SEMANTIC 3D RECONSTRUCTION AND BENCHMARKING IN DYNAMIC ENVIRONMENTS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER FOR THE DEGREE OF MASTER OF PHILOSOPHY IN THE FACULTY OF SCIENCE AND ENGINEERING

2020

Horia-Mihai Bujanca

School of Computer Science
Contents

Abstract .............................. 7
Declaration .......................... 8
Copyright .......................... 9
Acknowledgements ..................... 10
Symbols ............................. 11

1 Introduction .......................... 12
   1.1 Objectives ........................ 14
   1.2 Contributions ...................... 15
   1.3 Outline .......................... 16

2 Related Work ......................... 17
   2.1 Simultaneous Localization and Mapping .......................... 17
   2.2 Dense 3D reconstruction .......................... 19
   2.3 Robustness to dynamic environments ...................... 20
   2.4 Non-rigid 3D reconstruction ...................... 21
   2.5 Semantic scene understanding ...................... 22
   2.6 Benchmarking SLAM systems ...................... 24

3 SLAMBench 3.0: benchmarking beyond traditional SLAM ......... 26
   3.1 Background .......................... 26
      3.1.1 Trajectory evaluation .............. 26
      3.1.2 3D reconstruction .............. 28
      3.1.3 Semantic segmentation .............. 30
   3.2 Framework overview ...................... 31
3.3 Non-rigid 3D reconstruction ........................................ 33

4 FullFusion: 3D reconstruction in dynamic environments ................... 36

4.1 Background ................................................................ 36

4.1.1 RGB-D sensing ......................................................... 36

4.1.2 Signed-distance fields ................................................ 37

4.1.3 Non-rigid 3D reconstruction .......................................... 38

4.2 FullFusion Overview ......................................................... 39

4.3 Pose estimation .............................................................. 41

4.4 Segmentation module ....................................................... 41

4.5 Implementation details ...................................................... 44

5 Results ........................................................................... 46

5.1 Experimental setup ......................................................... 46

5.2 Trajectory accuracy ........................................................ 46

5.3 Reconstruction accuracy .................................................. 48

5.4 Runtime and memory analysis .......................................... 50

5.5 Qualitative results .......................................................... 51

5.6 MixedFusion: discussion .................................................. 53

6 Conclusion ....................................................................... 55

6.1 Summary ...................................................................... 55

6.2 Limitations and future work ............................................. 55

6.2.1 Semantic reconstruction .............................................. 56

6.2.2 Monocular depth estimation ......................................... 56

6.2.3 Data representation for dense 3D reconstruction ............... 58

6.2.4 Benchmarking ............................................................ 59

6.3 Final remarks ................................................................ 60

Bibliography ....................................................................... 61

Appendices ......................................................................... 75

A SLAMBench artifacts ........................................................ 76

Word Count: 9723
List of Tables

2.1 Comparison of tasks performed by other state-of-the-art algorithms and FullFusion ........................................ 25
3.1 The VolumeDeform dataset. ............................................. 35
4.1 PASCAL-VOC dataset with movement labels ....................... 42
5.1 Trajectory evaluation results on the freiburg1 and freiburg3 sequences of the TUM RGB-D dataset [120]. ......................... 47
5.2 ICL-NUIM reconstruction error (in cm) ............................... 50
5.3 Reconstruction error (in cm) on the VolumeDeform dataset [61] for our DynamicFusion [93] implementation. ...................... 50
5.4 Run time and memory usage statistics ................................ 51
A.1 List of datasets. * denotes datasets introduced in SLAMBench 3.0. ... 76
A.2 SLAM algorithms included. * denotes algorithms introduced in SLAM-Bench 3.0 ....................................................... 76
A.3 Output of ORB-SLAM2 run under SLAMBench with benchmark_loader on the freiburg1 rpe sequence of the TUM Dataset [120] ........... 78
List of Figures

2.1 LSD-SLAM can build large-scale, globally-consistent semi-dense maps using a single camera. Figure courtesy of Engel et al. [40]. ............ 17
2.2 Left: tabletop scene dense reconstruction from Keller et al. [68] Right: large scene reconstruction from Niessner et al. [95]. ................. 19
2.3 Dynamic scenes are significantly harder to reconstruct than static scenes. Left: RGB frame; Right: depth frame. From the freiburg3 dataset from the TUM RGB-D benchmark [120]. ....................... 20
2.4 Non-rigid scene reconstruction aims to recover the geometry of deforming objects. Image courtesy of Newcombe et al. [93]. ............ 21
2.5 Semantic scene reconstruction from SemanticFusion [84]. Left: reconstructed scene; Right: scene labels. ......................... 22
3.1 Metrics such as the Absolute Trajectory Error (ATE) are often used to evaluate the performance of pose estimation ................... 27
3.2 Left: ground-truth and estimated trajectory; Right: aligned trajectories using Umeyama's method. ......................... 28
3.3 Error heatmap visualisation in SLAMBench. Red represents large error, while green represents small error. ..................... 29
3.4 The point-to-plane error used to assess the quality of 3D reconstruction was introduced in [82]. The same error is used in the Iterative Closest Point (ICP) method which aligns two sets of points. Figure from the original publication. ......................... 30
3.6 Overview of the SLAMBench structure. The Loader module loads Datasets and Algorithms, and connects the outputs to the Metrics and Front-ends. From [13] .......................... 32
3.7 Workflow of the SLAMBench API ................................ 33
3.8 Left: input frame and internal 3D point cloud and the trajectory from ElasticFusion. Right: ORB-SLAM2 on a sequence from EUROCV-MAV; sparse map and trajectory. 33

4.2 The Stanford Armadillo [73]. Left: surface view; Right: discretised voxel volume. 38

4.3 RGB-D input from a sensor such as Microsoft Kinect is divided into a static and a dynamic frame by the Segmentation module. The Pose Estimation module uses the static frame and a reference frame from the static model to compute the camera position, thus reducing ambiguity between scene dynamics and camera movement. Finally, each of the reconstruction systems receives its processed input, along with the estimated camera pose and semantic labels. 39

4.4 Our implementation of the Segmentation module uses DeepLabv3+ trained on the PASCAL-VOC dataset along with a geometric clustering approach to produce geometrically-consistent semantic segmentation. 41

4.5 Semantic segmentation without (left) and with (right) depth clustering. 42

4.6 Visualisation of the FullFusion GUI, on two frames from the upper-body sequence. In clockwise order, from top-left: RGB frame, Semantic segmentation (without geometric clustering), Dynamic Reconstruction, Depth Frame. Top image: phong-shaded background reconstruction and dynamic model with normals. Bottom: dynamic model highlighted in red. 45

5.2 Example of failure in KinectFusion reconstruction vs FullFusion reconstruction on the highly dynamic sequences in the TUM RGB-D dataset. Left: input depth; Middle: KinectFusion; Right: FullFusion. Note that the scene is rotated to fit the estimated camera pose, hence the difference in perspective. 51

5.3 Final 3D reconstruction on the upperbody sequence from the VolumeDeform dataset, with semantic labels. 52

5.4 Reconstruction from the hoodie sequence from the VolumeDeform dataset. 53

6.1 Qualitative comparison of depth estimation using DenseDepth [6]. Left: ground-truth; Right: estimated depth. 56
Abstract

SEMANTIC 3D RECONSTRUCTION AND BENCHMARKING IN DYNAMIC ENVIRONMENTS
Horia-Mihai Bujanca
A thesis submitted to the University of Manchester for the degree of Master of Philosophy, 2020

Simultaneous Localisation and Mapping, or SLAM is a key component in many applications, such as autonomous robot and vehicle navigation, motion capture and augmented reality. With new sensors such as the Microsoft Kinect and the wide availability of GPUs, SLAM has evolved towards generating increasingly more meaningful representations of the environment: 3D reconstruction and semantic scene understanding are common tasks addressed by state-of-the-art systems.

The core contribution of this thesis is a technique and an evaluation methodology to address the problem of semantic 3D reconstruction in dynamic environments, which aims to unify previous paradigms and simultaneously recover the semantic and geometric aspects of deforming objects and the static background.

Firstly, we develop an evaluation method by extending the open-source SLAM-Bench [5] framework to support new sensors, metrics and datasets. We then propose FullFusion, a framework which uses RGB-D sensors to recover the geometry and semantic information of dynamic scenes, and develop a baseline implementation using KinectFusion [94], DynamicFusion [93], and a novel segmentation module which uses geometry and semantic labels.

Our method is the first to address the problem of semantic 3D reconstruction in dynamic environments. Additionally, our results show state-of-the-art performance in pose estimation, proving that semantic labels can be used to discard unreliable elements when estimating the pose of a moving sensor.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

Firstly, I would like to thank my supervisor, Mikel, for his ongoing support and patience both on an academic and personal level. I could not have asked for a more understanding and encouraging supervisor. I also wish to thank my examiners, Prof. John Goodacre and Prof. Andrew Davison for taking the time to review my work and for their thoughtful comments.

I also want to thank all those who contributed to the success of SLAMBench, in particular Bruno Bodin for his work on SLAMBench 2.0 and Paul Gafton for the work on SLAMBench 3.0. Without their dedication, my research would not have been possible. My colleagues in the PAMELA project and the support I have found in the RAIN Hub helped enormously, for which I am grateful.

I need to acknowledge The Association for Computational Heresy, of which I am a devout follower, and their flagship conference SIGBOVIK, for showing me the light-hearted side of research and reminding me that, at the end of the day, we do what we do because it’s fascinating and fun, despite the stress associated with deadlines.

Thanks to Luis Plana, Andy Nisbet, and Sami Alabed for the useful discussions and constructive criticism. My dear friends, Horia and Valentina, as well as Mariya made all the difference during hard times. I cannot thank them enough for the patience to listen to my worries, problems, and struggles before paper deadlines. Finally, I need to thank my family for their love and encouragement. The words of wisdom and advice will forever be appreciated.

I regard this as a tribute which doesn’t and cannot do justice to the endless support I’ve had, words are not enough!
Symbols

t        timestamp
\( \mathcal{W}_t \)   Warp field at time t
RMSE       Root Mean Square Error
ATE        Absolute Trajectory Error
RPE        Relative Pose Error
ICP        Iterative Closest Point
SO(3)      3D rotation group
X, \( \hat{X} \)   Ground-truth trajectory, estimated trajectory
P_i, \( \hat{P}_i \)   Ground-truth, estimated pose at frame i
M, \( \hat{M} \)   Ground-truth mesh, estimated mesh
V        Vertices of a mesh M
E        Edges of a mesh M
\( \Phi \) Truncated distance function
\( E_{\text{traj}} \) Trajectory alignment energy function
\( E_{\text{icp}} \) Iterative Closest Point energy function
\( E_{\text{warp}} \) Warp field state energy function
\( E_{\text{ARAP}} \) As-rigid-as-possible energy term of \( E_{\text{warp}} \)
\( E_{\text{data}} \) Data energy term of \( E_{\text{warp}} \)
\( \mathcal{F}_t, \mathcal{F}_{t, \text{static}}, \mathcal{F}_{t, \text{dynamic}} \) RGB-D frame pair at time t, static frame, dynamic frame
B, T        Base shape to be warped into the Target shape
\( \lambda \) As-rigid-as-possible regularisation coefficient
C_t, D_t   Colour, depth frame at time t
L        Set of possible semantic labels
\( \Omega \) Image plane
S        Semantic segmentation map
Chapter 1

Introduction

Visual sensing is ubiquitous: from simple photosensitive single-protein receptors to complex and diverse eyes, the vast majority of living organisms developed ways of perceiving light. Eyes appeared during the Cambrian period, and it is estimated that 96% of the living animal species have complex visual systems. In humans, half the brain is related either directly or indirectly to visual processing. Vision is used everywhere, in almost all the tasks we engage with, and this fact is reflected in our thinking, culture, and interactions. More importantly, much of our understanding of the world is closely tied with vision.

Given the utility of visual sensing, it is not surprising that humans have built artificial visual sensors. The first cameras were designed in the 19th century, and could capture black-and-white images on a silver plate. 200 years later, cameras are deeply integrated in everyday use: mobile devices, surveillance systems, and advertising all make extensive use of cameras. Moreover, specialised devices which play a crucial role in the advancement of science are available today: microscopes which can zoom to sub-atomic scales, telescopes which see light-years away, and cameras able to capture millions of frames every second. Yet, even equipped with advanced sensors, our machines are not able to perceive the world the way we, humans do. This highlights an important distinction between sensing and perception. Sensing refers to measuring a certain quantity - in this case, the intensity of light - while perception entails interpreting the measurements, often based on prior knowledge. It is perception which enables us to understand our environment, take decisions and act.

Having built the sensors, addressing perception was initially believed to be a reasonably straightforward task: in 1966, Seymour Papert proposed a summer project at the MIT Artificial Intelligence Lab, with the aim to “construct a significant part of
a visual system” [99]. It quickly became apparent that perception was a more than a matter of feeding images into the computer and applying basic pattern analysis algorithms. A new discipline dedicated to the creation of artificial visual perception systems emerged: computer vision. One of the fundamental tasks in computer vision is that of representing and understanding the 3D structure of the world. At the core of this task sits Simultaneous Localisation and Mapping (SLAM), originally described in the 1980s as the problem of building a consistent map of the surroundings and estimating the position of a mobile robot within the map, with no prior information about the initial position or the mapped environment. Durrant-Whyte, one of the pioneering researchers on SLAM has referred to the problem as a “holy grail” of mobile robotics, postulating that SLAM would play a crucial role in the development of fully autonomous robots [36].

In the wake of major advancements in Artificial Intelligence (AI), SLAM may be of greater impact than ever, powering technologies which already have a deep impact on the society and economy, such as robot navigation, autonomous driving and augmented reality. One of the early breakthroughs of self-driving cars using SLAM was in 2005, when the Stanley vehicle won the DARPA Grand Challenge, driving over 200km off-road in the Mojave Desert. Today, fully autonomous vehicles are entering the market, with companies such as Google, Tesla and Uber showcasing large fleets of self-driving cars and autonomous taxi services. The first notable success of Augmented Reality (AR) in the mainstream was Pokémon GO [1], an AR game which attracted over 150 million users and produced over $3 billion in revenue, additionally setting Nintendo on the path to more than triple their share price. Apple’s ARKit [2] and Google’s ARCore [3], as well as AR headsets such as Microsoft Hololens have now brought Standard Development Kits (SDKs) to developers, enabling them to create AR mobile apps without requiring specialised computer vision knowledge.

New technologies brought forth new challenges: while it can be argued that SLAM in a controlled, static environment is a solved problem, mobile robots operating in real-life, dynamic scenarios require an advanced understanding of their surroundings, beyond representing the geometry of static scene elements. In a recent “State of The Art Report on 3D reconstruction with RGB-D cameras” [141], Zollhfer identifies the capture and modelling of dynamic scenes and embedding semantic information as two of the important research directions. Semantic scene labelling facilitates a shared understanding of the environment between humans and machines, opening up new possibilities for meaningful interaction. For instance, semantic information may be used by
Chapter 1. Introduction

Figure 1.1: Applications of SLAM: (a) Augmented Reality games such as Pokémon Go, (b) Drone navigation, (c) Autonomous driving, and (d) Augmented Reality headsets. Images courtesy of Niantic, R-Style Lab, and Microsoft

Humans for queries: “How many students are there in the classroom?”, or providing textual or vocal commands: “Close the rightmost valve”. Combining semantics with dynamic scene reconstruction can enable even more powerful applications such as interaction and intent understanding, potentially allowing for more seamless cooperation between humans and machines. Current research is moving towards what has been referred to as the ‘Spatial AI’ [32] which aims to combine geometry with semantic scene understanding. Long-term, the goal is to develop means of organising sensory information to enable robotic systems to observe, interpret, and navigate the world like humans do.

1.1 Objectives

We propose a framework for reconstructing the semantic and geometric aspects of scenes containing both static and dynamic objects. Additionally, we develop a benchmarking methodology which aims to measure the performance of the various components.

The development of an evaluation methodology was chosen as the first step, as
it involved familiarisation with the state-of-the-art in 3D reconstruction and semantic segmentation, as well as evaluation techniques and metrics. Specifically, we wanted to evaluate the trajectory estimation, static and dynamic reconstruction and semantic segmentation accuracy. The SLAMBench framework [90] was identified as a good starting point as it already provided evaluation metrics and datasets for semantic segmentation, static reconstruction and trajectory estimation.

Secondly, we aim to create a modular framework for semantic 3D reconstruction of dynamic scenes. In contrast with the state-of-the-art, which commonly features tightly-coupled systems, one of our core aspirations was modularity, providing the specification for the interaction of the modules. This enables a wide range of benefits: future implementations will only need to respect the interface specification, without requiring knowledge of the other modules; depending on the task at hand, modules may be turned off or replaced by sensors (for instance, if an external positioning system is available).

To sum up, we set forth to tackle the following goals:

- Reconstruction of both static and non-rigidly deforming geometry
- Semantic labelling of 3D scenes
- Accurate localisation in dynamic environments
- Modular design with loosely-coupled subsystems

### 1.2 Contributions

To achieve the aforementioned objectives, a framework for 3D reconstruction of geometry and semantics for dynamic scenes was developed, along with evaluation methods for the various components of the framework. This thesis claims the following contributions:

- Extended the SLAMBench 2.0 framework to enable the evaluation of non-rigid 3D reconstruction algorithms by integrating the VolumeDeform dataset [61].
- Developed, integrated, and evaluated an implementation of the DynamicFusion method [93].
- Developed FullFusion, a modular framework able to reconstruct the geometric and semantic aspects of dynamic scenes.
A baseline implementation of FullFusion featuring a segmentation module based on scene semantics and 3D geometry.

State-of-the-art performance in trajectory estimation.

We show that semantic scene labels not only embed desirable information in 3D reconstruction systems but can additionally be used as priors for reconstruction mechanism selection, depending on the properties of the objects present in the scene.

The contributions towards extending SLAMBench presented in this thesis and the obtained results constituted part of the “SLAMBench 3.0: Systematic Automated Reproducible Evaluation of SLAM Systems for Robot Vision Challenges and Scene Understanding” [15] paper published at ICRA 2019, Montreal, Canada. The code will be made available on GitHub\(^1\).

The reconstruction framework “FullFusion: A Framework for Semantic Reconstruction of Dynamic Scenes” was accepted at the Workshop on 3D Reconstruction in the Wild at ICCV 2019, Seoul, South Korea and will be published with the ICCV workshop proceedings.

### 1.3 Outline

Chapter 2 first discusses the current literature to provide a context for our contributions. Next, evaluation methodologies are discussed in Chapter 3, and an overview of SLAMBench is provided, followed by a discussion of the additions to the framework in order to support the evaluation of non-rigid reconstruction systems.

Chapter 4 explores the FullFusion framework and the details of a baseline implementation, followed by a set of experiments performed under SLAMBench, and a discussion comparing FullFusion to state-of-the-art reconstruction algorithms in Chapter 5. Finally, in Chapter 6 we consider future research directions and show preliminary results in adapting FullFusion to work with colour cameras only.

\(^1\)SLAMBench GitHub repository (https://github.com/mihaibujanca/slambench3)
Chapter 2

Related Work

Recent years have seen Simultaneous Localisation and Mapping (SLAM) systems evolve to handle a broader number of real-world applications [20, 141]. This chapter aims to contextualise our research by defining a timeline of the most influential developments in SLAM and 3D reconstruction, as well as providing a brief review of the state-of-the-art.

2.1 Simultaneous Localization and Mapping

Figure 2.1: LSD-SLAM can build large-scale, globally-consistent semi-dense maps using a single camera. Figure courtesy of Engel et al. [40].
CHAPTER 2. RELATED WORK

The SLAM idea can be traced as far as the end of the 1980s [37, 117], with most systems using Extended Kalman filters [67] to jointly estimate the position of the camera and the observations of the environment. Research widely adopted Extended Kalman filter based methods (EKF-SLAM) [116], until the publication of FastSLAM [86]. FastSLAM achieved better accuracy and speed by employing the Rao-Blackwellized particle filter [21] for state estimation, removing some of the assumptions of EKF-SLAM such as linearity and Gaussian distribution of camera poses.

Soon afterwards, real-time methods were introduced [31], however they were restricted to small, controlled setups. The topic has benefited from a significant amount of attention after the release of MonoSLAM [33], which for the first time presented the application of SLAM to humanoid robots and Augmented Reality (AR). At the same time, efforts to improve runtime performance led to Parallel Tracking and Mapping (PTAM) [72], a system which split the tracking and mapping tasks onto different threads. In a landmark paper, the same authors adapt PTAM to demonstrate real-time markerless augmented reality on mobile devices.

Modern monocular (single-camera) SLAM methods are broadly divided in two categories: feature-based methods such as OKVIS [76], based on BRISK [75] features and ORB-SLAM [87], which performs sparse mapping using ORB features [104] generally build sparse maps based on landmarks. Recently ORB-SLAM has been extended to work with depth sensors and stereo cameras [87], and to integrate the Inertial Measurement Unit (IMU) on mobile devices to perform visual-inertial SLAM [88], however ORB features are still central to the technique. Meanwhile, direct methods compare consecutive frames or frames and keyframes to estimate depth and camera displacement. Important works include LSD-SLAM [40], a method which performs semi-dense mapping, and has been extended to work on mobile devices [110] and with stereo cameras [41] and Dense Tracking and Mapping (DTAM), which uses GPU acceleration estimates dense depth by minimising the photometric error across a set of colour frames.

New developments in the area of dense monocular SLAM use machine learning-based approaches to estimate depth. CNN-SLAM [121] uses a convolutional neural network (CNN) for depth estimation on keyframes, while CodeSLAM [11] uses a depth auto-encoder and fuses multiple views to reconstruct the scene. While these approaches achieve remarkable results, they require powerful hardware and can not yet be applied to computationally-restricted platforms.
2.2 Dense 3D reconstruction

Most work in SLAM was concentrated around stable camera pose estimation (i.e. localisation), while the map representation was seen only as a means to enable pose estimation by storing landmarks. A major shift in this view can largely be attributed to the release of the Microsoft Kinect RGB-D camera, followed by the publication of KinectFusion [94], a seminal work which introduced the first real-time dense RGB-D reconstruction algorithm. KinectFusion inspired a great deal of research to take advantage of the depth capabilities provided by this device and similar cameras. Much of the subsequent work endeavoured to reconstruct geometrically accurate and visually rich 3D models of real environments, with efforts targeted at improving the mapping, rather than focusing on estimating the camera trajectory.

KinectFusion estimates the pose of a moving sensor and uses Truncated Signed Distance Function (TSDF) [26], a data structure where each voxel stores the distance to the closest surface, as well as a weight that measures the uncertainty of the surface measurement. A major improvement on the technique is VoxelHashing [95], which propose a hierarchical hashing approach to store and access voxels. Further improvements are presented by Kahler et al. [65], using variable voxel sizes to improve memory efficiency while preserving mesh quality. Later, algorithms such as InfiniTAM [102, 66] focused on reconstructing large scenes. Recent developments include BundleFusion [28], which performs on-the-fly surface reintegration in real-time to produce globally consistent reconstructions.

ElasticFusion [133], based on Keller et al. [68] is a globally consistent approach that does not require a pose graph and uses fused surfels [101] (disk-shaped data structure describing a plane by using a point with a normal associated) rather than voxels as data structure to represent the environment.
2.3 Robustness to dynamic environments

![Figure 2.3](image)

From the earliest stages of 3D reconstruction, the issue of creating systems robust to dynamic environments was recognised as a major challenge. Movement creates ambiguity: rather than relying on the assumption that any difference from one frame to the next frame is due to camera movement, a system needs to determine which changes are caused by camera movement, and which are caused by movement in the scene. Extending KinectFusion, Izardi et al. [62], initially reconstructs parts of the static scene, and treats any dynamic objects entering the scene as outliers in the data association step. Keller et al. [68] introduced a surfel-based method which segments moving foreground objects without assuming that they would only occupy a small portion of the scene. Jaimez et al. [63] proposed a joint visual odometry and scene flow (VO-SF) method which segments the scene into rigid clusters and filters out clusters with high registration error. Building on VO-SF, StaticFusion [111] adopts the same segmentation approach and uses ElasticFusion [133] to reconstruct the scene. More recently, Re-Fusion [98] exploited registration residuals to segment out high-error regions corresponding to scene motion.

Making the observation that purely geometric approaches suffer from inherent ambiguities, new approaches leverage advancements in machine learning and rely on heuristics to segment the scene: PoseFusion [137] uses OpenPose to segment out humans by fitting a skeleton, and ElasticFusion as reconstruction system. DynaSLAM [9] uses ORB-SLAM2 [87] along with an approach which combines semantic segmentation with geometry to segment out the dynamic part, showing good improvements in
2.4 Non-rigid 3D reconstruction

Figure 2.4: Non-rigid scene reconstruction aims to recover the geometry of deforming objects. Image courtesy of Newcombe et al. [93].

Early attempts to reconstruct non-rigid scenes relied on strong priors: reconstructions of hands [97, 122], faces [78, 124, 131] and full bodies [126, 136]. Given that priors greatly restrict the applications, research into generalising the method to arbitrary non-rigid objects received significant attention. Due to the non-linear nature of the problem, most approaches either reconstruct scenes offline [52, 53, 77] or pre-scan target objects and track non-rigid movement online [140].

Building on the KinectFusion method, DynamicFusion [93] introduced the first system able to capture non-rigidly deforming scenes in real-time. Their system uses depth frames from a Kinect device to build a canonical model of the subject, and a coarse deformation graph which transforms the canonical model into the live frame. VolumeDeform [61] improves on the technique by taking into consideration colour frames, computing SIFT [83] features and using them to improve tracking by speeding up alignment between frames and enhance the reconstruction when geometry is uniform. They also provide a higher resolution of the deformation field, resulting in a higher reconstruction quality compared to DynamicFusion. Guo et al. [54] introduce a pipeline which uses shading information to improve the non-rigid registration and temporal correspondences to estimate surface appearance. SurfelWarp [46] employs surfels rather than a TSDF volume and a deformation graph similar to DynamicFusion for computing correspondences. KillingFusion [113] and SobolevFusion [114] take a
different approach and use displacement vectors directly on the TSDF volume, rather than explicit correspondences.

Priors or complex setups are widely used to perform high-quality online reconstruction: BodyFusion [134] fits a skeleton template for tracking, while HybridFusion [138] uses eight inertial measurement units attached to the reconstructed subject. Fusion4D [35] and Motion2Fusion [34] achieve impressive results wielding complex setups that involve four stereo-camera sensors positioned around a moving subject.

While these systems show great potential for applications such as motion capture, they either require complex setups, or do not scale well beyond modelling a single deforming object. As such, they are not fit to be used in applications such as human-robot cooperation, where memory and processing power are restricted, and both static and moving elements need to be modelled.

### 2.5 Semantic scene understanding

![Semantic scene reconstruction from SemanticFusion [84].](image)

Figure 2.5: Semantic scene reconstruction from SemanticFusion [84].
Left: reconstructed scene; Right: scene labels.

The problem of visual scene understanding can be defined as using images to infer geometric and semantic information about scenes [91]. Improvements in hardware, and GPU computing in particular, as well as advancements in machine learning have enabled the development of novel methods for semantic understanding of visual data. The breakthrough of AlexNet [74] winning the ImageNet [106] challenge in 2012, paved the way for convolutional neural networks (CNNs) to become the standard tool
for image processing in machine learning. Unfortunately, image-only approaches suffer from lack of perspective of the scene, omitting important information. Progress on both SLAM and neural networks led to approaches that attempt to jointly solve the two problems, with impressive results. Semantic labelling may be solved more accurately by considering additional information from multiple views of an individual object within a scene.

Semantics were initially introduced through template matching [112, 89, 109] or using various 3D feature descriptors [80, 55, 43] with different degrees of accuracy when used to perform batch processing. A more recent promising approach, [8] performs offline semantic segmentation of large spaces using a hierarchical clustering method based on matching "peak-gap-peak" signals in a density histogram of the acquired point cloud. SemanticPaint [128, 48] employs InfiniTAMv2 [64] to perform live reconstruction and trains random forests [14] online to predict whether new input matches known classes and performs automatic labelling, where possible.

Real-time semantic segmentation along and 3D reconstruction has received more and more attention in the past few years. Vineet et al. [129] presents a system for large-scale semantic reconstruction, with impressive results on the KITTI dataset [47]. Nguyen et al. [123] proposes an annotation tool that integrates both 2D and 3D segmentation, while providing a means to correct any inaccuracies the automatic segmentation system might have produced. SemanticFusion [84] is built upon ElasticFusion [133] to perform SLAM and extends the image semantic segmentation CNN proposed by [96].

Monocular semantic reconstruction methods have also emerged: CNN-SLAM [121] jointly predicts depth and semantic labels from single images. SceneCode [139] extends the code-based representation of CodeSLAM [11] to encompass both semantics and geometry and perform monocular dense semantic reconstruction.

Datasets for training and evaluating semantic segmentation in 2D and 3D scenarios are increasingly common. NYU RGB-Dv2 [92], and SceneNet RGB-D [85] provide pixel-level annotations. ScanNet [27] holds 2.5 million RGB-D views annotated with 3D camera poses, surface reconstructions, and semantic labels. Meanwhile, Armeni et al. [7] provides a dataset of RGB-D with instance-level semantic labels.

Applications such as scene completion e.g. ScanComplete [29] were recently put forward. Despite the wealth of research on semantic scene understanding, the vast majority of systems require scenes to be static.
2.6 Benchmarking SLAM systems

As previously shown, the SLAM area is diversifying to address an ever-growing number of tasks. With this, the development of increasingly diverse and complex evaluation methods and datasets was needed, often addressing a wide spectrum of related problems. Numerous real-world and synthetic datasets for various tasks in robotics and computer vision are now available: InteriorNet [79], SUNCG [118], and SceneNet [57] for semantic labelling; Sintel [19] for optical flow, MIT CBCL for face recognition [4]; New College [115], UnrealCV [103], ICL-NUIM [58], and EuRoC [17] for trajectory estimation. Often these datasets implement their own input and output interfaces and may bring various dependencies. The lack of standardisation can make evaluation time consuming and introduce errors.

To facilitate reproductibility, tools such as the KITTI Benchmark Suite [47] and the TUM RGB-D Benchmark [120] were developed, often requiring users to submit their code online to ensure consistent evaluation. More recently, open-source approaches to benchmarking have appeared. EVO [50] visual odometry evaluation tools which integrates a number of popular datasets. Similarly, tools such as DAWNBench [25] exist for machine learning and AI tasks.

SLAMBench [90] arguably one of the most advanced benchmarking tools for SLAM systems was created as part of the PAMELA project, an enterprise that focused on vision for low-power devices. While the initial version focused on different implementations of a single algorithm, SLAMBench 2.0 [13] was the first framework to integrate a variety of algorithms, datasets, and metrics, providing users with the necessary tools to effortlessly compare traditional SLAM systems. Recent developments on SLAMBench [15] expanded the scope of the framework to enable the evaluation of semantic scene reconstruction. Considering that SLAMBench is an open-source framework which already integrates a number of relevant algorithms, metrics and datasets, we adopt it for evaluation, and extend the existing codebase to include the evaluation of non-rigid reconstruction algorithms.

Remarks To the best of our knowledge, the issue of semantic 3D reconstruction with both static and non-rigid objects has not been addressed so far. While pipelines, benchmarks and datasets which target some of these elements are available, none apply to all the concepts. Our work distinguishes itself from the state-of-the-art by presenting FullFusion, a modular system addressing the aforementioned problem, and integrates
### 2.6. Benchmarking SLAM Systems

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Static</th>
<th>Non-rigid</th>
<th>Robust to dynamic scenes</th>
<th>Semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>KinectFusion [94]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DynamicFusion [93]</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>StaticFusion [111]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SemanticFusion [84]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MixedFusion [135]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>FullFusion (ours) [16]</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of tasks performed by other state-of-the-art algorithms and FullFusion

Existing datasets and metrics into an unified framework, SLAMBench 3.0, facilitating reproducible research. Table 2.1 shows an overview of the tasks performed by FullFusion in comparison with some of the algorithms mentioned earlier.
Chapter 3

SLAMBench 3.0: benchmarking beyond traditional SLAM

As the SLAM research area matures, and an increasingly wider range of datasets and algorithms are developed, it is often very time consuming to compare a new algorithm with existing ones. As the algorithms for scene understanding are getting more diverse, from pose estimation to object recognition, and new computational challenges are emerging, better tools are needed for efficient and easy benchmarking of the algorithms. SLAMBench [13, 90] is a benchmarking suite which allows users to compare their solutions against other algorithms and ground-truth data over a variety of metrics, for tasks such as pose estimation, semantic segmentation and 3D reconstruction.

We extend SLAMBench by adding an infrastructure to support the evaluation of non-rigid reconstruction systems, and integrate the VolumeDeform [61] dataset, along with an implementation of the DynamicFusion [93] method.

3.1 Background

3.1.1 Trajectory evaluation

Visual odometry is the task of estimating the pose of a moving sensor, defined as $P_i = \{t_i, R_i\}$, with translation $t_i \in \mathbb{R}^3$ and rotation $R_i \in \text{SO}(3)$. A trajectory $X = \{P_i\}_{i=0}^N$ is defined as the set of all poses in a sequence with $N$ frames. A complete SLAM system aims to combine visual odometry with globally-consistent mapping, and all extensions of SLAM systems depend on accurate pose estimation.

The TUM RGB-D benchmark [120] adopted the Absolute Trajectory Error (ATE)
3.1. BACKGROUND

Figure 3.1: Metrics such as the Absolute Trajectory Error (ATE) are often used to evaluate the performance of pose estimation.

and Relative Pose Error (RPE) metrics to assess the quality of trajectory estimation, and since then, they have been the most widely used metrics. SLAMBench reports both metrics for any datasets containing ground-truth trajectories. The RPE measures the difference between a fixed number $\Delta$ of estimated poses $\{\hat{P}_t, \ldots, \hat{P}_{t+\Delta}\}$ and the respective ground-truth poses $\{P_t, \ldots, P_{t+\Delta}\}$. The Relative Pose Error at frame $i$ is defined as:

$$RPE_{i,\Delta} = \left( P_{i-1}^{-1} P_{i+\Delta} \right)^{-1} \left( \hat{P}_{i-1}^{-1} \hat{P}_{i+\Delta} \right)$$

(1)

Intuitively, RPE can be used to measure the local pose drift. To obtain the final RPE over a full sequence, it is common to compute the root mean squared error (RMSE) using the translation component:

$$\text{RMSE} (\text{RPE}_{1:N, \Delta}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \| \text{translation} (\text{RPE}_{i,\Delta}) \|^2}$$

(2)

To evaluate the global consistency of the evaluated trajectory, the Absolute Trajectory Error can be computed by directly measuring the absolute distances between an
(aligned) estimated trajectory and the ground-truth trajectory:

\[
\Delta R_i = R_i(\hat{R}_i)^T, \\
\Delta t_i = t_i - \Delta R_i \hat{t}_i, R_i.
\]

\[
\text{ATE}_{\text{rot}} = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} \|\Delta R_i\|^2}, \\
\text{ATE}_{\text{trans}} = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} \|\Delta t_i\|^2}.
\]

(3)

As the ground-truth and the estimated trajectory each have their arbitrary coordinate frames, before computing the ATE, the two trajectories need to be aligned. The method proposed by Umeyama et al. [127] is most often used for this purpose. Given the ground-truth trajectory \(X\) and the estimated trajectory \(\hat{X}\), we need to find a transformation \(\sigma' = \{s', R', t'\}\) with scale \(s'\), rotation \(R'\) and translation \(t'\) by solving the least-squares minimisation problem:

\[
E_{\text{tra}} = \arg\min_{\sigma'} \sum_{i=0}^{N} \|t_i - s' R' \hat{t}_i - t'\|^2
\]

(4)

### 3.1.2 3D reconstruction

Evaluating the quality of a 3D reconstruction is an inherently difficult problem, as there are infinitely many ways of creating discrete representations of continuous surfaces. To

---

Figure 3.2: Left: ground-truth and estimated trajectory; Right: aligned trajectories using Umeyama’s method.
address this, it is common to use both datasets with synthetically-generated ground-truth reconstructions, as well as real-world datasets, each of which have their limitations. Real-world datasets such as Matterport3D [23] often use advanced 3D scanning devices to obtain high-quality reconstructions used as ground-truth. Nonetheless, no sensors and capture processes cannot perfectly capture the real world. On the other hand, synthetic datasets can ensure perfect ground-truth reconstructions, however accurate inputs are hard to simulate: generating plausible trajectories, noise models, and motion blur is some of the limiting aspects of synthetic datasets. As seen in Table A.1, SLAMBench considers these issues and includes both types of datasets to facilitate more complete evaluations.

For quantitative evaluation, the Reconstruction Error (RE) metric was proposed by Handa et al. [59], adopting the point-to-plane distance from Low et al. [82]. Given a ground-truth mesh $\mathcal{M} = \{V, E\}$ with vertices $V$ and edges $E$, and a reconstruction $\hat{\mathcal{M}}$, the RE is given by the mean of the distances from every vertex in $\hat{V}$ to the nearest surface in $\mathcal{M}$. Similarly to ATE, due to the fact that the ground-truth and the estimated reconstruction will often be in arbitrary coordinate spaces, the meshes are first aligned using the Iterative Closest Point (ICP) [105] method.

ICP minimises the point-to-plane distance: given a reconstructed vertex $\hat{v}_i = [\hat{v}_{ix}, \hat{v}_{iy}, \hat{v}_{iz}]^T$, and a point and normal pair in the ground-truth mesh $\{v_i = [v_{ix}, v_{iy}, v_{iz}]^T, n_i = [n_{ix}, n_{iy}, n_{iz}]^T\}$,
3.1.3 Semantic segmentation

SLAMBench provides metrics and datasets for benchmarking semantic scene reconstruction. The Pixel Accuracy metric is computed as the Mean Intersection over Union (mIoU) of the ground-truth segmentation and the segmentation of the scene reprojected into the camera frame. Given a ground-truth segmentation and a prediction, the (IoU) measure represents the similarity between the ground-truth and the prediction:

$$\text{IoU} = \frac{\text{true positives}}{\text{false positives} + \text{true positives} + \text{false negatives}}$$  \hspace{1cm} (6)

where for any class $L_i$, the true positives are the pixels predicted correctly, the false negatives are the pixels belonging to $L_i$ but are incorrectly predicted as a different class, and the false positives are the pixels belonging to a different class but are predicted as $L_i$. The Pixel accuracy is then the mean IoU across all the recognised classes.
3.2 FRAMEWORK OVERVIEW

In addition to the Pixel accuracy metric used for quantitative evaluation, SLAMBench facilitates qualitative evaluation by displaying the Confusion matrix (Fig. 3.5c) as a visualisation of the Pixel accuracy, as well as allowing users to view semantically-annotated point clouds at runtime (Fig. 3.5b).

3.2 Framework overview

SLAMBench is a dataset-agnostic and sensor-agnostic framework for qualitative, quantitative and easily reproducible evaluation of SLAM systems with plug-and-play algorithm support [13]. SLAMBench is structured into five core components: The I/O component defines a unified format which supports a variety of sensors and ground-truth data. The API component provides a generic interface integrating SLAM algorithms, with functions for configuration, processing and output extraction. The Metrics
Figure 3.6: Overview of the SLAMBench structure. The Loader module loads Datasets and Algorithms, and connects the outputs to the Metrics and Front-ends. From [13]

**component** provides a robust infrastructure for comparing the output of the algorithms with the ground-truth and extracting relevant quantitative metrics. The **UI component** allows loading the inputs, outputs and ground-truth of a running SLAM towards a visualisation pipeline for qualitative evaluation. Finally, the Loader module interfaces all other components.

The **I/O System** translates existing datasets into SLAMBench dataset files which contain a description of the dataset sensors, frames, and ground-truth data. To facilitate reproducibility and avoid potential issues when rebuilding SLAMBench dataset files, they are instead entirely self-contained. The dataset file format is described below:

```
DATAFILE = <HEADER><SENSORS><GT_FRAMES><IN_FRAMES>
HEADER = <VERSION><SENSOR_COUNT>
SENSORS = <SENSOR 1>...<SENSOR N>
SENSOR = <TYPE><PARAMETERS>
GT_FRAMES= <EMPTY>|<GT_FRAME 1>...<GT_FRAME N>
IN_FRAMES= <IN_FRAME 1>...<IN_FRAME N>
GT_FRAME = <TIMESTAMP><GT_TYPE><DATA>
IN_FRAME = <TIMESTAMP><IN_TYPE><DATA>
GT_TYPE = POSE|POINT_CLOUD|SEMANTIC
IN_TYPE = RGB_FRAME|DEPTH_FRAME|IMU_FRAME|...
```

**Listing 3.1**: The SLAMBench dataset file format.

SLAMBench defines an API used to interface SLAM algorithm implementation, and includes features for configuring algorithm hyperparameters, providing inputs, and registering the outputs. To integrate a new SLAM algorithm into SLAMBench, six functions need to be implemented, structured in a three-phase pipeline:

In the **Initialisation Phase**, the initial setup of the SLAM systems is done, as well as testing that the required sensors are available (either live, or dataset-provided). The **Processing Phase** represents the main pipeline: the sb_update_frame function
3.3 Non-rigid 3D reconstruction

To evaluate the quality of non-rigid 3D reconstruction, sequences from the VolumeDeform dataset [61] were included in SLAMBench. The VolumeDeform dataset features
Figure 3.9: The upperbody sequence from the VolumeDeform [61] dataset. (a) Ground-truth reconstructions at frames 0, 100, and 200. (b) Left: reconstruction error heatmap. Right: reconstruction (white) overlayed on ground-truth (coloured).

8 sequences containing non-rigid motion, captured using a PrimeSense RGB-D sensor, described in Table 5.3. Ground-truth reconstructions acquired using the VolumeDeform algorithm are provided every 100 frames, as well as at the last frame.

A number of additions to the core SLAMBench system were required: specifically extending the point cloud sensor class to refer to a specific frame where the ground-truth point cloud was taken. One of most challenging aspects of this change was managing the memory in SLAMBench, and isolating the algorithm timing and memory measurements from the metric computation. Point cloud alignment is a time-consuming process, however in the case of static reconstruction, measurements only need to be done after the Finalisation phase, when the reconstruction is completed. In the case of non-rigid reconstruction, aligning point clouds and computing the reconstruction error every 100 frames initially proved to affect time measurements. On the
3.3. NON-RIGID 3D RECONSTRUCTION

<table>
<thead>
<tr>
<th>Name</th>
<th>Frames</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar</td>
<td>328</td>
<td>221.8MB</td>
<td>Person moving a flat surface.</td>
</tr>
<tr>
<td>Boxing</td>
<td>331</td>
<td>189.1MB</td>
<td>Fist making contact with a second person.</td>
</tr>
<tr>
<td>Hoodie</td>
<td>456</td>
<td>293.4MB</td>
<td>Moving person with a hoodie hiding their face.</td>
</tr>
<tr>
<td>Minion</td>
<td>536</td>
<td>402.6MB</td>
<td>Moving person deforming a toy minion.</td>
</tr>
<tr>
<td>Shirt</td>
<td>507</td>
<td>364.6MB</td>
<td>Person moving a hanging shirt.</td>
</tr>
<tr>
<td>Sunflower</td>
<td>874</td>
<td>516.3MB</td>
<td>Toy sunflower moved. Contains significant occlusions.</td>
</tr>
<tr>
<td>Umbrella</td>
<td>550</td>
<td>360.0MB</td>
<td>Umbrella being deformed and rotated.</td>
</tr>
<tr>
<td>Upperbody</td>
<td>1235</td>
<td>712.6MB</td>
<td>Moving person.</td>
</tr>
</tbody>
</table>

Table 3.1: The VolumeDeform dataset.

On the other hand, keeping both the ground-truth and the reconstructions in memory proved intractable. To solve this issue, we save the reconstructions on the disk, and load all ground-truths and all reconstructions after the Finalisation phases. Users visualising the reconstruction through the Pangolin front-end also have the option to view the error at run-time.

For non-rigid sequences, the mean of the Reconstruction Error over all ground-truth meshes is reported.
Chapter 4

FullFusion: 3D reconstruction in dynamic environments

FullFusion is the first work to address the problem of semantic 3D reconstruction in dynamic environments. Specifically, the aim is to build virtual 3D models of real scenes, and represent:

- The geometry of static scene elements
- The geometry of non-rigidly moving elements, such as humans
- Semantic information about the scene (i.e. assign semantic labels to geometry)

In contrast to many of the currently available systems, FullFusion is designed as a modular framework, rather than a tightly-coupled pipeline, allowing different implementations of the modules.

In this chapter, we will discuss the fundamentals of 3D reconstruction and present an overview of FullFusion as a framework, as well as a baseline implementation.

4.1 Background

4.1.1 RGB-D sensing

Commercial RGB-D devices have become increasingly common in recent years, and are used extensively in SLAM research. The Microsoft Kinect v1 device: a structured-light RGB-D camera operating at 30Hz is commonly used when operating indoors. The Kinect v1 does not work well outdoors, due to infrared interference from direct
4.1. BACKGROUND

Figure 4.1: (a) The Kinect device. (b) A depth frame (colorized). Red indicates close points, blue indicates far points; black pixels are invalid measurements.

sunlight. The Kinect device typically operates at distances ranging between 0.7m to 6m, although it is most accurate in the 1.2m-3.5m range, with an error of up to 4cm at the maximum distance [70].

In addition to colour images, the Kinect device provides depth images, which store distances to the camera center as pixel values. Like other structured-light devices, Microsoft Kinect projects a matrix of infrared light points and uses two CMOS monochrome sensors to infer distances via triangulation. The triangulation process estimates the 3D position of a point images captured simultaneously by two calibrated cameras with a known baseline. To reduce and ensure more uniform noise patterns, methods such as bilinear filtering [125] are often employed as a preprocessing step.

4.1.2 Signed-distance fields

A signed distance field is a function \( \Phi : \mathbb{R}^3 \to \mathbb{R} \) which associates each point in the 3D space a value corresponding to the distance to the closest surface. Points located exactly on an object boundary are zero-valued, while points located outside the object have positive values. Conversely, points contained within an object hold negative values.

To make problems computationally-tractable, a truncated signed distance function (TSDF) is generally used, discretising the space into voxels, which can be conceptually thought of as cubes in space (Figure 4.2).
The problem of reconstructing the geometry of static scenes is well studied, with most solutions adding a dense representation of the map to SLAM systems. One of the crucial processes is estimating the camera pose, which in turn enables data association and fusing data into the map. This requires finding a single transformation to model camera displacement and rotation at every frame. In the case of non-rigid reconstruction, the problem becomes significantly more complex, as it involves estimating not only the camera movement but also the movement in the scene, thus necessitating thousands of transformations to be computed at every frame. Moreover, ambiguities such as occlusions prompt the use of regularisation techniques in order to ensure coherence across frames. Due to the computational demands which arise from such complexity, current non-rigid reconstruction systems suffer from severe scalability limitations.

In a non-rigid reconstruction system, given a base shape $B \in \mathbb{R}^3$ (the model reconstructed up to the previous frame) and a target shape $T \in \mathbb{R}^3$ (the live camera model), a function $\mathcal{W} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ called a Warp field is defined to approximate $\mathcal{W}(B) = T$. To estimate the warp field, the following energy function is minimized using a Gauss-Newton non-linear optimization process:

$$E_{\text{warp}}(\mathcal{W}_t, B, T) = E_{\text{Data}}(\mathcal{W}_{t-1}, B, T) + \lambda E_{\text{ARAP}}(\mathcal{W}_{t-1}, B)$$  \hspace{1cm} (6)
The energy function is described by two terms: the data term $E_{\text{data}}(\mathcal{W}_{t-1}, \mathcal{B}, T)$ is a non-rigid ICP function (N-ICP), measuring the difference between the model and the live frame. As non-rigid registration in $\mathbb{R}^3$ is an inherently ill-posed problem [45], an infinite number of solutions can be found, with no guarantee of consistency between frames. To address this issue, an As-Rigid-As-Possible (ARAP) [119] regularisation term $E_{\text{ARAP}}(\mathcal{W}_{t-1}, \mathcal{B})$ was introduced. $E_{\text{ARAP}}$ operates as a graph over the set of deformation nodes, with all nodes “pulling” together to promote solutions that deviate the least from the model at $t-1$, and thus ensures smooth deformations. Additionally, it enables the prediction of movement in occluded regions, as there is no $E_{\text{data}}$ associated. $\lambda$ is a hyperparameter that controls the rigidity of the warp field.

Most non-rigid reconstruction systems, including DynamicFusion use a global regularisation field, which extends over the whole scene. As scenes grow, it becomes intractable to compute the regularisation parameter. One of the key insights of FullFusion is that a global regularisation field is not only unnecessary, but can negatively impact the reconstruction quality. Instead, warp fields should be defined per-object.

To achieve this, we segment the scene into static and dynamic frames.

### 4.2 FullFusion Overview

![Diagram](image)

Figure 4.3: RGB-D input from a sensor such as Microsoft Kinect is divided into a static and a dynamic frame by the Segmentation module. The Pose Estimation module uses the static frame and a reference frame from the static model to compute the camera position, thus reducing ambiguity between scene dynamics and camera movement. Finally, each of the reconstruction systems receives its processed input, along with the estimated camera pose and semantic labels.

FullFusion is structured into four loosely-coupled modules, for the following processes:
1. Segmentation
2. Pose estimation
3. Static reconstruction
4. Dynamic reconstruction

We present an overview of our pipeline in Figure 4.3. The framework receives a registered RGB-D frame pair \( F_t = \{C_t, D_t\} \) at time \( t \) defined by a colour image \( C_t : \Omega \rightarrow \mathbb{N}^3 \) and a depth image \( D_t : \Omega \rightarrow \mathbb{N} \) where \( \Omega \in \mathbb{N}^2 \) is the image plane. The Segmentation module produces pairs of frames for the static and dynamic parts of the scene: \( F_{t}^{\text{static}} = \{C_{t}^{\text{static}}, D_{t}^{\text{static}}\} \) and \( F_{t}^{\text{dynamic}} = \{C_{t}^{\text{dynamic}}, D_{t}^{\text{dynamic}}\} \), and a label image \( L_t : \Omega \rightarrow \mathbb{R}^{|L|} \) of probabilities with \( |L| \) channels, where \( L \) is the set of labels. The pose estimation module uses \( F_{t}^{\text{static}} \) to compute the pose \( P_t \in SE(3) \), representing the 6-DoF transformation from the camera frame to the world frame. Finally, the static and dynamic reconstruction modules receive their respective RGB-D frames, along with the pose and labels.

The framework specifies an API consisting of abstract interfaces for Segmentation, Pose estimation, and Reconstruction. The constructor of each interface receives the global configuration and any implementation is expected to acquire its initialisation parameters through the global configuration object. Pseudocode is provided below:

```java
class SegmentationInterface {
    // Performs segmentation and stores the results for querying
    SegmentationInterface(config)
    segmentFrame(rgb, depth)
    getStaticFrame() -> static_rgb, static_depth
    getDynamicFrame() -> dynamic_rgb, dynamic_depth
    getSemanticFrame() -> segmentation, probability_map
}

class PoseEstimationInterface {
    PoseEstimator(config)
    getPose(static_frame, reference_frame) -> pose
}
```
4.3. **POSE ESTIMATION**

```java
class ReconstructionInterface {

    ReconstructionInterface (config)

    processFrame (pose, rgb, depth, segmentation, probability_map)

    renderModel (pose) -> reference_frame

}
```

4.3 Pose estimation

The 6-DoF camera pose is estimated using the Iterative Closest Point (ICP) [82] method, as in KinectFusion. ICP aims to minimize the distance between corresponding points in the current depth frame and a reprojection of the current model into the depth frame. Non-rigid motion prevents obtaining good pose predictions using traditional ICP, due to the ambiguity between local changes in non-rigidly moving objects and camera motion, which cannot be explained through a single transformation.

4.4 Segmentation module

Figure 4.4: Our implementation of the *Segmentation* module uses DeepLabv3+ trained on the PASCAL-VOC dataset along with a geometric clustering approach to produce geometrically-consistent semantic segmentation

The segmentation module’s job is to provide the other components with appropriate input to increase their performance. Our implementation is based on the observation that since integrating semantic labels in 3D models is desirable for several applications, priors offered by the semantic labels can be used to reason about scene motion. As shown in Figure 4.4, our implementation combines two approaches to perform semantic segmentation, as well as splitting the input into a static and a dynamic frame. We first use DeepLab v3+ [24] trained on the PASCAL-VOC dataset [42] to obtain

---

1From the Tensorflow GitHub repository
the label image \( L_i \) containing a per-pixel probability distribution over all the recognised classes.

![Figure 4.5: Semantic segmentation without (left) and with (right) depth clustering.](image)

While the semantic segmentation itself is sufficient to segment the scene into static and dynamic parts, the depth input can offer additional geometric priors that can help refine the segmentation. We build on the geometric clustering method introduced by Jaimez et al. [63] to segment the scene into \( K \) clusters using K-means on the depth image. The refined semantic mask is then obtained by labelling each cluster with the dominant semantic label. We first extract a segmentation map \( S : \Omega \rightarrow \mathbb{N} \) by taking the label with the maximum probability for each pixel: \( S = \{ x_j | x_j = \text{argmax}_j(y_j) \}, y_j \in L \). Each cluster is then labelled with the class that occupies the most pixels in the cluster. Finally, neighbouring clusters with the same label are merged to obtain the final segmentation map. The segmentation map is then used as a mask to extract the static and dynamic frame, respectively. Table 4.1 details the movement labels taken into account when separating the static and dynamic elements.

<table>
<thead>
<tr>
<th>Static</th>
<th>Non-rigid</th>
<th>Rigid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>Person</td>
<td>Aeroplane</td>
</tr>
<tr>
<td>Dining table</td>
<td>Bird</td>
<td>Boat</td>
</tr>
<tr>
<td>Bottle</td>
<td>Cat</td>
<td>Bus</td>
</tr>
<tr>
<td>Chair</td>
<td>Dog</td>
<td>Car</td>
</tr>
<tr>
<td>Potted plant</td>
<td>Horse</td>
<td>Motorbike</td>
</tr>
<tr>
<td>Sofa</td>
<td>Sheep</td>
<td>Train</td>
</tr>
<tr>
<td>TV/Monitor</td>
<td></td>
<td>Bicycle</td>
</tr>
</tbody>
</table>

Table 4.1: PASCAL-VOC dataset with movement labels
Since 3D points belonging to different objects may be clustered together, we mitigate this issue by requiring that a class occupies at least 70% of the pixels in a cluster. If a cluster is found not to have a dominant class, it is further split using K-means with $K = 2$.

**Algorithm 1** Segmentation refinement

```latex
# define num_classes = 20
# define min_coverage = 0.7 \Comment{At least 70\% of a cluster for dominant class}

\begin{algorithm}
\begin{footnotesize}
\caption{REFINESEGMENTATION(clusters, segmentation)}
\begin{algorithmic}
\State \textbf{procedure} \text{REFINESEGMENTATION} (clusters, segmentation)\Comment{Map holding (label, cluster)}
\State \hspace{1em} \textbf{map} \text{< int, obj \textgreater} \text{ cluster_map}
\State \hspace{1em} \hspace{1em} \Comment{Label clusters}
\For {cluster: clusters} \Comment{for cluster : clusters do}
\State \hspace{1em} \text{counter}[\text{num\_classes}] = \{0\}
\For {pixel: cluster} \Comment{for pixel : cluster do}
\State \hspace{1em} \hspace{1em} \text{counter}[\text{segmentation[pixel\_x][pixel\_y]]++}
\EndFor
\State \hspace{1em} \text{label} = \arg\max(\text{counter})
\If {label $\geq$ 0.7 * size(cluster)} \Comment{if label >= 0.7 * size(cluster) then}
\State \hspace{1em} \hspace{1em} \text{map.add}(\text{label}, \text{cluster})
\Else
\State \hspace{1em} \hspace{1em} \text{c1, c2 = kmeans(cluster, 2)}
\State \hspace{1em} \hspace{1em} \text{clusters.delete}(\text{cluster})
\State \hspace{1em} \hspace{1em} \text{clusters.push_back}(\text{c1})
\State \hspace{1em} \hspace{1em} \text{clusters.push_back}(\text{c2})
\EndIf
\EndFor
\Comment{Merge adjacent clusters with the same label}
\For {label = 0; label < num\_classes; label++} \Comment{for label = 0; label < num\_classes; label++ do}
\For {base\_cluster: cluster\_map.find(label)} \Comment{for base\_cluster : cluster\_map.find(label) do}
\For {cluster: cluster\_map.find(label)} \Comment{for cluster : cluster\_map.find(label) do}
\If {adjacent(base\_cluster, cluster)} \Comment{if adjacent (base\_cluster, cluster) then}
\State \hspace{1em} \hspace{1em} \text{base\_cluster = merge(base\_cluster, cluster)}
\State \hspace{1em} \hspace{1em} \text{cluster\_map.delete}(\text{cluster})
\EndIf
\EndIf
\EndFor
\EndFor
\EndFor
\State \hspace{1em} \Return \text{cluster\_map}
\EndAlgorithm
\end{algorithmic}
\end{footnotesize}
\end{algorithm}
```
4.5 Implementation details

FullFusion is designed as a generic framework with loosely-connected components. The system is implemented in C++, and only depends on the Eigen library [51], any other dependencies being specific to the implementation of each module. A global configuration file is defined, controlling all hyperparameters for the various modules. All inputs and outputs to functions defined by the interfaces are Eigen matrices (either images or 6-DoF pose in matrix form).

Our implementation adopts KinectFusion [94] for static reconstruction and DynamicFusion [93] for dynamic reconstruction. Although more advanced systems are currently available, a significant number of publications have been influenced by the ideas presented in KinectFusion and DynamicFusion, and as such, this implementation constitutes a good baseline for future evaluation. Our implementation of both KinectFusion and DynamicFusion uses CUDA for GPU acceleration, and OpenGL for rendering and visualisation. We use DeepLabv3+ to perform semantic segmentation.
Figure 4.6: Visualisation of the FullFusion GUI, on two frames from the upperbody sequence. In clockwise order, from top-left: RGB frame, Semantic segmentation (without geometric clustering), Dynamic Reconstruction, Depth Frame.
Chapter 5

Results

5.1 Experimental setup

We evaluate our implementation using the SLAMBench framework [15]. All experiments were performed on a machine with an Intel Core i7-6700HQ CPU with 16GB of memory, and an NVidia GeForce GTX 960M with 4GB VRAM, running Ubuntu 18.04. Unless otherwise noted, all software was compiled using GCCv6.5.0 and CUDA 9.1. Both KinectFusion and DynamicFusion use 1cm voxel sizes and $256^3$ volumes. For DynamicFusion, the decimation density used is 25mm, and we use the same hyperparameters recommended in the publication [93]. As no public implementation of either KinectFusion or DynamicFusion is provided by the authors, we evaluate our own implementation of the two algorithms.

5.2 Trajectory accuracy

One of the important hypotheses of our work is that using only the static component of the scene to estimate the camera pose increases the accuracy. As discussed in Section 2.3, the literature supports this claim.

We evaluate the accuracy of the trajectory estimation using the TUM RGB-D [120] dataset, which provides RGB-D input captured with a Microsoft Kinect device, as well as ground-truth trajectory measurements with static, low dynamic, and highly dynamic sequences. We use the relative pose error (RPE) and absolute trajectory error (ATE) metrics as implemented in SLAMBench. Results presented in Table 5.1, show significant improvements in pose estimation were achieved when using only the static part of the scene, surpassing state-of-the-art methods.
5.2. TRAJECTORY ACCURACY

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sequence</th>
<th>VO-SF</th>
<th>ElasticFusion</th>
<th>StaticFusion</th>
<th>KinectFusion</th>
<th>FullFusion (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>fr1/xyz</td>
<td>2.1</td>
<td>1.9</td>
<td>2.3</td>
<td>3.0</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>fr1/desk</td>
<td>3.7</td>
<td>2.9</td>
<td>3.0</td>
<td>8.2</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>fr1/desk2</td>
<td>5.4</td>
<td>7.2</td>
<td>5.0</td>
<td>6.6</td>
<td>7.9</td>
</tr>
<tr>
<td></td>
<td>fr1/plant</td>
<td>6.1</td>
<td>5.0</td>
<td>10.4</td>
<td>8.2</td>
<td>12.2</td>
</tr>
<tr>
<td>Slightly dynamic</td>
<td>fr3/stat/xyz</td>
<td>2.4</td>
<td>0.9</td>
<td>1.1</td>
<td>2.1</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>fr3/stat/halfsphere</td>
<td>5.7</td>
<td>1.6</td>
<td>2.8</td>
<td>4.4</td>
<td>3.6</td>
</tr>
<tr>
<td>Highly dynamic</td>
<td>fr3/walk_static</td>
<td>10.1</td>
<td>26.0</td>
<td>1.3</td>
<td>13.8</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>fr3/walk/xyz</td>
<td>27.7</td>
<td>24.0</td>
<td>12.1</td>
<td>tracking lost</td>
<td>6.0</td>
</tr>
<tr>
<td></td>
<td>fr3/walk/halfsphere</td>
<td>33.5</td>
<td>20.5</td>
<td>20.7</td>
<td>tracking lost</td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td>fr3/walk/halfsphere*</td>
<td>24.8</td>
<td>16.3</td>
<td>5.0</td>
<td>tracking lost</td>
<td>6.3</td>
</tr>
</tbody>
</table>

(a) Comparison of Relative Pose Error (translational RPE-RMSE)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sequence</th>
<th>VO-SF</th>
<th>ElasticFusion</th>
<th>StaticFusion</th>
<th>KinectFusion</th>
<th>FullFusion (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>fr1/xyz</td>
<td>5.1</td>
<td>1.2</td>
<td>1.4</td>
<td>1.7</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>fr1/desk</td>
<td>5.6</td>
<td>2.1</td>
<td>2.3</td>
<td>3.6</td>
<td>3.7</td>
</tr>
<tr>
<td></td>
<td>fr1/desk2</td>
<td>17.4</td>
<td>5.7</td>
<td>5.2</td>
<td>6.0</td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td>fr1/plant</td>
<td>7.8</td>
<td>5.3</td>
<td>11.3</td>
<td>9.2</td>
<td>9.1</td>
</tr>
<tr>
<td>Slightly dynamic</td>
<td>fr3/stat/xyz</td>
<td>2.9</td>
<td>0.8</td>
<td>1.3</td>
<td>1.7</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>fr3/stat/xyz</td>
<td>11.1</td>
<td>2.2</td>
<td>4.0</td>
<td>3.7</td>
<td>4.3</td>
</tr>
<tr>
<td></td>
<td>fr3/stat/halfsphere</td>
<td>18.0</td>
<td>42.8</td>
<td>4.0</td>
<td>39.6</td>
<td>3.4</td>
</tr>
<tr>
<td>Highly dynamic</td>
<td>fr3/walk_static</td>
<td>32.7</td>
<td>29.3</td>
<td>1.4</td>
<td>79.4</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>fr3/walk/xyz</td>
<td>87.4</td>
<td>90.6</td>
<td>12.7</td>
<td>tracking lost</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>fr3/walk/halfsphere</td>
<td>73.9</td>
<td>63.8</td>
<td>39.1</td>
<td>tracking lost</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>fr3/walk/halfsphere*</td>
<td>48.2</td>
<td>48.6</td>
<td>6.3</td>
<td>tracking lost</td>
<td>2.7</td>
</tr>
</tbody>
</table>

(b) Comparison of Absolute Trajectory Error (ATE)

Table 5.1: Trajectory evaluation results on the freiburg1 and freiburg3 sequences of the TUM RGB-D dataset [120].

We compare our trajectory accuracy against the following algorithms:

1. VO-SF, a system for visual odometry and scene flow developed by Jaimez et al. [63] is a method that computes camera position and is robust to dynamic scenes.


3. ElasticFusion [133], a surfel-based state-of-the-art method for reconstructing static scenes.

4. KinectFusion [94], the method FullFusion uses for static reconstruction and the first real-time RGB-D reconstruction system.

It is worth noting that for VO-SF, ElasticFusion and StaticFusion, we report the results described in the StaticFusion paper [111], and we did not perform the experiments independently. We use SLAMBench to evaluate the results of KinectFusion and FullFusion. In the case of static or slightly dynamic scenes, ElasticFusion tends to perform better than the other systems, as it is designed for high quality static reconstruction. Given the modularity of FullFusion, replacing KinectFusion’s ICP with
alternative formulations, such as the one employed by ElasticFusion for pose estimation would be straightforward. While in some cases, FullFusion performs worse than the other VO-SF, ElasticFusion, and StaticFusion, we attribute this to our pose estimation module implementation, which is a simple ICP algorithm, whereas ElasticFusion uses a joint ICP and photometric error. We note, however, that almost all cases which contain movement, FullFusion performs at least as well as KinectFusion, thus showing that the segmentation improves pose estimation quality. While FullFusion and KinectFusion use the same pose estimation technique, there are noticeable differences in the results on static scenes. These differences arise due to the sequences containing persons which are being segmented out of the frame when computing the pose. An important highlight of the results is that in the more challenging cases, KinectFusion eventually suffers from tracking failure, while FullFusion performs better than any of the other methods. We believe that the reason FullFusion has better performance then VO-SF and StaticFusion is that relying on semantic priors circumvents the need for an initialization period. For the sake of completeness, we show results on fr3_walk_halfsphere with the first 5 seconds skipped (marked as fr3_walk_halfsphere*), as done in StaticFusion.

5.3 Reconstruction accuracy

![Figure 5.1: The ICL-NUIM dataset. (a) Ground-truth 3D model. (b) Challenging, wall-facing colour frame from the kt3 sequence.](image)

For the baseline implementation of FullFusion, a central aim was to capture the geometry and semantic labels of dynamic scenes, with no particular emphasis on obtaining highly accurate reconstructions. Of course, reconstruction accuracy is a problem
of interest for the framework, and as such, an evaluation and discussion were necessary to inform future work directions. Results presented below show accuracy comparable with state-of-the-art algorithms on many of the test sequences, as well as cases which yield high reconstruction error or tracking failures.

We evaluate the accuracy of our reconstruction using two datasets: the newly-integrated VolumeDeform dataset, containing eight sequences with non-rigid movement captured with an RGB-D camera, and the ICL-NUIM dataset, featuring four static sequences, generated from synthetic scenes.

Table 5.2 provides a comparison of FullFusion’s reconstruction accuracy on the ICL-NUIM dataset against the following algorithms:

1. DVO [69], the first dense 3D reconstruction method to use a joint geometric and photometric error for pose estimation.
2. RGB-D SLAM [39], a SLAM method which uses geometry and visual feature tracking.
3. Kintinuous [132], a memory-efficient version of KinectFusion for large scale scenes.
4. BundleFusion [28], a voxel-based method which uses a coarse-to-dense hierarchical tracking algorithm, which continuously optimizes the global trajectory.
5. ElasticFusion (described in 5.2)

Like Kintinuous, our pose estimation does not use colour information, instead relying on depth only. Our results in Table 5.2 highlight one of the drawbacks of this approach: \(kt0\) and \(kt3\) contain sequences where the camera is pointed directly towards a completely flat wall (Figure 5.1b), making it impossible for ICP to track the camera based on geometry. In the case of \(kt3\), significantly more featureless frames, as well as more camera rotation results in greater error.

For the non-rigid sequences, the Reconstruction Error is averaged over all ground-truth reconstructions, using the warped models. The VolumeDeform dataset provides ground-truth reconstructions every 100 frames. Please note that in the case of the VolumeDeform dataset, the ground-truth reconstruction was generated using the VolumeDeform algorithm rather than motion capture hardware, which could provide high-quality reconstructions. As such, our results represent the deviation from the VolumeDeform results.
### 5.2 ICL-NUIM reconstruction error (in cm)

<table>
<thead>
<tr>
<th>Name</th>
<th>Length (frames)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boxing</td>
<td>0.7</td>
</tr>
<tr>
<td>Calendar</td>
<td>1.7</td>
</tr>
<tr>
<td>Hoodie</td>
<td>0.9</td>
</tr>
<tr>
<td>Minion</td>
<td>1.0</td>
</tr>
<tr>
<td>Shirt</td>
<td>2.1</td>
</tr>
<tr>
<td>Sunflower</td>
<td>1.1</td>
</tr>
<tr>
<td>Umbrella</td>
<td>tracking failure</td>
</tr>
<tr>
<td>Upperbody</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Table 5.2: Reconstruction error (in cm) on the VolumeDeform dataset [61] for our DynamicFusion [93] implementation.

#### 5.4 Runtime and memory analysis

To better understand the role of each component, we analyse the run time and memory requirements for the baseline implementation of FullFusion. While not all components have been optimised for resource consumption and minimising processing time, this analysis may allow us to better focus our future efforts towards running the full pipeline in real-time.
5.5. \textit{QUALITATIVE RESULTS}

Table 5.4 presents an overview of the four components of FullFusion, and the associated processing times and GPU memory requirements. As it immediately becomes obvious, the Dynamic Reconstruction module is currently the bottleneck. After looking into why this is the case, it became clear that the non-rigid optimisation in DynamicFusion takes about 400 ms. While this is most likely due to our implementation of DynamicFusion rather than because of the algorithm itself, it is also probable that newer systems will perform better than DynamicFusion, even with a very well optimised implementation. This may be addressed by substituting DynamicFusion with systems such as SurfelWarp \cite{46} or by heuristics such as forcing the system to skip frames according to certain measures of quality, such as how severe the deformations are. The segmentation module also introduces significant overhead, primarily due to the semantic segmentation network. Given that there are lots of open-source semantic segmentation alternatives and the geometric clustering significantly improves overall segmentation, in the future we will consider adopting different networks which focus on inference run time rather than pixelwise accuracy.

\begin{table}[h]
\centering
\begin{tabular}{| c | c | c | c | c |}
\hline
 & Segmentation & Pose est. & Static 3D rec. & Dynamic 3D rec. \\
\hline
Average time per frame & 74ms & 17ms & 38ms & 590ms \\
\hline
GPU memory & 1.3GB & 25MB & 560MB & 2.9GB \\
\hline
\end{tabular}
\caption{Run time and memory usage statistics}
\end{table}

\section{5.5 Qualitative results}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Figure5.2.png}
\caption{Example of failure in KinectFusion reconstruction vs FullFusion reconstruction on the highly dynamic sequences in the TUM RGB-D dataset. Left: input depth; Middle: KinectFusion; Right: FullFusion. Note that the scene is rotated to fit the estimated camera pose, hence the difference in perspective.}
\end{figure}
We qualitatively evaluate our results against KinectFusion as a baseline. Figure 5.2 shows a failure case for KinectFusion, due to movement in the scene while the camera is rotating.

Pose tracking errors result in erroneous reconstruction, causing further erroneous pose estimation, and the reconstruction breaks within less than a second. FullFusion, on the other hand only relies on static geometry for pose estimation.

As previously mentioned, labels are fused into the 3D scene. Figure 5.3 displays extracted geometry and scene labels on the upperbody sequence. Unlabelled or background geometry is displayed in white.

![Figure 5.3: Final 3D reconstruction on the upperbody sequence from the VolumeDeform dataset, with semantic labels.](image)

Many of the sequences in the VolumeDeform dataset contain non-rigidly moving elements which are not recognised by the segmentation module. The hoodie sequence features a person with a hoodie over their face. When testing FullFusion on the hoodie sequence, we found that the person would often be mislabelled as background, or a rigid object class. As such, for this sequence, we used DynamicFusion only, and truncated the depth at 1.5m to include the moving subject only.
5.6 MixedFusion: discussion

As MixedFusion is by far the most similar system to ours, it would be ideal to include an evaluation against FullFusion. Unfortunately, due to the lack of a public implementation, or any results on public datasets, we cannot offer any direct comparison. While we fully acknowledge that a quantitative comparison would be superior, we believe that it is necessary to draw a comparison based on our understanding of their work and the provided video\(^1\).

A summary of the differences between FullFusion and MixedFusion is necessary. MixedFusion is a reconstruction system which uses a formulation that jointly computes the camera pose and segments the scene. On the other hand, FullFusion is a framework which ensures the interaction of loosely-coupled subsystems working together to achieve a more complex goal, while also showing improvements in some of the tasks performed by the subsystems. Secondly, rather than using a purely geometric approach to segment the scene, we leverage both geometry and semantics. Finally, we

\(^1\)Available on IEEE Xplore (https://ieeexplore.ieee.org/document/8241434/media#media)
not only use semantics for segmentation, but also fuse the labels into the reconstruction.

An assumption of S-ICP, used for segmentation in MixedFusion, is that a dynamic objects will occupy a small portion of the scene. As shown above, FullFusion shows good performance on cases such as *fr3_walk_halfsphere*, a sequence where there is significant movement from the very beginning, whereas algorithms segmenting the scene solely based on geometry such as StaticFusion require an initialization period, and thus perform poorly. Considering the inherent ambiguities in geometry, we believe any geometry-based method, including that of MixedFusion would produce similar behaviour.

Further to this, the authors of MixedFusion note that one of the limitations of their system is that since their segmentation pipeline is based on geometry connectivity, dynamic objects cannot be accurately segmented if they are connected with the static scene, and suggest that semantic information can help solve the issue. Our results indicate that a joint semantic and geometric segmentation module achieve good performance.

One of the downsides of using semantic priors to predict movement is that exhaustive labelling of all moving classes may not be possible. Moreover, depending on the context, objects might exhibit different behaviour (*e.g.* indoor plants may be generally static, but outdoor plants will likely be affected by wind). As such, MixedFusion generalises better to any scene movement, as it is not restricted to a finite number of recognised classes. FullFusion could benefit from implementing more robust pose estimation methods such as S-ICP into the segmentation module, or using generic moving object segmentation [30] may increase robustness.
Chapter 6

Conclusion

6.1 Summary

This thesis addresses the problem of semantic reconstruction in dynamic environments, which has a wide spectrum of applications, from robotic navigation and interaction with humans to Virtual Reality and Augmented Reality, to developing low-cost motion capture systems.

We have extended SLAMBench by modifying the core of the framework to support the evaluation of non-rigid 3D reconstruction systems. Furthermore, we integrate the VolumeDeform dataset and provide a baseline implementation of DynamicFusion. These developments mark a significant transition, taking the framework beyond the evaluation of traditional SLAM systems. The contributions were published in ICRA 2019.

A novel framework named FullFusion is introduced, and a baseline implementation provided. Our results show that in addition to performing a more complex overall task than each of its individual components. Pose estimation is improved by using only the static part of the scene, and the system is able to semantically reconstruct both the static and dynamic scene parts.

6.2 Limitations and future work

Our system allows overcoming issues presented by 3D reconstruction systems such as KinectFusion and DynamicFusion. Nonetheless, many of the shortcomings of the individual systems affect FullFusion: relocalisation of dynamic models that exit the scene continues to be a challenge, and we do not address the non-functional aspects
of 3D scenes, such as producing textured models. We have identified and present preliminary work on a few future research directions.

### 6.2.1 Semantic reconstruction

Our current implementation uses a per-voxel probability distribution over the set of labels. This does not scale well with the number of classes, and thus a method similar to SemanticFusion [84] which stores a single label and its probability may be preferable. Additional improvements to the 3D segmentation may include instance segmentation, for example using Mask R-CNN [60] or panoptic segmentation [71], as in the current implementation, two objects of the same class located at a similar distance from the camera could be treated as a single object.

### 6.2.2 Monocular depth estimation

![Figure 6.1: Qualitative comparison of depth estimation using DenseDepth [6]. Left: ground-truth; Right: estimated depth.](image)

Colour-only cameras are far more widespread than RGB-D cameras: mobile phones, laptops, webcams and photo cameras are able to produce colour images. Advances in deep learning have enabled remarkable advancements in monocular depth estimation [10]. We have, therefore, decided to explore the possibility of extending the FullFusion pipeline to perform the same semantic reconstruction task, without structured-light depth input, relying on colour images only instead.

Given the large number of monocular depth estimation models, we relied on surveys [10] and added metrics to evaluate the prediction quality in SLAMBench. Two widely used metrics, introduced by Eigen et al. [38] were adopted in SLAMBench:
6.2. LIMITATIONS AND FUTURE WORK

*absolute relative difference* (ARD) – normalised sum of differences between ground-truth and estimated distances. (smaller is better)

\[
\frac{1}{|T|} \sum_{y \in T} \frac{|y - y^*|}{y^*}
\]  

(8)

*accurate depth percentage* (AD%) – percentage of accurate pixels within threshold, per frame. (larger is better)

\[
\max \left( \frac{y_i}{y_i^*}, \frac{y_i^*}{y_i} \right) = \delta < thr, thr \in \{1.25, 1.25^2, 1.25^3\}
\]  

(9)

As an initial step, run time was not taken into consideration, instead the selection criterion was the performance on the above metrics. In our tests, all frames were preprocessed. DenseDepth [6] was identified to perform best on the indoor NYU RGB-D [92] dataset:

<table>
<thead>
<tr>
<th>AD%</th>
<th>AD% (thr²)</th>
<th>AD% (thr³)</th>
<th>ARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.89</td>
<td>0.98</td>
<td>0.99</td>
<td>0.10</td>
</tr>
</tbody>
</table>

We initially hypothesized that current depth estimation methods are not accurate enough for dynamic reconstruction. When testing DenseDepth using our DynamicFusion implementation, as shown in Figure 6.2, it became evident that the accuracy is, in fact, better than expected. Instead, the tests highlighted that in important issue is the *inconsistency of prediction* across frames. Accumulated small errors gradually lead to the failure of the non-rigid ICP.

Moving forward, we intend to explore additional metrics to measure depth estimation consistency, and adopt video-based estimation methods, such as Struct2Depth [22] rather than frame-by-frame estimation.

As a different direction to explore, end-to-end methods [56, 100] aim to directly reconstruct the underlying geometry from colour images. One argument to support this idea, is that any intermediate representation will introduce errors which accumulate, as well as processing overhead, however such methods offer less control and require careful tuning.
6.2.3 Data representation for dense 3D reconstruction

As previous literature suggests, our experiments confirm that there are methods which perform better than KinectFusion for static reconstruction and DynamicFusion for dynamic reconstruction. Nonetheless, these systems were chosen in our baseline implementation as a reference point for future work which will use state-of-the-art methods.
6.2. LIMITATIONS AND FUTURE WORK

Voxels have been used for representing 3D data in volumes for over 30 years. Thanks to the large body of literature, as well as the mathematical properties of voxels, they are the preferred data representation for most SLAM systems. While very reliable for representing 3D geometry, voxels have disadvantages in the semantic reconstruction paradigm: at a conceptual level, we would like to represent physical objects as entities with a certain geometry and semantic meaning - rather than as a multitude of subvolumes, each with its own probability distribution over a set of classes. From the perspective of resource usage, voxels are also inefficient - in the case of non-rigid reconstruction with DynamicFusion, transforming between TSDF and explicit geometry requires additional computation - and can be memory hungry, when semantics are added.

In the future, we plan on further modularising FullFusion by decoupling the underlying map representation from the SLAM systems. A comparative study using the different representation back-ends with specific SLAM processes will be done to assess reconstruction quality along with efficiency and resource usage. In this direction, autoencoders, such as in SceneCode [139], show promising results in memory efficiency for representing geometry and semantics. Additionally, hybrid approaches should be explored: it is likely that non-rigid reconstruction and static reconstruction should be modelled differently due to their inherent properties: SurfelWarp [46] successfully adapts the DynamicFusion method using surfels, to obtain faster and less memory hungry reconstruction, without any decrease in quality. SurfelWarp shows that surfels are more efficient when deformation graphs are used, bypassing much of the processing done to convert between implicit and explicit representations.

One of difficult problems all current non-rigid reconstruction systems suffer from is lacking the capacity to relocalise \textit{i.e.} allow a deforming model to exit the camera frame and re-enter it in a different position.

6.2.4 Benchmarking

With the contributions presented above, SLAMBench has been extended to support a wider variety of tasks, metrics and datasets. Many of these metrics and tasks are, however, highly correlated: for instance, pose estimation has significant impact on reconstruction quality. We plan to further develop SLAMBench to facilitate ablation analysis by decoupling the measurements. In a future version, the framework might
receive a specification of the metrics which need to be measured, and would automatically produce results which include ablation testing. In the case of the correlation between pose estimation and reconstruction quality, one might compare the reconstruction results using the estimated poses to the one using the ground-truth poses.

The presented FullFusion results indicate the performance of some of the modules, but no quantitative evaluation of the semantic reconstruction in dynamic environments problem can currently be performed. As Wasenmüller et al. [130] notes, the lack of benchmarking tools and datasets for non-rigid reconstruction makes quantitative evaluations and comparisons with other algorithms difficult. We observe that the lack of datasets, as well as metrics to measure the correlations between the modules makes evaluating the overall problem we address intractable, and plan to explore the possibility of building such datasets using modern motion capture systems and semantic labelling tools.

6.3 Final remarks

We see our work in the context of emerging technologies for benchmarking and hyperparameter tuning (e.g HyperMapper [12]), which provide opportunities to tailor SLAM systems to suit specific applications. Our results show that the FullFusion framework is more than the sum of its parts: components provide useful information to each other, improving individual performance. The current implementation selected systems to provide a good baseline implementation to evaluate future research against, rather than to maximise performance: KinectFusion was the first real-time RGB-D reconstruction system; DynamicFusion was the first real-time non-rigid reconstruction system, and both informed a significant number of works.

Having built the groundwork for reproducible evaluation and reconstruction, we hope to explore the full potential of FullFusion: from deployment on robots using low-power modules to employing high-quality subsystems to reconstruct large-scale scenes with multiple moving elements, we are keen on finding the limits of the framework, which we believe to lie far beyond the current state-of-the-art.
Bibliography


[22] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. Depth prediction without
the sensors: Leveraging structure for unsupervised learning from monocular
videos. In Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19),
2019. 57

A. Zeng, and Y. Zhang. Matterport3D: Learning from RGB-D data in indoor
environments. International Conference on 3D Vision (3DV), 2017. 29

with atrous separable convolution for semantic image segmentation. In ECCV,
2018. 41

K. Olukotun, C. Ré, and M. Zaharia. Dawnbench : An end-to-end deep learning
benchmark and competition. 2017. 24

[26] 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, pages 303–312. ACM, 1996. 19

of the IEEE Conference on Computer Vision and Pattern Recognition, pages
5828–5839, 2017. 23, 31, 76

Real-time globally consistent 3d reconstruction using on-the-fly surface re-

[29] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. Scancom-
plete: Large-scale scene completion and semantic segmentation for 3d scans. In

moves. CoRR, abs/1902.03715, 2019. 54

[31] A. J. Davison. Real-time simultaneous localisation and mapping with a single
Conference on, pages 1403–1410. IEEE, 2003. 18


[73] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. 6, 38


[77] H. Li, B. Adams, L. J. Guibas, and M. Pauly. Robust single-view geometry and motion reconstruction. ACM Transactions on Graphics (ToG), 28(5):175, 2009. 21

[78] H. Li, J. Yu, Y. Ye, and C. Bregler. Realtime facial animation with on-the-fly correctives. 21


[82] K.-L. Low. Linear least-squares optimization for point-to-plane icp surface registration. 5, 29, 30, 41


[97] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. 21


[99] S. A. Papert. The summer vision project. 1966. 13


[119] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. 39


[125] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. 37


Appendices
Appendix A

SLAMBench artifacts

For reference, we provide a list of datasets and algorithms included in SLAMBench 3.0, as well as a sample of the output generated by the benchmark loader front-end in SLAMBench.

<table>
<thead>
<tr>
<th>Name</th>
<th>Sensors</th>
<th>Trajectory</th>
<th>3D Point Cloud</th>
<th>2D semantic labels</th>
<th>Non-rigid</th>
<th>Synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICL-NUIM [58]</td>
<td>RGB-D</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>TUM RGB-D [120]</td>
<td>RGB-D, IMU</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>InteriorNet [79]</td>
<td>RGB-D, IMU</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ICL [108]</td>
<td>RGB-D, IMU</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>EuRoC MAV [18]</td>
<td>Stereo, IMU</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>NYU RGB-D-v2* [92]</td>
<td>RGB-D</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ScanNet* [27]</td>
<td>RGB-D</td>
<td>Yes</td>
<td>Yes, semantic</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VolumeDeform* [61]</td>
<td>RGB-D</td>
<td>No</td>
<td>Yes</td>
<td>Yes, partial</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table A.1: List of datasets. * denotes datasets introduced in SLAMBench 3.0.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Type</th>
<th>Sensors</th>
<th>Implementations</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORB-SLAM [87]</td>
<td>Sparse</td>
<td>RGB-D, Stereo, Monocular</td>
<td>C++</td>
<td>2016</td>
</tr>
<tr>
<td>OKVIS [76]</td>
<td>Sparse</td>
<td>Stereo, IMU</td>
<td>C++</td>
<td>2015</td>
</tr>
<tr>
<td>SVO [44]</td>
<td>Sparse</td>
<td>Monocular</td>
<td>C++</td>
<td>2014</td>
</tr>
<tr>
<td>MonoSLAM [33]</td>
<td>Sparse</td>
<td>Monocular</td>
<td>C++, OpenCL</td>
<td>2007</td>
</tr>
<tr>
<td>PTAM [72]</td>
<td>Sparse</td>
<td>Monocular</td>
<td>C++</td>
<td>2007</td>
</tr>
<tr>
<td>BundleFusion [28]*</td>
<td>Dense</td>
<td>RGB-D</td>
<td>CUDA</td>
<td>2016</td>
</tr>
<tr>
<td>ElasticFusion [133]</td>
<td>Dense</td>
<td>RGB-D</td>
<td>CUDA</td>
<td>2015</td>
</tr>
<tr>
<td>InfiniTAM [64]</td>
<td>Dense</td>
<td>RGB-D</td>
<td>C++, OpenMP, CUDA</td>
<td>2015</td>
</tr>
<tr>
<td>KineFusion [94]</td>
<td>Dense</td>
<td>RGB-D</td>
<td>C++, OpenMP, OpenCL, CUDA</td>
<td>2011</td>
</tr>
<tr>
<td>LSD-SLAM [40]</td>
<td>Semi-Dense</td>
<td>Monocular</td>
<td>C++</td>
<td>2014</td>
</tr>
<tr>
<td>SemanticFusion [84]*</td>
<td>Dense, semantic</td>
<td>RGB-D</td>
<td>CUDA</td>
<td>2016</td>
</tr>
<tr>
<td>ORB-SLAM2-CNN [107]*</td>
<td>Sparse, semantic</td>
<td>Monocular</td>
<td>C++</td>
<td>2018</td>
</tr>
<tr>
<td>DynamicFusion [93]*</td>
<td>Dense, non-rigid</td>
<td>RGB-D</td>
<td>CUDA</td>
<td>2015</td>
</tr>
<tr>
<td>PLaME [49]*</td>
<td>Depth estimation</td>
<td>Monocular</td>
<td>C++</td>
<td>2017</td>
</tr>
</tbody>
</table>

Table A.2: SLAM algorithms included. * denotes algorithms introduced in SLAMBench 3.0
> ./build/bin/benchmark_loader -i ./datasets/TUM/freiburg1/rgbd_dataset_freiburg1_rpy.slam \
- load ./build/lib/liborbslam2-original-library.so

Properties:
=================
frame-limit: 0
log-file: orbslam2_freiburg1_rpy.log
input: ./datasets/TUM/freiburg1/rgbd_dataset_freiburg1_rpy.slam
load-library: ./build/lib/liborbslam2-original-library.so
dse: false
negative-focal-length: false
realtime-mode: false
realtime-multiplier: 1
Depth-intrinsics-parameters: 0.9235938, 1.229375, 0.5171875, 0.4875
Depth-disparity-params: 0.001, 0
Grey-intrinsics-parameters: 0.8082812, 1.076042, 0.4978125, 0.531875
RGB-intrinsics-parameters: 0.8082812, 1.076042, 0.4978125, 0.531875
mode: auto
vocabulary: ./benchmarks/orbslam2/src/original/Vocabulary/ORBvoc.txt
max-features: 1000
scale-levels: 8
scale-factor: 1.2
initial-fast-threshold: 20
second-fast-threshold: 7
camera-fps: 40
depth-threshold: 40
<table>
<thead>
<tr>
<th>Frame_no</th>
<th>Timestamp</th>
<th>AbsoluteError</th>
<th>MeanATE</th>
<th>MaxATE</th>
<th>RPE, RMSE</th>
<th>Duration_Frame</th>
<th>CPU_Memory</th>
<th>GPU_Memory</th>
<th>CUDA_Memory</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1305031229.5287</td>
<td>0.0606</td>
<td>0.0606</td>
<td>0.0606</td>
<td>-nan</td>
<td>0.0250</td>
<td>348433053</td>
<td>0</td>
<td>402718720</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>1305031229.5644</td>
<td>0.0556</td>
<td>0.0581</td>
<td>0.0606</td>
<td>0.0475</td>
<td>0.0354</td>
<td>358129619</td>
<td>0</td>
<td>402718720</td>
<td>-0.0099</td>
<td>-0.0025</td>
<td>0.0135</td>
</tr>
<tr>
<td>3</td>
<td>1305031229.5966</td>
<td>0.0737</td>
<td>0.0633</td>
<td>0.0737</td>
<td>0.0541</td>
<td>0.0628</td>
<td>358812707</td>
<td>0</td>
<td>402718720</td>
<td>0.0084</td>
<td>-0.0178</td>
<td>-0.0016</td>
</tr>
<tr>
<td>4</td>
<td>1305031229.6287</td>
<td>0.0699</td>
<td>0.0639</td>
<td>0.0737</td>
<td>0.0461</td>
<td>0.0836</td>
<td>361648651</td>
<td>0</td>
<td>402718720</td>
<td>-0.0021</td>
<td>-0.0086</td>
<td>0.0068</td>
</tr>
<tr>
<td>5</td>
<td>1305031229.6646</td>
<td>0.0773</td>
<td>0.0666</td>
<td>0.0773</td>
<td>0.0443</td>
<td>0.0478</td>
<td>361473309</td>
<td>0</td>
<td>402718720</td>
<td>0.0114</td>
<td>-0.0286</td>
<td>0.0084</td>
</tr>
<tr>
<td>6</td>
<td>1305031229.6968</td>
<td>0.0783</td>
<td>0.0686</td>
<td>0.0783</td>
<td>0.0397</td>
<td>0.0993</td>
<td>359451671</td>
<td>0</td>
<td>402718720</td>
<td>0.0128</td>
<td>-0.0270</td>
<td>0.0088</td>
</tr>
<tr>
<td>7</td>
<td>1305031229.7290</td>
<td>0.0737</td>
<td>0.0693</td>
<td>0.0783</td>
<td>0.0369</td>
<td>0.0611</td>
<td>362087327</td>
<td>0</td>
<td>402718720</td>
<td>0.0001</td>
<td>-0.0188</td>
<td>0.0108</td>
</tr>
<tr>
<td>8</td>
<td>1305031229.7648</td>
<td>0.0827</td>
<td>0.0710</td>
<td>0.0827</td>
<td>0.0354</td>
<td>0.0456</td>
<td>361930809</td>
<td>0</td>
<td>402718720</td>
<td>0.0155</td>
<td>-0.0261</td>
<td>0.0084</td>
</tr>
<tr>
<td>9</td>
<td>1305031229.7969</td>
<td>0.0853</td>
<td>0.0726</td>
<td>0.0853</td>
<td>0.0337</td>
<td>0.0967</td>
<td>360145907</td>
<td>0</td>
<td>402718720</td>
<td>0.0109</td>
<td>-0.0281</td>
<td>0.0081</td>
</tr>
<tr>
<td>10</td>
<td>1305031229.8256</td>
<td>0.0878</td>
<td>0.0741</td>
<td>0.0878</td>
<td>0.0320</td>
<td>0.0477</td>
<td>360301423</td>
<td>0</td>
<td>402718720</td>
<td>0.0117</td>
<td>-0.0277</td>
<td>0.0065</td>
</tr>
<tr>
<td>11</td>
<td>1305031229.8630</td>
<td>0.0842</td>
<td>0.0750</td>
<td>0.0878</td>
<td>0.0307</td>
<td>0.0423</td>
<td>362304597</td>
<td>0</td>
<td>402718720</td>
<td>0.014</td>
<td>-0.0250</td>
<td>0.0109</td>
</tr>
<tr>
<td>12</td>
<td>1305031229.8969</td>
<td>0.0853</td>
<td>0.0759</td>
<td>0.0878</td>
<td>0.0309</td>
<td>0.1071</td>
<td>360925335</td>
<td>0</td>
<td>402718720</td>
<td>0.0061</td>
<td>-0.0204</td>
<td>0.0077</td>
</tr>
<tr>
<td>13</td>
<td>1305031229.9262</td>
<td>0.0878</td>
<td>0.0769</td>
<td>0.0878</td>
<td>0.0301</td>
<td>0.0680</td>
<td>363997089</td>
<td>0</td>
<td>402718720</td>
<td>0.0121</td>
<td>-0.0230</td>
<td>0.0106</td>
</tr>
<tr>
<td>14</td>
<td>1305031229.9662</td>
<td>0.0892</td>
<td>0.0776</td>
<td>0.0892</td>
<td>0.0294</td>
<td>0.0630</td>
<td>364247351</td>
<td>0</td>
<td>402718720</td>
<td>0.0183</td>
<td>-0.0254</td>
<td>0.0145</td>
</tr>
<tr>
<td>15</td>
<td>1305031229.9979</td>
<td>0.0791</td>
<td>0.0777</td>
<td>0.0892</td>
<td>0.0297</td>
<td>0.0323</td>
<td>361521151</td>
<td>0</td>
<td>402718720</td>
<td>0.0224</td>
<td>-0.0165</td>
<td>0.0258</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>707</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>708</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>709</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>710</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>711</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>712</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>713</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>714</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>715</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>716</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>717</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>718</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>719</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>720</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>721</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>722</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.3: Output of ORB-SLAM2 run under SLAMBench with benchmark_loader on the freiburg1 rpe sequence of the TUM Dataset [120]