 114: 104992, 2022. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2022.104992. 16

M. Chli and A. Davison. Active Matching. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 72–85, 2008. ISBN 978-3-540-88681-5. doi: 10.1007/978-3-540-88682-2_7. 110

G. Costante, M. Mancini, P. Valigi, and T. Ciarfuglia. Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation. *IEEE Robotics and Automation Letters*, 1(1):18–25, 2016. doi: 10.1109/LRA.2015.2505717. 128

Z. Cui, L. Heng, Y. Yeo, A. Geiger, M. Pollefeys, and T. Sattler. Real-Time Dense Mapping for Self-Driving Vehicles using Fisheye Cameras. *2019 International Conference on Robotics and Automation (ICRA)*, pages 6087–6093, 2019. 148

B. Curless. *New Methods for Surface Reconstruction from Range Images*. PhD thesis, Stanford, Stanford, CA, USA, 1998. 44

B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 303–312, 1996. ISBN 0897917464. doi: 10.1145/237170.237269. 44

J. Czarnowski, T. Laidlow, R. Clark, and A. Davison. DeepFactors: Real-Time Probabilistic Dense Monocular SLAM. *IEEE Robotics and Automation Letters*, 5:721–728, 2020. doi: 10.1109/lra.2020.2965415. 55, 129

A. Dahal, V. R. Kumar, S. Yogamani, and C. Eising. An online learning system for wireless charging alignment using surround-view fisheye cameras. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):20553–20562, 2022. doi: 10.1109/TITS.2022.3182165. 17

A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics 2017 (TOG)*, 2017. doi: 10.1145/3054739. 3, 13, 52, 72, 99, 106, 124, 125

T. Dang, C. Hoffmann, and C. Stiller. Continuous Stereo Self-Calibration by Camera Parameter Tracking. *IEEE Transactions on Image Processing*, 18(7):1536–1550, 2009. doi: 10.1109/TIP.2009.2017824. 126

DARPA. The DARPA Grand Challenge: Ten Years Later, 2014. URL https://www.darpa.mil/news-events/2014-03-13. 15

A. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. doi: 10.1109/TPAMI.2007.1049. 4, 12, 32, 75, 128

A. J. Davison. Real-Time Simultaneous Localisation and Mapping With a Single Camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1403–1410 vol.2, 2003. doi: 10.1109/ICCV.2003.1238654. 2, 32

A. J. Davison. Active Search for Real-Time Vision. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 66–73 Vol. 1, October 2005. doi: 10.1109/ICCV.2005.29. 110, 111

F. Dellaert. Square Root SAM. In *Robotics: Science and Systems*, 2005. 2, 27, 29

R. Dubois, A. Eudes, J. Moras, and V. Frémont. Dense Decentralized Multi-Robot SLAM Based on Locally Consistent TSDF Submaps. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4862–4869, 2020. doi: 10.1109/IROS45743.2020.9341680. 70, 71, 72

E. Eade and T. Drummond. Unified Loop Closing and Recovery for Real Time Monocular SLAM. In *In British Machine Vision Conference*, 2008. 128

D. Eigen, C. Puhrsch, and R. Fergus. Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2366–2374. MIT Press, 2014. 46, 129

R. A. El-laithy, J. Huang, and M. Yeh. Study on the use of Microsoft Kinect for Robotics Applications. In *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pages 1280–1288, 2012. doi: 10.1109/PLANS.2012.6236985. 125

R. Elvira, J. Tardós, and J. Montiel. ORBSLAM-Atlas: A Robust and Accurate Multi-Map System. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6253–6259, 2019. doi: 10.1109/IROS40897.2019.8967572. 69

J. Engel, J. Sturm, and D. Cremers. Semi-Dense Visual Odometry for a Monocular Camera. In *2013 IEEE International Conference on Computer Vision*, pages 1449–1456, 2013. doi: 10.1109/ICCV.2013.183. 38, 128

J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014. 13, 38, 124, 128, 147

J. Engel, V. Koltun, and D. Cremers. Direct Sparse Odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017. 41, 124, 128, 148

C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff. The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles. *IEEE Wireless Communications*, 23(4):146–152, 2016. doi: 10.1109/MWC.2016.7553038. 15

R. Eustice, H. Singh, and J. Leonard. Exactly Sparse Delayed-State Filters for View-Based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114, 2006. doi: 10.1109/TRO.2006.886264. 28

M. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake. Continuous Humanoid Locomotion Over Uneven Terrain Using Stereo Fusion. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 881–888, 2015. doi: 10.1109/HUMANOIDS.2015.7363465. 125

P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart. Kinect V2 for Mobile Robot Navigation: Evaluation and Modeling. In *2015 International*

*Conference on Advanced Robotics (ICAR)*, pages 388–394, 2015. doi: 10.1109/ICAR.2015.7251485. 125

J. Fenwick. *Collaborative Concurrent Mapping and Localization*. PhD thesis, Massachusetts Institute of Technology, 2001. 60

J. Fenwick, P. Newman, and J. J. Leonard. Cooperative Concurrent Mapping and Localization. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA*, 2002. 60, 66

M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24 (6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. 34

C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza. Collaborative Monocular SLAM With Multiple Micro Aerial Vehicles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3962–3970, 2013. doi: 10.1109/IROS.2013.6696923. 67

C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014. doi: 10.1109/ICRA.2014.6906584. 41, 124, 128

C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017. doi: 10.1109/TRO.2016.2623335. 124, 128

D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed Multirobot Exploration and Mapping. *Proceedings of the IEEE*, 94(7):1325–1339, 2006. doi: 10.1109/JPROC.2006.876927. 19, 59

U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. G. Herrtwich. Making Bertha See. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 214–221, 2013. doi: 10.1109/ICCVW.2013.36. 16

U. Frese. Interview: Is SLAM Solved? *KI - Künstliche Intelligenz*, 24:255–257, 2010. 3

P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Mühlfellner, S. Wonneberger, J. Timpner, S. Rottmann, B. Li, B. Schmidt, T. N. Nguyen, E. Cardarelli, S. Cattani, S. Brüning, S. Horstmann, M. Stellmacher, H. Mielenz, K. Köser, M. Beermann, C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, R. Iser, R. Triebel, I. Posner, P. Newman, L. Wolf, M. Pollefeys, S. Brosig, J. Effertz, C. Pradalier, and R. Siegwart. Toward Automated Driving in Cities Using Close-To-Market Sensors: An Overview of the V-Charge Project. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 809–816, 2013. doi: 10.1109/IVS.2013.6629566. 15

L. Gallagher and J. McDonald. Collaborative Dense SLAM. In *Proceedings of the 2018 IPRCS, Irish Machine Vision and Image Processing Conference*, 2018. 20

L. Gallagher and J. McDonald. Efficient Surfel Fusion Using Normalised Information Distance. In *Computer Vision and Pattern Recognition (CVPR) workshop on* 3*D Scene Understanding for Vision, Graphics, and Robotics*, 2019. 21

L. Gallagher, V. Kumar, S. Yogamani, and J. McDonald. A Hybrid Sparse-Dense Monocular SLAM System for Autonomous Driving. In *2021 European Conference on Mobile Robots (ECMR)*, pages 1–8, 2021. doi: 10.1109/ECMR50962.2021.9568797. 21, 131, 155, 156

L. Gallagher, G. Sistu, J. Horgan, and J. B. McDonald. A system for dense monocular mapping with a fisheye camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 6478–6486, June 2023. 21

D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. doi: 10.1109/CVPR.2007.383245. 46

D. Galvez-López and J. Tardós. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012. doi: 10.1109/TRO.2012.2197158. 134

X. Gao, R. Wang, N. Demmel, and D. Cremers. LDSO: Direct Sparse Odometry with Loop Closure. In *IROS*, Oct. 2018. 128

R. Garg, V. Bg, G. Carneiro, and I. Reid. Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–756. Springer, 2016. 132

A. Geiger, P. Lenz, and R. Urtasun. Are we Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012a. ix, 16, 126, 127, 139, 140, 143

A. Geiger, F. Moosmann, O. Car, and B. Schuster. Automatic calibration of range and camera sensors using a single shot. In *International Conference on Robotics and Automation (ICRA)*, 2012b. 158

A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision Meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013. 4, 126, 140, 158

C. Geyer and K. Daniilidis. A Unifying Theory for Central Panoramic Systems and Practical Applications. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, ECCV '00, pages 445–461, 2000. ISBN 3540676864. 147

B. Glocker, J. Shotton, A. Criminisi, and S. Izadi. Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding. *IEEE Transactions on Visualization and Computer Graphics*, 21(5):571–583, May 2015. ISSN 1077-2626. doi: 10.1109/TVCG.2014.2360403. 84

C. Godard, O. Mac Aodha, and G. Brostow. Unsupervised Monocular Depth Estimation With Left-Right Consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 129

S. Golodetz, T. Cavallari, N. Lord, V. Prisacariu, D. Murray, and P. Torr. Collaborative Large-Scale Dense 3D Reconstruction with Online Inter-Agent Pose Optimisation. *IEEE Transactions on Visualization and Computer Graphics*, 24:2895–2905, 2018. 69, 70, 71, 72

H. Grimmett, M. Buerki, L. Paz, P. Pinies, P. Furgale, I. Posner, and P. Newman. Integrating Metric and Semantic Maps for Vision-Only Automated Parking. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2159–2166, 2015. doi: 10.1109/ICRA.2015.7139484. 16

V. Guizilini, J. Li, R. Ambrus, S. Pillai, and A. Gaidon. Robust Semi-Supervised Monocular Depth Estimation with Reprojected Distances. In *Proceedings of 3rd Annual Conference on Robot Learning (CoRL)*, volume 100, pages 503–512. PMLR, 2019. 155

V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon. 3D Packing for Self-Supervised Monocular Depth Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 16, 129, 146, 149, 150, 154, 155

Q. Guo, I. Frosio, O. Gallo, T. Zickler, and J. Kautz. Tackling 3D ToF Artifacts Through Learning and the FLAT Dataset. In *European Conference on Computer Vision*, pages 381–396, 2018. doi: 10.1007/978-3-030-01246-5_23. 126

A. Handa, R. Newcombe, A. Angeli, and A. Davison. Real-Time Camera Tracking: When is High Frame-Rate Best? In *Computer Vision – ECCV 2012*, pages 222–235, 2012. 127

A. Handa, T. Whelan, J. McDonald, and A. Davison. A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014. ix, 4, 74, 99, 108, 114, 119

C. G. Harris and J. M. Pike. 3D Positional Integration from Image Sequences. *Image Vision Comput.*, 6(2):87–90, May 1988. ISSN 0262-8856. doi: 10.1016/0262-8856(88)90003-0. 32

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. ISBN 0521540518. 10, 26, 78

S. W. Hasinoff. *Photon, Poisson Noise*, pages 608–610. Springer US, Boston, MA, 2014. ISBN 978-0-387-31439-6. doi: 10.1007/978-0-387-31439-6_482. 127

K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.322. 54

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments. *The International Journal of Robotics Research*, 31(5):647–663, Apr. 2012. ISSN 0278-3649. doi: 10.1177/0278364911434148. 3, 13, 46, 72

H. Hoppe. *Surface Reconstruction from Unorganized Points*. PhD thesis, June 1994. PhD Thesis. 43

H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface Reconstruction From Unorganized Points. In *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, pages 71–78, 1992. 42, 43

B. K. P. Horn. Closed-Form Solution of Absolute Orientation Using Unit Quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, April 1987. 40

A. Howard. Multi-robot Simultaneous Localization and Mapping Using Particle Filters. *The International Journal of Robotics Research*, 25(12):1243–1256, Dec. 2006. ISSN 0278-3649. doi: 10.1177/0278364906072250. 62, 66

A. Howard, G. S. Sukhatme, and M. J. Mataric. Multirobot Simultaneous Localization and Mapping Using Manifold Representations. *Proceedings of the IEEE*, 94(7):1360–1369, July 2006. ISSN 0018-9219. doi: 10.1109/JPROC.2006.876922. 63, 64, 66

A. S. Huang, E. Olson, and D. C. Moore. LCM: Lightweight Communications and

Marshalling. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4057–4062, October 2010. doi: 10.1109/IROS.2010.5649358. 88, 90

C. Hughes, M. Glavin, E. Jones, and P. Denny. Wide-angle Camera Technology for Automotive Applications: A Review. *Iet Intelligent Transport Systems*, 3:19–31, 2009. 7

M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. VolumeDeform: Real-Time Volumetric Non-rigid Reconstruction. In *Computer Vision – ECCV 2016*, pages 362–379, 2016. 53

S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, 2011. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047270. 14, 50, 51, 53

J. Janai, F. Güney, A. Behl, and A. Geiger. Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art. *Found. Trends. Comput. Graph. Vis.*, 12(1–3): 1–308, jul 2020. ISSN 1572-2740. doi: 10.1561/0600000079. 16

S. Julier and J. Uhlmann. A Counter Example to the Theory of Simultaneous Localization and Map Building. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, volume 4, pages 4238–4243 vol.4, 2001. doi: 10.1109/ROBOT.2001.933280. 28

M. Kaess and F. Dellaert. Visual SLAM with a Multi-Camera Rig. 2006. 29

M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental Smoothing and Mapping. *IEEE Trans. on Robotics (TRO)*, 24(6):1365–1378, Dec. 2008. 29, 50, 65

O. Kahler, V. Prisacariu, C. Ren, X. Sun, P. Torr, and D. Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1241–1250, Nov. 2015. ISSN 1077-2626. doi: 10.1109/TVCG.2015.2459891. 107

J. Kannala and S. Brandt. A Generic Camera Model and Calibration Method for Conventional, Wide-angle, and Fish-eye Lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006. doi: 10.1109/TPAMI.2006.153. 153

N. Katayama and S. Satoh. The SR-tree: An Index Structure for High-dimensional Nearest Neighbor Queries. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, SIGMOD '97, pages 369–380, 1997. ISBN 0-89791-911-4. doi: 10.1145/253260.253347. 61

M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion. In *Proceedings of the 2013 International Conference on 3D Vision*, 3DV '13, pages 1–8, 2013. ISBN 978-0-7695-5067-1. doi: 10.1109/3DV.2013.9. 3, 14, 42, 46, 50, 51, 53, 169

C. Kerl, J. Sturm, and D. Cremers. Dense Visual SLAM for RGB-D Cameras. In *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013. 14, 42

A. Kim and R. Eustice. Real-Time Visual SLAM for Autonomous Underwater Hull Inspection Using Visual Saliency. *IEEE Transactions on Robotics*, 29(3):719–733, 2013. doi: 10.1109/TRO.2012.2235699. 126

B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller. Multiple Relative Pose Graphs for Robust Cooperative Mapping. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3185–3192. IEEE, 2010. 65, 66, 68, 69

G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nov. 2007. 2, 4, 11, 12, 13, 29, 33, 128

G. Klein and D. Murray. Improving the Agility of Keyframe-Based SLAM. In *Computer Vision – ECCV 2008*, pages 802–815, 2008. 2, 34

V. Kumar, C. Eising, C. Witt, and S. Yogamani. Surround-View Fisheye Camera Perception for Automated Driving: Overview, Survey & Challenges. *IEEE Transactions on*

*Intelligent Transportation Systems*, pages 1–22, 2023. doi: 10.1109/TITS.2023.3235057. 151

M. Labbé and F. Michaud. Online Global Loop Closure Detection for Large-scale Multi-session Graph-based SLAM. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666, Sept 2014. doi: 10.1109/IROS.2014.6942926. 68

M. Lauer. Grand cooperative driving challenge 2011 [its events]. *IEEE Intelligent Transportation Systems Magazine*, 3(3):38–40, 2011. doi: 10.1109/MITS.2011.942107. 15

J. Leonard and H. Durrant-Whyte. Simultaneous Map Building and Localization for an Autonomous Mobile Robot. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 1442–1447 vol.3, Nov 1991. doi: 10.1109/IROS.1991.174711. 2, 75

V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81:155–166, 2009. 39

S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013. doi: 10.15607/RSS.2013.IX.037. 23

S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-Based Visual–Inertial Odometry Using Nonlinear Optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. doi: 10.1177/0278364914554813. 126

Y. Liao, J. Xie, and A. Geiger. KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D. *arXiv preprint arXiv:2109.13410*, 2021. ix, 4, 16, 147, 158, 159, 162, 163

C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey. BARF: Bundle-Adjusting Neural Radiance Fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 170

C. Loop, Q. Cai, S. Orts-Escolano, and P. Chou. A Closed-Form Bayesian Fusion Equation Using Occupancy Probabilities. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 380–388, 2016. doi: 10.1109/3DV.2016.47. 46

W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, 1987. ISBN 0897912276. doi: 10.1145/37401.37422. 43, 44

D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 2004. 38

Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003. ISBN 0387008934. 79

P. C. Mahalanobis. On the Generalized Distance in Statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936. 64

M. Maimone, Y. Cheng, and L. Matthies. Two years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, 24(3):169–186, 2007. doi: https://doi.org/10.1002/rob.20184. 8, 9, 10

A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte. An Experiment in Integrated Exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 534–539 vol.1, 2002. doi: 10.1109/IRDS.2002.1041445. 15

D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982. ISBN 0716715678. 1

Z. Marton, R. Rusu, and M. Beetz. On Fast Surface Reconstruction Methods for Large and Noisy Point Clouds. In *2009 IEEE International Conference on Robotics and Automation*, pages 3218–3223, 2009. doi: 10.1109/ROBOT.2009.5152628. 53

H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler, and D. Cremers. Omnidirectional DSO: Direct Sparse Odometry With Fisheye Cameras. *IEEE Robotics and Automation Letters*, 3(4):3693–3700, 2018. doi: 10.1109/LRA.2018.2855443. 148, 165

L. Matthies and S. Shafer. Error Modeling in Stereo Navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248, 1987. doi: 10.1109/JRA.1987.1087097. 30

L. Matthies, M. Maimone, A. Johnson, Y. Cheng, R. Willson, C. Villalpando, S. Goldberg, A. Huertas, A. Stein, and A. Angelova. Computer Vision on Mars. *International Journal of Computer Vision*, 75(1):67 – 92, October 2007. 9, 10, 126

L. H. Matthies. Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-time Implementation. *Int. J. Comput. Vis.*, 8(1):71–91, 1992. doi: 10.1007/BF00126401. 30

J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger. SemanticFusion: Dense 3D Semantic Mapping With Convolutional Neural Networks. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2016. 53, 171

J. McCormac, R. Clark, M. Bloesch, A. J. Davison, and S. Leutenegger. Fusion++: Volumetric Object-Level SLAM. *2018 International Conference on 3D Vision (3DV)*, pages 32–41, 2018. 54, 171

J. McDonald. *Multi-session Visual Simultaneous Localisation and Mapping*. PhD thesis, National University of Ireland Maynooth, December 2013. 19, 23

J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. Leonard. Real-time 6-DOF Multi-Session Visual SLAM Over Large-Scale Environments. *Robotics and Autonomous Systems*, 61(10):1145–1158, 2013. 68

C. Mei, G. Sibley, M. J. Cummins, P. Newman, and I. D. Reid. RSLAM: A System for Large-Scale Mapping in Constant-Time Using Stereo. *International Journal of Computer Vision*, 94:198–214, 2010. 1, 36

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng.

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020. 170

D. Ming, X. Wu, Y. Wang, Z. Zhu, H. Ge, and R. Liu. A Real-Time Monocular Visual SLAM Based on the Bundle Adjustment with Adaptive Robust Kernel. *Journal of Intelligent & Robotic Systems*, 107, 03 2023. doi: 10.1007/s10846-023-01817-2. 16

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *AAAI/IAAI*, pages 593–598. AAAI Press / The MIT Press, 2002. 32

J. Montiel, J. Civera, and A. Davison. Unified Inverse Depth Parametrization for Monocular SLAM. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006. doi: 10.15607/RSS.2006.II.011. 33

H. Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Computer Science Department, Stanford University, 1980. 30

G. R. Mueller and H.-J. Wuensche. Continuous Extrinsic Online Calibration for Stereo Cameras. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 966–971, 2016. doi: 10.1109/IVS.2016.7535505. 126

G. R. Mueller and H.-J. Wuensche. Continuous Stereo Camera Calibration in Urban Scenarios. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017. doi: 10.1109/ITSC.2017.8317675. 126

T. Müller, A. Evans, C. Schied, and A. Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. 170

R. Mur-Artal and J. D. Tardós. Visual-Inertial Monocular SLAM With Map Reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017a. doi: 10.1109/LRA.2017.2653359. 124, 128

R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017b. doi: 10.1109/TRO.2017.2705103. 12, 14, 29, 41, 124, 128, 129, 137, 139, 142

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. doi: 10.1109/TRO.2015.2463671. 12, 16, 29, 39, 69, 124, 128

K. P. Murphy. Bayesian map learning in dynamic environments. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. 32

NASA. MER Twins. Available at https://mars.nasa.gov/imgs/general/layout/overview/MER_twins_white_cyc.png [accessed on 2022/05/31], 2022. 9

J. Neira and J. D. Tardos. Data Association in Stochastic Mapping Using the Joint Compatibility Test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, Dec 2001. ISSN 1042-296X. doi: 10.1109/70.976019. 60

E. W. Nettleton, P. W. Gibbens, and H. F. Durrant-Whyte. Closed Form Solutions to the Multiple-Platform Simultaneous Localization and Map Building (SLAM) Problem. volume 4051, pages 4051 – 4051 – 10, 2000. doi: 10.1117/12.381658. 60

R. Newcombe. *Dense visual SLAM*. PhD thesis, Imperial College London, UK, 2012. 82

R. A. Newcombe and A. J. Davison. Live Dense Reconstruction With a Single Moving Camera. In *IEEE Conference on Computer Vision and pattern Recognition*, 2010. 3, 4, 45, 72

R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time Dense Surface Mapping and Tracking. In *2011 10th IEEE International Symposium on Mixed and*

*Augmented Reality*, pages 127–136, 2011a. doi: 10.1109/ISMAR.2011.6092378. 3, 14, 42, 46, 47, 48, 50, 72, 106, 124, 125

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *2011 International Conference on Computer Vision*, pages 2320–2327, 2011b. doi: 10.1109/ICCV.2011.6126513. 4, 14, 45, 46, 72, 128, 167

R. A. Newcombe, D. Fox, and S. M. Seitz. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015. 23, 53, 169

P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, L. Murphy, W. Churchill, D. Cole, and I. Reid. Navigating, Recognizing and Describing Urban Spaces With Vision and Lasers. *The International Journal of Robotics Research*, 28(11-12):1406–1433, 2009. doi: 10.1177/0278364909341483. 28

M. Niessner, M. Zollhofer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale Using Voxel Hashing. *ACM Trans. Graph.*, 32(6):169:1–169:11, Nov. 2013. ISSN 0730-0301. doi: 10.1145/2508363.2508374. 14, 44, 107

D. A. O'Handley. Scene Analysis in Support of a Mars Rover. *Computer Graphics and Image Processing*, 2(3-4):281–297, 1973. doi: 10.1016/0146-664X(73)90007-5. 8

M. Okutomi and T. Kanade. A Multiple-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993. doi: 10.1109/34.206955. 30

C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone. Rover Navigation Using Stereo Ego-Motion. *Robotics and Autonomous Systems*, 43(4):215–229, 2003. ISSN 0921-8890. doi: 10.1016/S0921-8890(03)00004-6. 30, 31

B. O'Neill. *Elementary Differential Geometry*. Academic Press, Boston, Second edition, 2006. ISBN 978-0-12-088735-4. doi: 10.1016/B978-0-12-088735-4.50008-0. 42

G. Pascoe, W. Maddern, and P. Newman. Robust Direct Visual Localisation using Normalised Information Distance. In *Proceedings of the British Machine Vision Con-*

*ference (BMVC)*, pages 70.1–70.13, September 2015. ISBN 1-901725-53-7. doi: 10.5244/C.29.70. 111

G. Pascoe, W. Maddern, M. Tanner, P. Pinies, and P. Newman. NID-SLAM: Robust Monocular SLAM Using Normalised Information Distance. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1446–1455, July 2017. doi: 10.1109/CVPR.2017.158. 110, 111

Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung. Real-Time Progressive 3D Semantic Segmentation for Indoor Scenes. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1089–1098, 2019. doi: 10.1109/WACV.2019.00121. 54

L. Platinsky, A. J. Davison, and S. Leutenegger. Monocular visual odometry: Sparse Joint Optimisation or Dense Alternation? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5126–5133, 2017. doi: 10.1109/ICRA.2017.7989599. 45, 168

M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed Real-Time Urban 3D Reconstruction from Video. *International Journal of Computer Vision*, 78:143–167, 07 2008. doi: 10.1007/s11263-007-0086-4. 45, 72

D. Pomerleau and T. Jochem. Rapidly Adapting Machine Vision for Automated Vehicle Steering. *IEEE Expert*, 11(2):19–27, 1996. doi: 10.1109/64.491277. 9

V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. Torr, and D. W. Murray. InfiniTAM v3: A Framework for Large-Scale 3D Reconstruction with Loop Closure. *ArXiv e-prints*, Aug. 2017. 69

S. Qiu, H. Liu, Z. Zhang, Y. Zhu, and S.-C. Zhu. Human-Robot Interaction in a Shared Augmented Reality Workspace. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020. 20

V. Ravi Kumar, S. Yogamani, M. Bach, C. Witt, S. Milz, and P. Mäder. UnRectDepthNet: Self-Supervised Monocular Depth Estimation using a Generic Framework for Handling Common Camera Distortion Models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, 2021*, pages 8177–8183, 2020. doi: 10.1109/iros45743.2020.9340732. 129, 130, 131

L. Riazuelo, J. Civera, and J. M. M. Montiel. C2TAM: A Cloud Framework for Cooperative Tracking and Mapping. *Robot. Auton. Syst.*, 62(4):401–413, April 2014. ISSN 0921-8890. doi: 10.1016/j.robot.2013.11.007. 67

A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: From SLAM to Spatial Perception With 3D Dynamic Scene Graphs. *The International Journal of Robotics Research*, 40:1510 – 1546, 2021. 171

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544. 39

M. Rünz and L. de Agapito. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects. *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20, 2018. 54

S. Saeedi, M. Trentini, M. Seto, and H. Li. Multiple-Robot Simultaneous Localization and Mapping: A Review. *J. Field Robot.*, 33(1):3–46, Jan. 2016. ISSN 1556-4959. doi: 10.1002/rob.21620. 18, 55, 59

R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, June 2013. doi: 10.1109/CVPR.2013.178. 54

R. F. Salas-Moreno, B. Glocken, P. H. J. Kelly, and A. J. Davison. Dense planar SLAM. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 157–164, 2014. doi: 10.1109/ISMAR.2014.6948422. 14, 46

D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart. Absolute Scale in Structure from Motion from a Single Vehicle Mounted Camera by Exploiting Nonholonomic Constraints. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1413–1419, 2009. doi: 10.1109/ICCV.2009.5459294. 35

J. L. Schönberger and J.-M. Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 11, 26, 42

T. Schöps, T. Sattler, C. Häne, and M. Pollefeys. Large-scale Outdoor 3D Reconstruction on a Mobile Device. *Computer Vision and Image Understanding*, 157:151–166, 2017. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2016.09.007. 148

T. Schöps, T. Sattler, and M. Pollefeys. BAD SLAM: Bundle Adjusted Direct RGB-D SLAM. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 134–144, 2019. doi: 10.1109/CVPR.2019.00022. 52, 72, 73, 125

T. Schöps, T. Sattler, and M. Pollefeys. SurfelMeshing: Online Surfel-Based Mesh Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42: 2494–2507, 2020. 3, 42, 53

D. Schubert, N. Demmel, V. Usenko, J. Stuckler, and D. Cremers. Direct Sparse Odometry With Rolling Shutter. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 41

H. Seok and J. Lim. ROVO: Robust Omnidirectional Visual Odometry for Wide-baseline Wide-FOV Camera Systems. *2019 International Conference on Robotics and Automation (ICRA)*, pages 6344–6350, 2019. 148

C. Shu, K. Yu, Z. Duan, and K. Yang. Feature-metric Loss for Self-Supervised Learning of Depth and Egomotion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 572–588. Springer, 2020. 132

G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive Relative Bundle Adjustment. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009. doi: 10.15607/RSS.2009.V.023. 36

G. Sibley, C. Mei, I. Reid, and P. Newman. Vast-scale Outdoor Navigation Using Adaptive Relative Bundle Adjustment. *The International Journal of Robotics Research*, 29(8): 958–980, 2010. doi: 10.1177/0278364910369268. 36

R. Smith, M. Self, and P. Cheeseman. *Estimating Uncertain Spatial Relationships in Robotics*, pages 167–193. 1990. ISBN 978-1-4613-8997-2. doi: 10.1007/978-1-4613-8997-2_14. 9

R. C. Smith and P. Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986. doi: 10.1177/027836498600500404. 9, 30

C. Stachniss. *Robotic Mapping and Exploration*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 3642010962. 15, 19

A. D. Stewart and P. Newman. Laps-Localisation Using Appearance of Prior Structure: 6-DOF Monocular Camera Localisation Using Prior Pointclouds. In *2012 IEEE International Conference on Robotics and Automation*, pages 2625–2632. IEEE, 2012. 110, 111, 113

H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale Drift-Aware Large Scale Monocular Slam. In *Proc. Robotics: Science and Systems (RSS)*, 2010a. 35, 38, 40, 67, 128, 157

H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-Time Monocular SLAM: Why Filter? In *2010 IEEE International Conference on Robotics and Automation*, pages 2657–2664, May 2010b. doi: 10.1109/ROBOT.2010.5509636. 12, 29

H. Strasdat, A. J. Davison, J. Montiel, and K. Konolige. Double Window Optimisation for Constant Time Visual SLAM. In *2011 International Conference on Computer Vision*, pages 2352–2359, 2011. doi: 10.1109/ICCV.2011.6126517. 29, 36, 39

C. Studholme, D. Hill, and D. Hawkes. An Overlap Invariant Entropy Measure of 3D Medical Image Alignment. *Pattern Recognition*, 32(1):71 – 86, 1999. ISSN 0031-3203. doi: https://doi.org/10.1016/S0031-3203(98)00091-0. 110

J. Stühmer, S. Gumhold, and D. Cremers. Real-Time Dense Geometry from a hand-held Camera. In *Proceedings of the 32nd DAGM Conference on Pattern Recognition*, page 11–20, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3642159850. 3, 4, 45, 72

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. ix, 4, 74, 99, 101, 108, 114, 118

E. Sucar, S. Liu, J. Ortiz, and A. Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 46, 55, 128, 170, 171

R. W. Sumner, J. Schmid, and M. Pauly. Embedded Deformation for Shape Manipulation. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, 2007. doi: 10.1145/1275808.1276478. 85

N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid. Meaningful Maps With Object-Oriented Semantic Mapping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5079–5085, 2017. doi: 10.1109/IROS.2017.8206392. 54

R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt. Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–32, 2022. URL https://doi.org/10.1145/1122445.1122456. 20

R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. ISBN 1848829345, 9781848829343. 10, 76

M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar. Block-NeRF: Scalable Large Scene Neural View Synthesis. In

*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8248–8258, June 2022. 170

K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6565–6574, 2017. doi: 10.1109/CVPR.2017.695. 54, 129

C. Thorpe, M. Herbert, T. Kanade, and S. Shafer. Toward Autonomous Driving: the CMU Navlab. I. Perception. *IEEE Expert*, 6(4):31–42, 1991a. doi: 10.1109/64.85919. 9

C. Thorpe, M. Herbert, T. Kanade, and S. Shafter. Toward Autonomous Driving: the CMU Navlab. II. Architecture and systems. *IEEE Expert*, 6(4):44–52, 1991b. doi: 10.1109/64.85920. 9

S. Thrun and Y. Liu. Multi-robot SLAM with Sparse Extended Information Filters. In *Robotics Research. The Eleventh International Symposium*, pages 254–266, 2005. ISBN 978-3-540-31508-7. 61, 66

S. Thrun and M. Montemerlo. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25 (5-6):403–429, 2006. doi: 10.1177/0278364906065387. 29

S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Y. Ng. *Simultaneous Mapping and Localization with Sparse Extended Information Filters: Theory and Initial Results*, pages 363–380. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-45058-0. doi: 10.1007/978-3-540-45058-0_22. 28

P. H. S. Torr and A. Zisserman. Feature Based Methods for Structure and Motion Estimation. In *Vision Algorithms: Theory and Practice*, pages 278–294, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. 40, 41

S. Tourani, S. Mittal, A. Nagariya, V. Chari, and M. Krishna. Rolling Shutter and Motion Blur Removal for Depth Cameras. In *2016 IEEE International Conference on Robotics*

*and Automation (ICRA)*, pages 5098–5105, 2016. doi: 10.1109/ICRA.2016.7487715. 4, 126

B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67973-1. 10, 26, 38

N. Tripathi and S. Yogamani. Trained Trajectory Based Automated Parking System Using Visual SLAM. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021. 16

I. Vasiljevic, V. Guizilini, R. Ambrus, S. Pillai, W. Burgard, G. Shakhnarovich, and A. Gaidon. Neural Ray Surfaces for Self-Supervised Learning of Depth and Ego-motion. In *International Conference on 3D Vision*, 2020. 165

E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. Kelly, and S. Leutenegger. Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151, 2018. doi: 10.1109/LRA.2018.2792537. 14, 107

N. X. Vinh, J. Epps, and J. Bailey. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *Journal of Machine Learning Research*, 11(95):2837–2854, 2010. 110

M. Walker, H. Hedayati, J. Lee, and D. Szafir. Communicating Robot Motion Intent with Augmented Reality. In *2018 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 316–324, 2018. 20

M. E. Walker, H. Hedayati, and D. Szafir. Robot Teleoperation with Augmented Reality Virtual Surrogates. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 202–210, 2019. doi: 10.1109/HRI.2019.8673306. 20

M. R. Walter, R. M. Eustice, and J. J. Leonard. Exactly Sparse Extended Information

Filters for Feature-based SLAM. *The International Journal of Robotics Research*, 26(4): 335–359, 2007. doi: 10.1177/0278364906075026. 28

K. Wang, F. Gao, and S. Shen. Real-time Scalable Dense Surfel Mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6919–6925, 2019a. doi: 10.1109/ICRA.2019.8794101. 129

S. Wang, R. Clark, H. Wen, and N. Trigoni. DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, 2017. doi: 10.1109/ICRA.2017.7989236. 129

Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8437–8445, 2019b. doi: 10.1109/CVPR.2019.00864. 125, 126

Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu. NeRF−−: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064*, 2021. 170

D. Wettergreen, C. Thorpe, and R. Whittaker. Exploring Mount Erebus by Walking Robot. *Robotics and Autonomous Systems*, 11(3):171–185, 1993. ISSN 0921-8890. doi: 10.1016/0921-8890(93)90022-5. 9

T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially Extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, July 2012. 14, 49

T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald. Robust Real-Time Visual Odometry for Dense RGB-D Mapping. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, May 2013a. 49

T. Whelan, M. Kaess, J. Leonard, and J. McDonald. Deformation-based Loop Closure for

Large Scale Dense RGB-D SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS*, Tokyo, Japan, November 2013b. 49, 50

T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. Leonard, and J. McDonald. Real-time large scale dense RGB-D SLAM with volumetric fusion. *Intl. J. of Robotics Research, IJRR*, 2014a. 14, 49, 50, 72, 73, 106, 107, 124, 125

T. Whelan, L. Ma, E. Bondarev, P. H. N. de With, and J. McDonald. Incremental and Batch Planar Simplification of Dense Point Cloud Maps. *Robotics and Autonomous Systems (RAS) ECMR '13 Special Issue*, 2014b. 14, 107

T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker, and A. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Proceedings of Robotics: Science and Systems*, July 2015. doi: 10.15607/RSS.2015.XI.001. 14, 20, 51, 53, 73, 74, 75, 105, 108, 124, 125, 130, 134, 136, 137, 143, 150, 156, 157

T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. ElasticFusion: Real-Time Dense SLAM and Light Source Estimation. *Intl. J. of Robotics Research, IJRR*, 2016. 75, 83, 99, 100, 106, 116

S. B. Williams, G. Dissanayake, and H. Durrant-Whyte. Towards Multi-Vehicle Simultaneous Localisation and Mapping. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, volume 3, pages 2743–2748, 2002. doi: 10.1109/ROBOT.2002.1013647. 60, 61

C. Won, J. Ryu, and J. Lim. OmniMVS: End-to-End Learning for Omnidirectional Stereo Matching. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8986–8995, 2019. doi: 10.1109/ICCV.2019.00908. 148

C. Won, H. Seok, Z. Cui, M. Pollefeys, and J. Lim. OmniSLAM: Omnidirectional Localization and Dense Mapping for Wide-baseline Multi-camera Systems. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 559–566, 2020. 148

N. Yang, L. von Stumberg, R. Wang, and D. Cremers. D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 129, 137, 139

S. Yogamani, C. Hughes, J. Horgan, G. Sistu, S. Chennupati, M. Uricar, S. Milz, M. Simon, K. Amende, C. Witt, H. Rashed, S. Nayak, S. Mansoor, P. Varley, X. Perrotton, D. Odea, and P. Pérez. WoodScape: A Multi-Task, Multi-Camera Fisheye Dataset for Autonomous Driving. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9307–9317, 2019. doi: 10.1109/ICCV.2019.00940. 16

Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving. In *ICLR*, 2020. 126

Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza. Benefit of Large Field-of-View Cameras for Visual Odometry. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 801–808, 2016. doi: 10.1109/ICRA.2016.7487210. 7, 173, 174, 175

H. Zhou, B. Ummenhofer, and T. Brox. DeepTAM: Deep Tracking and Mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 129

T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619, 2017. doi: 10.1109/CVPR.2017.700. 46, 129, 132

X. S. Zhou and S. I. Roumeliotis. Multi-robot SLAM with Unknown Initial Correspondence: The Robot Rendezvous Case. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1785–1792, Oct 2006. doi: 10.1109/IROS.2006.282219. 64, 66

Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 170