Robust Dense Visual SLAM Using Sensor Fusion and Motion Segmentation

Raluca Scona

SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

HERIOT-WATT UNIVERSITY

School of Engineering and Physical Sciences

AWARDED JOINTLY WITH
THE UNIVERSITY OF EDINBURGH

December 2019

Abstract

Visual simultaneous localisation and mapping (SLAM) is an important technique for enabling mobile robots to navigate autonomously within their environments. Using cameras, robots reconstruct a representation of their environment and simultaneously localise themselves within it. A *dense* visual SLAM system produces a high-resolution and detailed reconstruction of the environment which can be used for obstacle avoidance or semantic reasoning.

State-of-the-art dense visual SLAM systems demonstrate robust performance and impressive accuracy in ideal conditions. However, these techniques are based on requirements which limit the extent to which they can be deployed in real applications. Fundamentally, they require constant scene illumination, smooth camera motion and no moving objects being present in the scene. Overcoming these requirements is not trivial and significant effort is needed to make dense visual SLAM approaches more robust to real-world conditions.

The objective of this thesis is to develop dense visual SLAM systems which are more *robust* to real-world *visually challenging* conditions. For this, we leverage sensor fusion and motion segmentation for situations where camera data is unsuitable.

The first contribution is a visual SLAM system for the NASA Valkyrie humanoid robot which is robust to the robot's operation. It is based on a sensor fusion approach which combines visual SLAM and leg odometry to demonstrate increased robustness to illumination changes and fast camera motion.

Second, we research methods for robust visual odometry in the presence of moving objects. We propose a formulation for joint visual odometry and motion segmentation that demonstrates increased robustness in scenes with moving objects compared to state-of-the-art approaches.

We then extend this method using inertial information from a gyroscope to compare the contributions of motion segmentation and motion prior integration for ro-

bustness to scene dynamics. As part of this study we provide a dataset recorded in scenes with different numbers of moving objects.

In conclusion, we find that both motion segmentation and motion prior integration are necessary for achieving significantly better results in real-world conditions. While motion priors increase robustness, motion segmentation increases the accuracy of the reconstruction results through filtering of moving objects.

To my family and partner.

Acknowledgements

I would like to thank my supervisors Dr. Maurice Fallon and Prof. Yvan Petillot for introducing me to Robotics and for their guidance, availability and support.

I am grateful to the SLMC group at the University of Edinburgh working on the Valkyrie project for their support during experiments and data collection, in particular to Wolfgang Merkt and Vladimir Ivan.

Thanks to Prof. Daniel Cremers and the Computer Vision Group at the Technical University of Munich for welcoming me as a visitor and for their kind words towards my work. Thanks in particular to my collaborator Mariano Jaimez Tarifa and to my office mate Robert Maier for their friendship that manifested through Karaoke sessions.

Thank you to my colleagues, namely Christian Rauch, Simona Nobili, Georgi Tinchev and Marcelo Saval Calvo who helped me with data collection and experimental setups. My projects would not have been possible without you. Similarly, I would like to thank Jan Stankiewicz for taking the time to show me how to use the Vicon system.

Thanks to my CDT cohort at the Edinburgh Centre for Robotics for your friendship and support, in particular to Iris Kyranou and James Garforth. All of the coffee and nachos were had. Also thanks to Radim Tyleček for his pleasant company during our lunches. I would like to thank Elizabeth Vargas Vargas for her friendship and for taking the time to help me with the thesis submission procedures.

Thanks to NVIDIA and Prof. Dieter Fox for welcoming me as an intern to their Robotics Research Lab in Seattle and to Arsalan Mousavian for teaching me Deep Learning. Thanks to Yashraj Narang, Kei Kase, Chris Paxton for the hiking trips, Brenda Tyler for the hipster smoothies and Ankur Handa for the walks.

I would like to extend my gratitude to my friends outside work, namely Migle Graužinytė, Olga Baranova, Darie Picu and Josie Shek, for their support and extended sessions of avocado toast.

Evidently, I am very grateful to my partner Christian Börmel who supported me through many moments of doubt and who was so generous with his time. This thesis would not have been possible without you. Also thank you for our US road trip, it was one of the highlights of my life.

Thank you to the Edinburgh Centre for Robotics and EPSRC for funding my research and enabling my diverse and international learning experience.

To my family, a few words in Romanian:

Mulţumesc familiei mele pentru răbdarea, sprijinul emoţional şi financiar acordat de-a lungul vieţii mele.

HERIOT WATT UNIVERSITY

Research Thesis Submission

Name*:*	*Raluca Scona*		
School:	School of Engineering and Physical Sciences		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought:	*PhD*

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1. The thesis embodies the results of my own work and has been composed by myself
2. Where appropriate, I have made acknowledgement of the work of others
3. The thesis is the correct version for submission and is the same version as any electronic versions submitted*.
4. My thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5. I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.
6. I confirm that the thesis has been verified against plagiarism via an approved plagiarism detection application e.g. Turnitin.

ONLY for submissions including published works

7. Where the thesis contains published outputs under Regulation 6 (9.1.2) or Regulation 43 (9) these are accompanied by a critical review which accurately describes my contribution to the research and, for multi-author outputs, a signed declaration indicating the contribution of each author (complete)
8. Inclusion of published outputs under Regulation 6 (9.1.2) or Regulation 43 (9) shall not constitute plagiarism.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate*:*	Scona	Date:	03/04/2020

Submission

Submitted By *(name in capitals)*:	RALUCA SCONA
Signature of Individual Submitting:	Scona
Date Submitted:	03/04/2020

For Completion in the Student Service Centre (SSC)

Limited Access	Requested	Yes		No		Approved	Yes		No
*E-thesis Submitted (**mandatory for final theses**)*									
Received in the SSC by *(name in capitals)*:						Date:			

Inclusion of Published Works

Declaration

This thesis contains one or more multi-author published works. In accordance with Regulation 6 (9.1.2) I hereby declare that the contributions of each author to these publications is as follows:

Citation details	Direct Visual SLAM Fusing Proprioception for a Humanoid Robot R. Scona, S. Nobili, Y. R. Petillot, M.Fallon IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2017
R. Scona	Main author on literature review, technical contribution, experimental set-up, evaluation and paper content.
S. Nobili	Assistance with data collection and paper content.
Y.R. Petillot, M. Fallon	Assistance with paper content.
Signature:	Scona
Date:	03/04/2020

Citation details	StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, D. Cremers IEEE International Conference on Robotics and Automation (ICRA) 2017
R. Scona	Main author on literature review, technical contribution, experimental set-up, evaluation and paper content.
M. Jaimez	Assistance with technical contribution, evaluation and paper content.
Y. R. Petillot, M. Fallon, D. Cremers	Assistance with paper content.
Signature:	Scona
Date:	03/04/2020

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction

Chapter 1

Introduction

Simultaneous localisation and mapping (SLAM) is a well-studied problem within the fields of Robotics and Computer Vision. It is a fundamental capability for achieving autonomy in mobile robots:

A robot should use its on-board sensors to build an internal model of its environment and simultaneously use this model to track its own location in real-time.

Within this thesis we focus on *visual* SLAM, meaning we address the problem of SLAM using cameras, as these are widely available and low-cost sensors. In particular, we research techniques for stereo and RGB-D cameras.

Despite the long history of visual SLAM, few techniques are robust enough to function reliably in real-world conditions, which results in limited commercial applications of SLAM in robots or other embedded devices.

The work in this thesis focuses on achieving *robust* visual SLAM operation in *visually challenging* situations. For this, we leverage sensor fusion and motion segmentation for situations where camera data is unsuitable.

We begin this chapter by highlighting notable applications of visual SLAM both in academic and industrial projects, which illustrates the broad applicability of this field. Then, we discuss the basic building blocks of visual SLAM and their evolution in time, from sparse to dense approaches. This is followed by a discussion of the fundamental limitations of visual SLAM and the methods researchers have attempted to tackle them. Finally, we state the contributions of our research and outline the in-depth chapters of this thesis.

1.1 Examples of Visual SLAM Implementations

Visual SLAM is a versatile and low-cost technology that has been successfully implemented within research projects across different domains. In this section we provide some examples of the situations where visual SLAM has been used.

1.1.1 Research Projects

In Table 1.1.1 we give examples of applications which used visual SLAM within research projects. We describe approaches that span a multitude of robotics domains, including space, marine, aerial, autonomous driving, humanoid robots and medicine.

Domain	Applications
Space	Maimone *et al*. [1] demonstrated the first implementation of vision-based motion estimation applied to exploration on Mars with rovers, with the purpose of detecting and handling wheel slippage during driving. In related work, Olson *et al*. [2] propose a technique for Mars terrain reconstruction using cameras mounted on rovers.
Marine	Visual SLAM is also used for autonomous underwater robotics applications, such as reconstructing 3D models of the sea floor [3] or structures such as wrecks [4] or caves [5]. Due to poor visibility conditions underwater, researchers in this domain often fuse vision with additional sensors such as sonars [6].
Aerial	There are numerous applications of visual SLAM in autonomous drones. As cameras are lightweight sensors and visual SLAM can be sufficiently light in computation, drones rely on such systems to localise themselves within their environment [7]. Examples include collaborative SLAM techniques which enable multiple drones to share a common world representation [8] [9] and methods for dense surface reconstructions used for inspection and manipulation [10] [11]. Recent innovations in this domain include the use of event cameras for handling the high velocities of drones which tend to induce significant motion blur in regular cameras [12].

Automotive	Many visual SLAM approaches are implemented on land-based robots. The 2005 DARPA Grand Challenge[1] was an autonomous driving competition that sparked significant subsequent investment in the development of driver-less cars. Within this competition, cars had to autonomously navigate 244km over desert terrain within the Mojave Desert. Cars tracked their locations on the path using GPS (global positioning system) and used sensors such as cameras, radars and LIDARs (Light Detection and Ranging) to compute maps of their surrounding terrains in order to identify obstacles [13] [14] [15].
Humanoids	In 2012, the nuclear disaster at the Fukushima Daiichi Power Plant in Japan resulted in an environment that was too dangerous for human responders. DARPA issued a new Robotics Challenge[2] to develop ground robots that could operate in hazardous human-engineered environments and perform complex tasks semi-autonomously. Tasks included driving a vehicle, walking over uneven terrain, opening doors, climbing a ladder and manipulating objects or tools typically used by humans. Many teams developed humanoid robot designs where state estimation is implemented through sensor fusion of leg kinematics and IMUs (inertial measurement units) and 3D terrain mapping using LIDARs and cameras [16] [17] [**?**].
Medicine	Visual SLAM can also find uses in medicine, particularly in endoscopy, which is the process of examining a particular organ or cavity using an endoscope that has a light source and a camera attached. In this case, visual SLAM is used to estimate the pose of the endoscope and build a 3D reconstruction of the organ or cavity which is being inspected [18] [19] [20].

1.1.2 Commercial Projects

Implementations of Visual SLAM have started to appear in consumer-oriented commercial products as well. We mention here (Table 1.1.2) prominent examples from the Robotics and Virtual/Augmented Reality (AR/VR) domains.

[1]https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles
[2]https://www.darpa.mil/program/darpa-robotics-challenge

Domain	Applications
Robotics	Robotics applications of visual SLAM includes the iRobot Roomba[3] vacuum cleaner, which can build a 2D reconstruction of an apartment and track its location within it. This way, it can be programmed to only vacuum particular rooms or areas of the house.
AR/VR	Virtual or augmented reality headsets such as the Oculus Quest[4] or the Microsoft Hololens[5] implement so-called ”inside-out tracking” - they track the motion of the headset in order to correctly render views of the virtual scene from the point of view expected by the user. These devices implement room-scale 3D visual SLAM solutions. High level accuracy is achieved by installing several cameras within the headset to maximise the field of view.

1.2 A Review of Visual SLAM

The problem of using images to build a model of the environment and estimate the motion of the camera has been tackled through two main fields of research:

1. The term *Visual SLAM*, originating from the Robotics community, implies that the input is a video stream and the computation is expected to run in real-time on the robot hardware [21].

2. Within the Computer Vision community, researchers have tackled the problem of *Structure from Motion* (SfM), where the input is typically a set of unordered images (potentially taken with different cameras) and without the requirement for real-time operation [22].

A visual SLAM system is a tightly coupled combination of a number of components:

- *Visual odometry* estimates the incremental motion of the camera from the most recent stream of consecutive images. The minimal case is a pair of images. It is related to the concept of *localisation* in the sense that it tracks the motion of the camera with respect to a local reference frame. However, as it is an incremental process, it suffers from unbounded drift over time.

[3]https://www.irobot.co.uk/roomba
[4]https://www.oculus.com
[5]https://www.microsoft.com/en-us/hololens

- *Mapping* is the process of estimating a model of the environment. This also includes design decisions regarding how to represent this model in memory.
- *Loop closure* recognises an already visited scene. It is an important component as it is the main approach to correct drift from visual odometry.
- *Global localisation* implements solutions to handle the "kidnapped robot" problem, where the robot is placed in an unknown location within a previously reconstructed environment. The task is to identify the current location of the robot within the environment to resume tracking it.

To achieve real-time operation, a common strategy among many visual SLAM systems is to sub-divide the estimation process into two main components [23, 24, 25]:

1. *Local estimation*: for estimating the local camera pose and map in real time, a process which drifts over time.
2. *Global estimation*: which keeps in memory a larger history of the visited scenes as well as the past poses of the camera. Once a loop closure is detected, this process re-estimates the whole map and the trajectory of the camera, correcting drift. This process is typically slower than real-time.

These processes would run on separate threads, with the local estimation expected to run at frame-rate while the global estimation is typically slower than frame-rate but expected to produce a globally consistent camera trajectory and reconstruction.

Next, we discuss how the original sparse Visual SLAM systems were designed. As this thesis is concerned with dense systems, we will then review how modern dense Visual SLAM systems differ in their design compared to the sparse counterparts.

1.2.1 Sparse Visual SLAM

Sparse visual SLAM is one of the original formulations of visual SLAM. The term *sparse* means that these approaches reconstruct maps with a small set of 3D landmarks which are represented as points [27] [24]. An example of a sparse reconstruction can be seen in Figure 1.1 (centre). While these maps are useful for localisation, they do not store information about occupied space. Nevertheless, they are efficient representations which require little memory. Many such SLAM systems work in real-time on a single CPU.

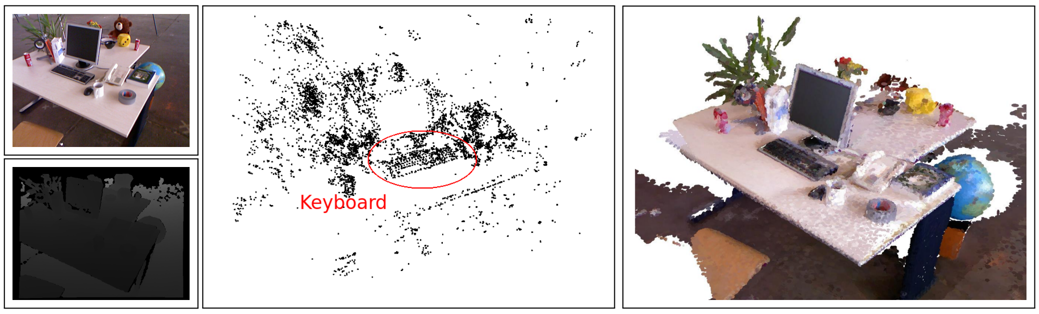


Figure 1.1: **Left**: RGB (top) and Depth (bottom) sample input images of a desk from the TUM/Freiburg dataset [26]. **Centre**: Sparse reconstruction of the desk obtained with ORB-SLAM [24]. **Right**: Dense surfel-based reconstruction of the desk obtained with proposed system StaticFusion described in Chapter 4.

1.2.1.1 Sparse Visual Odometry

In sparse visual odometry, camera motion is estimated from a few pairs of corresponding features or pixels in the input images.

Feature-based techniques take as input raw images and extract features such as SIFT [28] or the more efficient options, ORB [29] or FAST [30] from these images. A global search procedure then matches features from one image with corresponding features from the other. The camera pose is then computed by minimising the error of reprojecting features from one image onto the other [31, 23, 32].

Direct approaches instead make use of the raw pixel values without requiring the computation of features. Here, the camera motion is estimated by repeatedly warping one of the input images towards the other until the per-pixel intensity difference between the two images is minimised [33, 34].

1.2.1.2 Local Estimation

Filtering In Robotics, SLAM was originally formulated as a filtering problem and several methods proposed Extended Kalman Filter (EKF) - based solutions, where the current pose of the robot and all landmark locations are estimated in a joint probability represented by a state vector and covariance [35] [36]. As a new image is received, the former camera pose is marginalised out while the landmark poses are retained. MonoSLAM [21] was the first visual SLAM system to apply this strategy, demonstrating real-time incremental tracking and mapping of a room-scale environment with a single hand-held camera.

Optimisation Techniques inspired by SfM propose solutions based on solving non-linear cost functions, such as the landmark reprojection error. Here, the task is to solve for the camera poses and landmark 3D locations which most accurately predict the reprojected 2D pixel coordinates of the landmarks within all input images. This formulation is also known as *bundle adjustment*. To run in real-time, SfM-based techniques typically implement *sliding-window* bundle adjustment by using only a recent history of previously seen frames, called *keyframes* [23] [37].

Research by Strasdat *et al.* [38] indicates that for real-time operation, better accuracy is obtained by solving for many landmarks rather than many poses. Optimisation techniques handle this by dropping redundant frames, while filtering approaches must maintain large non-sparse covariance matrices encoding correlations between landmark positions. Overall, they conclude that optimisation techniques outperform filtering in terms of accuracy for the same computational cost.

1.2.1.3 Global Estimation and Loop Closure Implementations

Within visual SLAM applications, one common way of recognising loop closures is by implementing place recognition approaches [39] [40]. Features are computed for each individual frame and a "bag-of-words" database is built to contain images and associated feature descriptors. Place recognition is performed by computing the feature descriptors for the current frame and querying the database in order to determine whether an existing frame with similar features has been seen before.

Once a place has been recognised, *global* bundle adjustment is performed incorporating the new loop closure constraint. This results in a globally consistent map and camera trajectory [24].

1.2.2 Dense Visual SLAM

In recent years, advances have been made in the field of dense visual SLAM, supported by the arrival of low-cost RGB-D cameras such as the Microsoft Kinect or the Asus Xtion Pro Live. Such sensors provide as input registered colour and depth images, typically at a rate of 30Hz. The field of view is typically rather low, around 50 degrees, with usual resolutions of 640x480. For a more detailed discussion of the capabilities of these sensors, please consult Section 2.2.1.

These sensors proved very useful for the development of *dense* visual SLAM, as they could be used for building dense 3D models. An example of a dense reconstruction can be seen in Figure 1.1 (right). As opposed to maps consisting of keyframes and sparse landmarks, dense 3D models are useful not only for localisation but also for obstacle avoidance and semantic reasoning about the environment. For example, McCormac *et al*. [41] build a dense 3D reconstruction where every element in the map is also attributed a semantic label of a particular object. Other approaches propose building object-centred reconstructions by representing the environment as a graph with nodes consisting of objects which are densely reconstructed [42] [43], and edges constraining their poses relative to each other. Using such reconstructions, a robot can reason about the different objects within its environment and manipulate them.

Next, we will summarise some of the important techniques for representing these dense 3D models. Then we will discuss notable design decisions which differentiate dense visual SLAM techniques from the classical sparse keyframe-based approaches mentioned so far.

1.2.2.1 Representations for Dense 3D Models

This section summarises some of the most prominent methods for representing dense 3D models, including a discussion of their benefits and weaknesses.

Voxel-based Representations discretise the space into cells which contain information about occupancy. Their advantage is that they encode local connectivity, however the discretisation procedure implies that sub-voxel information is lost. Examples of such representations include:

- **Occupancy Grid** maps are one of the earliest methods for representing environments as occupied or clear space [44, 45]. This method consists of discretising the space into individual cells of fixed sizes, where each cell indicates the probability that it is either fully clear or fully occupied. It has significant drawbacks, particularly it requires a large amount of memory for storing empty space.
- **Octree** representations are more efficient as they recursively subdivide the space into octants according to the resolution necessary to represent the scene [46, 47, 48]. Voxels containing fine structure can be subdivided to the required resolution while neighbouring voxels containing empty space can be merged into a large block.

- **Signed Distance Function (SDF)** representations use each cell to store the signed distance to the closest surface [49]. An important benefit is that the location of the surface can be extracted by performing interpolation between the closest outer and inner voxels, resulting in sub-voxel accuracy for the 3D reconstruction. Thus, it makes it possible to represent fine structures without being limited to the resolution of the voxels. It is the current state-of-the-art representation for high quality 3D reconstructions [50] [51].

Surfels are point-based representations. They represent small discs with a specific position, normal, colour and radius [52]. They are an efficient representation both in terms of memory and rendering speed and which can be interpreted directly [53] [54]. The disadvantage is that they do not encode local connectivity information and result in generally noisier reconstructions compared to those obtained using signed distance functions. However, they allow for an easier representation of non-rigidly deforming objects, as each surfel can be independently transformed with its own motion [55].

1.2.2.2 Large Scale Dense 3D Mapping

Many dense RGB-D SLAM systems have been developed in the last 10 years, with a seminal technique being KinectFusion [56]. It was the first method to implement real-time dense tracking and fusion of depth data using the SDF representation. The approach used a volumetric representation of fixed size integrated on a GPU, which made its application limited to small-scale scenes.

Future research focused on techniques for extending dense SLAM to large scale environments. Whelan *et al*. proposed Kintinuous [57], a moving volume technique where only the current local map is stored in a regular grid SDF representation on the GPU. Voxels corresponding to previously visited scenes are streamed to the CPU to free up memory for newly visited areas. Nießner *et al*. proposed instead a more efficient technique for handling SDF reconstructions based on hierarchical hashing for allocating and accessing voxels [58]. It is the current state-of-the-art technique for methods using SDF representations [59].

Another approach is the system proposed by Keller *et al*. [53] who use surfels to represent the 3D scene as these are less memory intensive than the regular grids corresponding to SDF-based implementations.

1.2.2.3 Dense Visual Odometry

Frame-to-Model Alignment Another notable design decision common in many systems is to replace keyframe-based visual odometry with frame-to-model alignment [56] [54]. This means that incoming frames are aligned against virtual views rendered from the existing reconstruction instead of against previously seen frames. The motivation for this is that storing keyframes as well as a 3D model would be redundant. This works well for depth registration because 3D fusion typically results in smoother and less noisy rendered depth images. However, current methods for real-time fusion of colour typically implement per-pixel independent averaging, which can produce artifacts. This can be seen in Figure 1.1 (right) where the floor and the desk are reconstructed with incorrect non-uniform colour. In more severe cases these artifacts could be problematic for photometric registration and an important direction would be to research more accurate methods for real-time photometric fusion.

Visual Odometry from Intensity Images Techniques which are equivalent to direct visual odometry are also investigated for intensity images provided by RGB-D cameras. These techniques include direct and dense methods, where the current image is repeatedly warped towards a reference until the per-pixel error over all pixels is minimised [60, 61].

Visual Odometry from Range Images As RGB-D cameras provide both colour and depth data, methods for motion estimation from range data are also implemented. Many approaches propose variations of the *iterative closest point* (ICP) algorithm [62]. In order to make ICP run in real-time, corresponding points between two point clouds can be quickly established using projective data association. Given these correspondences, the transformation between two point clouds is computed by minimising a cost function such as the point-to-point error. However, it was found that indoor scenes which contain many planar structures benefit from implementing a point-to-plane error, which also requires the computation of per-point normals [63].

A different approach for aligning depth images was proposed by Jaimez *et al.* [64], a formulation that adapts direct image alignment to depth images: instead of using point clouds, this technique aligns the depth images.

Computing visual odometry from depth images provides increased robustness in the case of changing illumination. However, these techniques can suffer if there is not sufficient structure in the environment to constrain the motion in all degrees of freedom. For this reason, depth registration is typically complemented by techniques which align colour images as well [54] [57].

1.2.2.4 Global Estimation and Loop Closure Implementations

Using dense 3D models instead of keyframe-based implementations poses a challenge with respect to how loop closures can be implemented in order to correct drift accumulated within the 3D model.

ElasticFusion [54] make the interesting decision of implementing loop closures as non-rigid deformations on the 3D model [65]. Their choice of representing the environment using surfels rather than an SDF aids this decision, as it is easier to deform points by transforming each one individually rather than a volumetric model. Nevertheless, recent efforts with BundleFusion [50] do implement loop closure ”deformations” on volumetric SDF representations through repeated surface reintegration operations. They de-integrate RGB-D images from old poses and re-integrate them at the new poses. They demonstrate high quality reconstructions of apartment-sized spaces but do require two GPUs to run in real-time.

1.3 The Problem of Robustness in Visual Odometry

The geometric aspects of visual SLAM are well understood and many solutions are now available that work reliably in ideal conditions. The current focus in state-of-the-art techniques is to enable these systems to operate robustly in real-world conditions [66].

1.3.1 Challenges for Visual Odometry

In this thesis, we focus in particular on the following challenges related to visual odometry:

1. Lack of texture in the images can lead to degenerate motion estimates.
2. Significant noise such as motion blur, illumination change or rolling-shutter effect can induce errors.

3. Many visual SLAM systems assume they operate within a static environment and moving objects can cause errors within visual odometry.

Next, we discuss two common strategies for achieving robustness, namely through sensor fusion or robust computer vision.

1.3.2 Sensor Fusion

Sensor fusion is a branch of research which concerns how to best use information from multiple sensors to obtain the highest accuracy state and reconstruction estimates. It is often inspired by human biology. For example, human perception makes use of senses such as touch and sound to complement vision. Further review regarding fusion of vision with legged odometry and inertial sensors is provided in Chapters 3 and 5.

Typical state-of-the-art approaches often formulate this problem as bundle adjustment optimisation problems, where visual constraints are complemented with pose and measurements constraints from the different sensors.

Successful approaches have been proposed, including visual-inertial systems [67] [68] [69], sonar-vision systems for underwater navigation [4] [70], fusion of wheel or leg odometry with vision for land robots [71] [72].

While there can be significant benefits in terms or robustness and accuracy with implementing sensor fusion systems, there are also complications:

1.3.2.1 Calibration

The sensors must be extrinsically calibrated, such that measurements from one sensor can be transformed into the coordinate system of another. One possibility to achieve this is to physically measure the distance between the sensors. Common approaches rely on recording calibration sequences, where both sensors execute the same motion and estimate their own trajectories [73]. The calibration method then aligns these trajectories to recover the transformation between the sensors.

1.3.2.2 Synchronisation

The sensors must be synchronised so that there is temporal correspondence between measurements of different sensors. In practice, the best synchronisation can be achieved through hardware synchronisation where all sensors involved have messages time stamped with respect to the same clock. This can be found typically in stereo camera set-ups or in visual-inertial sensors [74]. However, systems which fuse sonars or leg/wheel odometry cannot implement this, and instead resort to techniques such as time-of-arrival, where messages are stamped the moment they are received by the computer running their driver.

1.3.3 Robust Computer Vision

In this section we discuss approaches for robust computer vision that are specifically targeted towards visual odometry. Techniques for increased robustness to dynamic environments are reviewed in more detail in Chapters 4 and 5.

1.3.3.1 Dense and Direct Visual Odometry

In dense and direct image alignment applications, a common strategy is to first build a pyramid of images and align these instead of the original images, starting from the lowest resolution towards the highest [33] [60]. This is necessary because dense and direct image alignment algorithms rely on image gradients which are only valid locally.

Another important strategy is the use of an M-estimator to downweight the influence of high residuals that could originate from outlier pixel correspondences [75] [64].

Recent approaches are also investigating using learned features instead of raw pixel values. Czarnowski *et al.* [76] replace pyramid-based dense image alignment with aligning the pyramid of features from a convolutional neural network and demonstrate increased robustness to strong illumination changes.

1.3.3.2 Direct vs Feature-based Formulations

Regarding the robustness of feauture-based vs direct SLAM, Yang *et al.* [77] performed a study which compares the current state of the art monocular SLAM systems, ORB-SLAM [24] and DSO [27]. Both systems implement sliding-window bundle adjustment, however ORB-SLAM implements feature-based alignment while DSO is a direct method. While they display state-of-the-art accuracy, their robustness was found to differ:

- DSO is more robust to motion blur than ORB-SLAM, as this can affect the appearance of features significantly.

- ORB-SLAM is more robust to rolling-shutter effects than DSO. In this case, features are affected to a lesser extent compared with the raw image.

- DSO works more reliably in scenes with little texture, as it can make use of any pixel with gradient. On the other hand, ORB features would not be detected unless there is significant gradient in the neighbourhood of a pixel.

In light of these differences, one possible research direction is to develop a system which combines both techniques, to achieve robustness in all scenarios.

1.3.3.3 Handling Motion Degeneracies

Research also concerns strategies for handling situations where the current image does not contain enough features to constrain the motion estimate. Zhang *et al*. [78] analyse optimisation-based problems by inspecting the structure of the constraints and eliminating the unconstrained degrees of freedom from the estimation procedure, *i.e*. they only produce motion estimates for the observable states. Jaimez *et al*. [79] propose a filtering technique where motion from the unconstrained degrees of freedom is approximated by previous velocity estimates.

1.4 Contributions and Outline

In this thesis, we focus on addressing the challenges discussed in Section 1.3.1 by proposing dense visual SLAM and odometry methods which are robust to (1) lack of features in the image, (2) noise such as illumination change or motion blur, (3) the presence of moving objects in the scene. This is achieved through two main strategies, namely sensor fusion and motion segmentation.

The main contributions of the thesis are as follows:

Chapter 3 addresses these challenges and proposes a visual SLAM system designed to operate on a humanoid robot. Robotic applications of visual SLAM are challenging due to the dynamic motion of the robot, the lack of features in man-made environments and the presence of dynamics in the scene. To achieve robustness in these situations, we propose a sensor fusion system that combines semi-dense visual odometry from a stereo camera mounted on the robot with a motion prior from a kinematic-inertial state estimator. We evaluate this system using the NASA Valkyrie humanoid robot in a laboratory environment equipped with a Vicon motion capture system. Our experiments demonstrate pose tracking robustness to sudden view change, motion blur in the image, change in illumination and tracking through several sequences of featureless areas in the environment. Finally, we provide a qualitative evaluation of our stereo 3D reconstruction against a LIDAR map. This contribution is based on the following publication:

Direct Visual SLAM fusing Proprioception for a Humanoid Robot. Raluca Scona, Simona Nobili, Yvan R. Petillot, Maurice Fallon. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.*

Chapter 4 is the second contribution of this thesis and proposes a technique to increase the robustness of dense visual SLAM in dynamic scenes. The method performs simultaneous visual odometry and motion segmentation, resulting in increased robustness compared to state-of-the-art techniques designed for indoor scenes. This contribution is based on the following publication:

StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments. Raluca Scona, Mariano Jaimez, Yvan R. Petillot, Maurice Fallon, Daniel Cremers. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2018*

Chapter 5 elaborates on the technique introduced in Chapter 4. As this was a vision-only method, it is still susceptible to failure when moving objects occupy a large portion of the image. Here, we extend this method by combining it with information from a gyroscope. We collect a dataset of RGB-D-gyroscope sequences in increasingly dynamic scenes with motion capture ground truth trajectories and evaluate the contributions of motion segmentation and motion prior integration for achieving robustness.

In addition to the technical contribution, the remaining structure of the thesis is as follows:

Chapter 2 describes the tools used in this work, including mathematical preliminaries, the sensors used during experiments and the dense visual SLAM system ElasticFusion [54] which formed the baseline.

Chapter 6 summarises the contributions of this thesis and discusses future research directions.

Appendix A provides links to open-source code, datasets and videos produced as part of our research.

Appendix B cites other research this project has contributed to which does not form a part of this thesis.

Chapter 2

Preliminaries

This chapter provides an overview of the tools used within this thesis including concepts of 3D Geometry and methods for minimising non-linear cost functions. We describe the sensors used during experiments and the dense visual SLAM system which formed the baseline of this research, ElasticFusion [54].

2.1 Mathematical Concepts

This section describes some of the tools required for computing motion estimation, including methods for how to minimise least-squares energies and how to formulate such energies in terms of 3D motion.

Notation We refer to matrices as bold upper-case letters ($\boldsymbol{M}$), vectors as bold lower-case letters ($\boldsymbol{v}$) and scalars as lower-case letters (s).

2.1.1 Non-linear Least-squares Energy Minimisation

Within this thesis motion estimation is formulated as an energy minimisation problem. The proposed energies are non-convex and non-linear which means it is not trivial to find a solution. We solve these problems using least-squares optimisation techniques as they are efficient enough to run in real-time, as opposed to global search methods.

Least-squares problems aim to find a solution $\boldsymbol{x} \in \mathbb{R}^n$ which minimises a sum of m squared non-linear residual functions $r_i : \mathbb{R}^n \to \mathbb{R}$. This can be formulated as:

$$E(\boldsymbol{x}) = \frac{1}{2} \sum_{i=1}^{m} r_i^2(\boldsymbol{x}) \ . \tag{2.1}$$

These are iterative search techniques which, given an initial guess of the solution, perform local gradient operations on the residual functions in order to improve the solution until a local minima is found.

Next, we discuss the least-squares methods used within this thesis.

2.1.1.1 Newton Method

The Newton Method is an iterative approach for approximating the root of a scalar function [80]: for a smooth function $f : \mathbb{R} \to \mathbb{R}$, the goal is to find $x \in \mathbb{R}$ such that $f(x) = 0$.

Given an initial solution guess x_n, the Newton Method uses first order approximations to estimate the next solution $x_{n+1} = x_n + \delta$:

$$f(x_{n+1}) = f(x_n + \delta) \simeq f(x_n) + f'(x_n)\delta \ . \tag{2.2}$$

The goal is to find a root, (*i.e.* $f(x_{n+1}) = 0$), so the iterative update step is:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \ . \tag{2.3}$$

This is equivalent to iterated tangent line approximations and Figure 2.1 illustrates the behaviour of the Newton Method for an example function.

Eq. 2.3 is iterated until $f(x_{n+1}) \simeq 0$ or until consecutive solutions are very similar to each other $(x_{n+1} \simeq x_n)$, indicating that a local minima has been reached and the solution cannot be improved further.

For this procedure to work, it is necessary that the initial guess is close to a true root and that the derivative of the function is not close to zero or very small within the neighbourhood of the root.

If the problem is to find the minimum of a function, and not the root, the Newton Method is applied onto the function derivative instead of the original function. In this case, the update equation is:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} \ . \tag{2.4}$$

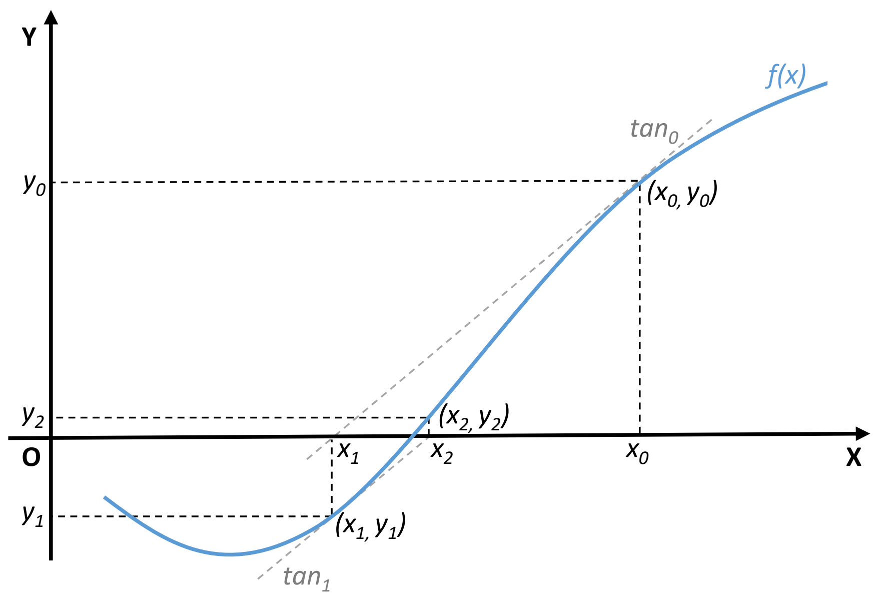


Figure 2.1: A sketch example of the performance of the Newton Method for iteratively finding the root of a given function, namely finding x for which $f(x) = 0$. The approach requires an initial guess (x_0, y_0). It then computes the intersection of the tangent (tan_0) through (x_0, y_0) and the X axis, x_1. (x_1, y_1) is now the new guess, which is already much closer to the solution than the initial guess (x_0, y_0). As a solution is not yet found, the iteration is repeated (*e.g.* (x_2, y_2) is the solution of the next step).

2.1.1.2 Gauss-Newton Method

Gauss-Newton is an application of the Newton method to least-squares problems. The goal is find a solution $\boldsymbol{x} \in \mathbb{R}^n$ to minimise a sum of m squared non-linear residual functions $r_i : \mathbb{R}^n \rightarrow \mathbb{R}$ [81]:

$$E(\boldsymbol{x}) = \frac{1}{2} \sum_{i=0}^{m} r_i^2(\boldsymbol{x}) \, . \tag{2.5}$$

Finding the minimum of this function with the Newton Method implies iteratively applying an update step similar to Equation 2.4, which requires computing the first and second derivatives of E:

$$\nabla E(\boldsymbol{x}) = \sum_{i=0}^{m} r_i(\boldsymbol{x}) \nabla r_i(\boldsymbol{x}) \, ; \tag{2.6}$$

$$\nabla^2 E(\boldsymbol{x}) = \sum_{i=0}^{m} \nabla r_i(\boldsymbol{x}) \nabla r_i(\boldsymbol{x})^T + \sum_{i=0}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}) \simeq \sum_{i=0}^{m} \nabla r(\boldsymbol{x}) \nabla r(\boldsymbol{x})^T \, . \tag{2.7}$$

Gauss-Newton performs an approximation when computing $\nabla^2 E$ by neglecting the second order derivative term of E (Equation 2.7). This is done because in many cases the second order derivative is difficult to compute analytically. The approximation is reasonable when residuals are very small (*i.e.* $r_i(\boldsymbol{x}) \rightarrow 0$), as in this case $\nabla^2 r_i(\boldsymbol{x})$ is cancelled out.

One can rearrange these error terms using the stacked residual vector $\boldsymbol{r}$ and stacked Jacobian $\boldsymbol{J}$, where:

$$\boldsymbol{r}(\boldsymbol{x}) = \begin{bmatrix} r_1(\boldsymbol{x}) \\ \vdots \\ r_m(\boldsymbol{x}) \end{bmatrix} \quad \text{and} \quad \boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial r_1}{x_1} & \dots & \frac{\partial r_1}{x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{x_1} & \dots & \frac{\partial r_m}{x_n} \end{bmatrix} . \tag{2.8}$$

To obtain simpler forms for the gradients in Equations 2.6 and 2.7:

$$\nabla E(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) \; ; \qquad \nabla^2 E(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{J}(\boldsymbol{x}) \, . \tag{2.9}$$

Resulting in the update step:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - (\boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{J}(\boldsymbol{x}))^{-1} \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) \, . \tag{2.10}$$

2.1.1.3 Iteratively Reweighted Least Squares

Optimising over the squared residual functions r_i^2 is problematic in practice due to the presence of outliers in the data which can produce high errors and guide the solver towards the wrong solution. In practice it is common to use an M-estimator to weight the residual functions in order to discriminate between inliers and outliers [60] [82]. In this case, the formulation of the energy function becomes:

$$E = \sum_{i=1}^{m} F(r_i(\boldsymbol{x})) \, . \tag{2.11}$$

There are many options for choosing the penalty function F, with popular choices including Huber [27], which is convex but linear, Cauchy [79], a logarithmic penalty, or Tukey's biweight function [75] which sets the influence of very high residuals to zero.

In this thesis, we make use of the Cauchy robust penalty, a compromise between Huber and Tuckey, which applies a logarithmic penalty onto the influence of residuals:

$$F(r) = \frac{c^2}{2} \ln\left(1 + \left(\frac{r}{c}\right)^2\right) , \tag{2.12}$$

where c defines how strongly residuals are penalised.

However, in this case E does not have a least-squares formulation (Eq. 2.11). Nevertheless, it is possible to obtain a least-squares alternative formulation to E. To minimise E, one computes its derivative with respect to $\boldsymbol{x}$ and sets this to zero:

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{x}} &= \sum_{i=1}^{m} \frac{\partial F(r_i(\boldsymbol{x}))}{\partial r_i(\boldsymbol{x})} \frac{\partial r_i(\boldsymbol{x})}{\partial \boldsymbol{x}} = \\ &= \sum_{i=1}^{m} \frac{\partial \frac{c^2}{2} \ln\left(1 + \left(\frac{r_i(\boldsymbol{x})}{c}\right)^2\right)}{\partial r_i(\boldsymbol{x})} \frac{\partial r_i(\boldsymbol{x})}{\partial \boldsymbol{x}} = \\ &= \sum_{i=1}^{m} \frac{1}{1 + \left(\frac{r_i(\boldsymbol{x})}{c}\right)^2} r_i(\boldsymbol{x}) \frac{\partial r_i(\boldsymbol{x})}{\partial \boldsymbol{x}} = \\ &= \sum_{i=1}^{m} w(r_i(\boldsymbol{x})) r_i(\boldsymbol{x}) \frac{\partial r_i(\boldsymbol{x})}{\partial \boldsymbol{x}} = 0 , \end{aligned} \tag{2.13}$$

for $w(r_i(\boldsymbol{x})) = \frac{1}{1+\left(\frac{r_i(\boldsymbol{x})}{c}\right)^2}$.

If one computes $w(r_i(\boldsymbol{x}))$ for the current estimate $\boldsymbol{x}_c$ and maintains this quantity constant for the current iteration, one can instead solve for the weighted-least squares problem [83]:

$$E'(\boldsymbol{x}) = \frac{1}{2} \sum_{i=1}^{m} w(r_i(\boldsymbol{x}_c)) r_i^2(\boldsymbol{x}). \tag{2.14}$$

This is because the minimizer of E' takes the same form as that of E:

$$\begin{aligned} \frac{\partial E'}{\partial \boldsymbol{x}} &= \frac{1}{2} \sum_{i=1}^{m} w(r_i(\boldsymbol{x}_c)) 2 r_i(\boldsymbol{x}) \frac{\partial (r_i(\boldsymbol{x}))}{\partial \boldsymbol{x}} = \\ &= \sum_{i=1}^{m} w(r_i(\boldsymbol{x}_c)) r_i(\boldsymbol{x}) \frac{\partial (r_i(\boldsymbol{x}))}{\partial \boldsymbol{x}} = 0 . \end{aligned} \tag{2.15}$$

To solve the weighted least-squares energy E', the update iteration then becomes:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - (\boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{W} \boldsymbol{J}(\boldsymbol{x}))^{-1} \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{W} \boldsymbol{r}(\boldsymbol{x}) , \tag{2.16}$$

where $\boldsymbol{W}$ is the diagonal matrix containing the per-residual weights $w(r_i)$.

2.1.2 3D Rigid Body Motion

Given two 3D reference frames, A and B, the pose of frame B with respect to frame A can be described using a transformation $\boldsymbol{T}_{A\to B} \in \mathbb{SE}(3)$ which consists of a translation $\boldsymbol{t}_{A\to B} \in \mathbb{R}^3$ and a rotation $\boldsymbol{R}_{A\to B} \in \mathbb{SO}(3)$. Transformations can be stored in matrix representation, taking the form:

$$\boldsymbol{T} = \left(\begin{array}{ccc|c} & \boldsymbol{R}_{3\times3} & & \boldsymbol{t}_{3\times1} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) .$$

Using transformation matrices, one can transform a point from one reference frame to another or apply multiple consecutive transformations through matrix multiplication. Importantly, transformation matrices do not suffer from singularities, unlike Euler angles.

However, they are overparametrised representations due to the rotation component which requires 9 interdependent parameters to be represented. For this reason, transformation matrices cannot directly be used within optimisation procedures such as the ones described in the previous section, as these are designed for flat Euclidean spaces, *i.e.* $\mathbb{R}^n$. Optimisation requires a minimal representation of 3D poses, meaning there should be 6 parameters fully representing the 6 degrees of freedom.

For this, we make use of Lie group theory [84] which allows to relate elements of Lie groups, $\mathbb{SE}(3)$, with elements of their corresponding Lie algebras, $\mathfrak{se}(3)$, with a minimal representation (*i.e.* $\boldsymbol{\xi} \in \mathfrak{se}(3) \in \mathbb{R}^6$) . The Lie algebra is the tangent space at the identity of a Lie group: it is locally flat and defines how the group behaves around the identity.

Mapping between the Lie group $\mathbb{SE}(3)$ and Lie algebra $\mathfrak{se}(3)$ is done through the exponential and logarithmic operations [84]. Consider $\boldsymbol{\xi} \in \mathfrak{se}(3)$, where $\boldsymbol{\xi} = (\boldsymbol{\nu},\ \boldsymbol{\omega})^T$, with $\boldsymbol{\nu}, \boldsymbol{\omega} \in \mathbb{R}^3$ representing the translational and rotational components of $\boldsymbol{\xi}$ respectively and where the skew-symmetric form of $\boldsymbol{\omega}$ is denoted as $\boldsymbol{\omega}_\times$:

$$\boldsymbol{\omega}_\times = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} .$$

Mapping from a twist $\boldsymbol{\xi}$ to a transformation matrix $\boldsymbol{T}$ is done through the exponential operation:

$$\exp : \mathfrak{se}(3) \rightarrow \mathbb{SE}(3) \; ; \tag{2.17}$$

$$\exp(\hat{\boldsymbol{\xi}}) = \boldsymbol{T}_{4\times 4} \; , \tag{2.18}$$

where $\hat{\boldsymbol{\xi}}$ contains the skew-symmetric form of the rotational component of $\boldsymbol{\xi}$:

$$\hat{\boldsymbol{\xi}} = \left(\begin{array}{c|c} \boldsymbol{\omega}_\times & \boldsymbol{\nu} \\ \hline 0 & 0 \end{array}\right) = \begin{pmatrix} 0 & -\omega_z & \omega_y & \nu_x \\ \omega_z & 0 & -\omega_x & \nu_y \\ -\omega_y & \omega_x & 0 & \nu_z \\ 0 & 0 & 0 & 0 \end{pmatrix} .$$

Inversely, mapping from a transformation $\boldsymbol{T}$ to a twist $\boldsymbol{\xi}$ is computed using the logarithmic operation:

$$\hat{\boldsymbol{\xi}} = \log(\boldsymbol{T}) \; . \tag{2.19}$$

For simplicity of notation, this thesis assumes that both $\exp$ and $\log$ operations also perform the conversion to/from skew-symmetric notation.

When estimating the pose of a moving sensor in time, transformation matrices are used to represent global poses and twist representations are used to compute incremental pose updates through optimisation.

For example, for a sensor with a global pose $\boldsymbol{T}$, the computation of its infinitesimal motion update $\boldsymbol{\xi}$ is defined as a local optimisation problem, where an initialisation of the twist at the identity is valid (*i.e.* $\boldsymbol{\xi}_0 = \mathbf{0}_{6\times 1}$). The optimisation problem is then solved using the non-linear least squares techniques discussed previously:

$$E(\boldsymbol{\xi}) = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\boldsymbol{\xi}) \; . \tag{2.20}$$

After every iteration, the global pose $\boldsymbol{T}$ is then updated as:

$$\boldsymbol{T}_{n+1} = \boldsymbol{T}_n \exp(\boldsymbol{\xi}) \; . \tag{2.21}$$

This allows maintaining the global pose in a singularity free transformation while updates are performed using a minimal parameter representation which is free from singularities in the local space.

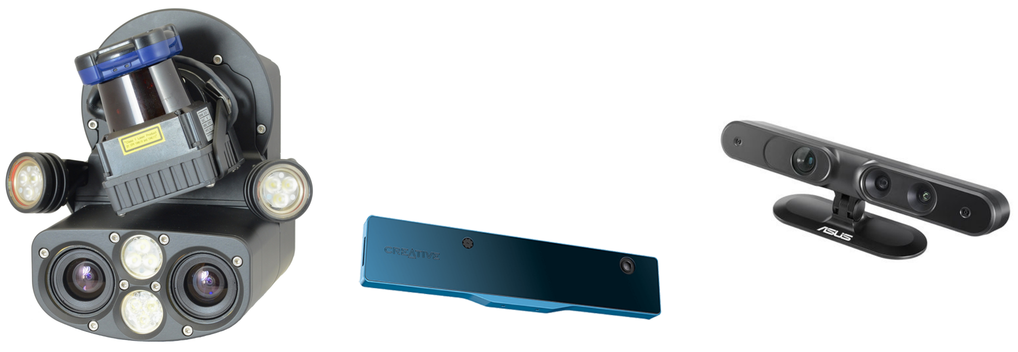

(a) Carnegie Robotics MultiSense SL (b) Intel RealSense ZR300 (c) Asus Xtion Pro Live

Figure 2.2: The 3D visual sensors used within this thesis: passive stereo (2.2a), active stereo 2.2b and structured light 2.2c.

2.2 Computer Vision

In this section we provide an overview of the visual sensors and the baseline dense visual SLAM system used within this thesis, ElasticFusion [54].

2.2.1 3D Visual Sensors

We use cameras which sense intensity as well as distance for each pixel in an image. There are many methods for producing depth cameras but in this section we provide an overview of the ones used within this thesis.

2.2.1.1 Depth from Stereo

In this category, we use the stereo pair within the Carnegie Robotics MultiSense SL and the Intel RealSense ZR300 camera.

In a stereo set-up two cameras are used to compute per-pixel depth. Given two images captured simultaneously by both cameras, one identifies, for each pixel in the first image, its corresponding pixel in the second image. The difference between the coordinates of these pixels is called *disparity*. The larger the disparity, the closer the corresponding 3D point is to the camera. Given the disparity and the known spatial calibration between the two cameras, the 3D coordinates of each pixel can be computed by triangulation.

The Carnegie Robotics MultiSense SL[1] (Figure 2.2a) functions in this manner. It contains a stereo pair consisting of one colour camera and one greyscale camera. It

[1]http://docs.carnegierobotics.com/SL/

estimates disparity using the Semi-global Matching algorithm [85] implemented on the FPGA of the sensor. It is an entirely *passive* camera, meaning that it relies only on the existing light in order to capture its data.

Computing depth from stereo does not work well in the case of repetitive texture or when there is a lack of texture in the scene. In both cases, this is because unique per-pixel correspondences cannot be established correctly, rendering the depth estimation problem impossible to solve.

The Intel RealSense ZR300 [2] (Figure 2.2b) handles textureless scenes by generating *artificial* texture. This sensor contains a pair of infrared stereo cameras, as well as an infrared projector. The projector emits a pattern onto the scene, generating texture and improving the robustness of the stereo matching algorithm. However, this is not a universal solution. Outdoors, strong sun light may mask the pattern, making it invisible to the camera. In this case, only the standard passive stereo implementation of the camera might work.

2.2.1.2 Structured-light Cameras

Within this category, we use the Asus Xtion Pro Live [3] (Figure 2.2c).

This sensor consists of a colour camera, an infrared camera and an infrared projector. The projector emits a known infrared pattern onto the world, which is perceived by the camera. The 3D geometry of the world induces a deformation onto the pattern, which is used to compute per-pixel depth.

This approach for computing depth is more accurate than general stereo depth estimation as implemented by the Carnegie Robotics MultiSense SL or the Intel RealSense ZR300. However, and as it is also the case for the Intel RealSense ZR300, the pattern is likely invisible in strong sunlight, meaning that the camera can only be used indoors.

2.2.1.3 Operational Range of 3D Visual Sensors

It must be noted that for all sensors discussed within this section the operational range is limited, typically between $0.5m$ and $5m$ from the camera. In the case of stereo setups, the operational range depends on the distance between the two cameras (in our

[2]https://software.intel.com/en-us/realsense/zr300
[3]https://www.asus.com/us/3D-Sensor/Xtion_PRO_LIVE/

case, this is smaller than $\simeq 10cm$ for both sensors). In the case of the structured light camera, this is because the infrared pattern is not clearly visible after a certain distance.

2.2.2 Pinhole Camera Model

Given a 3D point $\boldsymbol{v} \in \mathbb{R}^3$ in the camera coordinate frame C, we compute its 2D pixel coordinate $\boldsymbol{x} \in \Omega$ where $\Omega \in \mathbb{N}^2$ is the image plane using the pinhole projection function π [86]:

$$\boldsymbol{x} = \pi(\boldsymbol{v}) = \begin{bmatrix} f_x \frac{X}{Z} + o_x \\ f_y \frac{Y}{Z} + o_y \end{bmatrix}, \boldsymbol{v} = [X, Y, Z, 1]^T, \boldsymbol{x} = [u, v]^T \ , \tag{2.22}$$

where f_x, f_y are the horizontal and vertical focal lengths and o_x, o_y are offsets to account for the fact that the optical centre does not coincide with the origin of the image coordinates. Note that $\boldsymbol{v}$ is expressed in homogeneous coordinates as these are easier to manipulate using transformation matrices.

The inverse of this function, known as the backprojection operation, is defined as π^{-1}:

$$\boldsymbol{v} = \pi^{-1}(\boldsymbol{x}, Z) = \begin{bmatrix} \frac{u-o_x}{fx} Z \\ \frac{v-o_y}{fy} Z \\ Z \\ 1 \end{bmatrix}, \tag{2.23}$$

which computes the 3D coordinates of pixel $\boldsymbol{x}$ given its corresponding known depth value Z.

2.2.3 Baseline Dense Visual SLAM System - ElasticFusion

The contributions listed in Section 1.4 are formulated and implemented as extensions of an existing visual SLAM baseline system, ElasticFusion [54]. ElasticFusion is a real-time dense RGB-D SLAM system with state-of-the art performance regarding camera pose tracking accuracy and 3D reconstruction quality.

Several design decisions within ElasticFusion motivate us to use this system as a baseline:

1. It estimates dense 3D models of the scene which can be used for robotic applications by serving as collision environments.

2. At the time of publishing, it was one of few systems which implemented loop

closure corrections on the dense 3D model.

3. It supports locally loopy trajectories and revisiting previously seen areas, which are common robotic motions.

ElasticFusion works in a decoupled fashion, alternating between camera pose tracking and dense 3D reconstruction.

Next, we discuss the main components of the system:

1. **Dense 3D reconstruction** is implemented by representing the environment using surfels, in a similar technique as proposed by Keller *et al.* [53] (see Section 1.2.2.1). New surfels are generated as the camera explores new areas and are added to the 3D model as follows: if a new surfel has a model correspondence, these are fused together. Otherwise, the new surfel is simply included in the 3D model. Surfels are also allocated a timestamp, which indicates the last time they have been observed. Based on this, older portions of the model are termed *inactive* while new ones are *active*. This distinction is important for loop closure implementations.

2. **Camera pose estimation** works by aligning incoming RGB-D image pairs against virtual RGB-D image pairs rendered from the *active* portion of the 3D model from the current point of view of the camera. The alignment implements photometric and geometric cost functions.

3. **Local loop closures** are implemented to correct small drift within the trajectory estimates which can occur in situations where the camera repeatedly scans the same areas of a small scene. A local loop closure is implemented by registering virtual views from the *active* and the *inactive* portions of the 3D model from the current camera pose estimate. If these views are considered to be in alignment, the model is non-rigidly deformed into place.

4. **Global loop closures** are implemented to handle more significant drift. This implements a place recognition technique based on the work proposed by Glocker *et al.* [87]. It consists of building a random fern database of rendered views of previously visited scenes. At every time step, the database is queried for a match for the current view. If a match is found, the whole model is non-rigidly deformed to guarantee a consistent reconstruction.

This thesis is mainly concerned with robust camera pose estimation, so we provide additional details regarding this component here.

2.2.3.1 Camera Pose Estimation

Camera pose tracking works in a frame-to-model formulation: the input is an RGB-D image pair (C_D, Z_D) which is aligned against an artificial RGB-D image pair (C_M, Z_M) rendered from the 3D model from the point of view of the previous camera pose.

Visual tracking in ElasticFusion is formulated as an optimisation problem over both photometric and geometric data. We discuss these components next.

Geometric Alignment The geometric component of camera pose estimation implements Iterative Closest Point (ICP).

Given the current depth image Z_D, a 3D point $\boldsymbol{v}_D^p$ is computed for every pixel p given its coordinate on the image plane $\boldsymbol{x}^p \in \Omega$, its depth value $Z_D(\boldsymbol{x}^p)$ and assuming the pinhole reprojection function π^{-1} (Eq. 2.23). Similarly, vertices $\boldsymbol{v}_M^q$ are also computed from the predicted depth map Z_M, in addition to per-point normals $\boldsymbol{n}_M^q$.

Correspondences between the two point clouds are computed using projective data association. To remove outliers, correspondence pairs are then filtered using thresholds on the Euclidean distance between the vertices ($\boldsymbol{v}_D^p$, $\boldsymbol{v}_M^q$) and on a maximum angular distance between the input vertex $\boldsymbol{v}_D^p$ and the normal $\boldsymbol{n}_M^q$.

After this preprocessing step, one residual is formulated for each pair of correspondences, minimising the point-to-plane error between input and predicted vertices:

$$r_{icp}^p(\boldsymbol{\xi}) = \left(\boldsymbol{v}_M^q - \exp(\boldsymbol{\xi})\boldsymbol{T}\boldsymbol{v}_D^p\right) \cdot \boldsymbol{n}_M^q \, , \tag{2.24}$$

where $\boldsymbol{T}$ is the current total estimated transformation between the input image Z_D and the model Z_M. The camera pose tracking problem is then implemented within a least-squares formulation over the list of valid correspondence pairs M:

$$E_{icp}(\boldsymbol{\xi}) = \frac{1}{2} \sum_{p \in M} (r_{icp}^p(\boldsymbol{\xi}))^2 \, . \tag{2.25}$$

Photometric Alignment For photometric alignment, both current and predicted colour images (C_D, C_M) are first converted to intensity images (I_D, I_M). They are then registered in a direct and dense formulation.

For every pixel $p \in \Omega$, a residual is formulated to compute the difference in intensities against its corresponding pixel in I_M, established by projection:

$$r^p_{rgb}(\boldsymbol{\xi}) = I_M\left(\pi(\exp(\boldsymbol{\xi})\boldsymbol{T}\,\pi^{-1}(\boldsymbol{x}^p, Z_D(\boldsymbol{x}^p)))\right) - I_D(\boldsymbol{x}^p)\,, \tag{2.26}$$

where $\boldsymbol{T}$ is the current total estimated transformation between the input image I_D and the model I_M.

This procedure also includes a filtering step for pixel correspondences, removing those pairs for which corresponding vertices are further than a set Euclidean distance and also discounting pixel pairs which are located in textureless regions.

Finally, the photometric energy term is formulated as a least-squares problem over the set of valid pixel correspondences N:

$$E_{rgb}(\boldsymbol{\xi}) = \frac{1}{2}\sum_{p\in N}(r^p_{rgb}(\boldsymbol{\xi}))^2\,. \tag{2.27}$$

Photometric alignment is sensitive to illumination changes and increased robustness to this situation can be achieved by combining it with geometric alignment.

Complete Formulation The combined pose tracking energy function is defined as follows:

$$E(\boldsymbol{\xi}) = wE_{rgb}(\boldsymbol{\xi}) + E_{icp}(\boldsymbol{\xi})\,. \tag{2.28}$$

The weight w is empirically set to 0.1 reflecting the difference in units between the two error terms: metres as used in E_{icp} and pixel intensity values as used in E_{rgb}.

Design of Solver This energy function is solved using the Gauss-Newton iterative algorithm for solving non-linear least squares problems (described in Section 2.1.1.2). Next we discuss how it is applied to the problem of image alignment. Algorithm 1 lists the procedure for implementing the solver.

Linearisation The solver is initialised with the initial guess $\boldsymbol{\xi} = \mathbf{0}_{6\times1}$, which is reasonable for high frame-rate cameras. At each iteration, an incremental estimate $\Delta\boldsymbol{\xi}$ is computed by solving the linear system of equations:

$$(w\boldsymbol{J}^\top_{rgb}\boldsymbol{J}_{rgb} + \boldsymbol{J}^\top_{icp}\boldsymbol{J}_{icp})\Delta\boldsymbol{\xi} = -(w\boldsymbol{J}^\top_{rgb}\boldsymbol{r}_{rgb} + \boldsymbol{J}^\top_{icp}\boldsymbol{r}_{icp})\,. \tag{2.29}$$

Algorithm 1: Frame-to-model alignment in ElasticFusion

```
Input: $I_D, Z_D, I_M, Z_M$
compute image pyramids for $I_D, Z_D, I_M, Z_M$;
initialise $\boldsymbol{\xi} = \mathbf{0}_{6\times 1}$;
foreach level of the pyramid do
    foreach iteration do
        compute ICP correspondences given current $\boldsymbol{\xi}$;
        compute RGB correspondences given current $\boldsymbol{\xi}$;
        compute $\boldsymbol{r}_{icp}$ and $\boldsymbol{r}_{rgb}$;
        solve for $\Delta\boldsymbol{\xi}$;
            $(w\boldsymbol{J}_{rgb}^\top \boldsymbol{J}_{rgb} + \boldsymbol{J}_{icp}^\top \boldsymbol{J}_{icp})\Delta\boldsymbol{\xi} = -(w\boldsymbol{J}_{rgb}^\top \boldsymbol{r}_{rgb} + \boldsymbol{J}_{icp}^\top \boldsymbol{r}_{icp})$;
        update $\boldsymbol{\xi}$;
            $\boldsymbol{\xi} = \log(\exp(\boldsymbol{\xi})\exp(\Delta\boldsymbol{\xi}))$
```

Using the stacked Jacobian and residual notation introduced in Section 2.1.1.2. The Jacobians $\boldsymbol{J}_{icp}$ and $\boldsymbol{J}_{rgb}$ are computed as:

$$\boldsymbol{J}_{rgb} = \left.\frac{\partial \boldsymbol{r}_{rgb}}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}}, \qquad \boldsymbol{J}_{icp} = \left.\frac{\partial \boldsymbol{r}_{icp}}{\partial \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\mathbf{0}}. \tag{2.30}$$

After solving this linear system of equations, $\Delta\boldsymbol{\xi}$ is used to update the estimate $\boldsymbol{\xi}$ between two consecutive iterations k and $k+1$ as follows:

$$\boldsymbol{\xi}_{k+1} = \log(\exp(\boldsymbol{\xi}_k)\exp(\Delta\boldsymbol{\xi}))\ . \tag{2.31}$$

It is possible to still evaluate the Jacobians $\boldsymbol{J}_{icp}$ and $\boldsymbol{J}_{rgb}$ at $\boldsymbol{\xi} = \mathbf{0}$ even after $\boldsymbol{\xi}$ is updated. This is because the residual formulations of r_{icp} and r_{rgb} decouple the total motion estimated up to this point, $\boldsymbol{T}$, from the current incremental estimate, $\boldsymbol{\xi}$.

Image Pyramid Registration Gauss-Newton makes use of second-order approximations to solve E, which are only valid locally and provide a small basin of convergence for the solver. In order to ensure robustness to large motions, the solver is run in a coarse-to-fine scheme. The image pairs (I_D, Z_D) and (I_M, Z_M) are repeatedly blurred and downsampled to create an image pyramid and the energy minimisation procedure is run starting from the coarsest levels towards the finest. After the solver has converged, the global pose of the camera is updated as:

$$\boldsymbol{T}_{t+1} = \boldsymbol{T}_t \exp(\boldsymbol{\xi})\ . \tag{2.32}$$

2.3 Evaluation Metrics

In this thesis, we are estimating the pose of a moving sensor. In order to evaluate the performance of proposed approaches, we make use of several quantitative metrics.

These metrics assume that, along with the estimated trajectory of the moving sensor, there is also ground truth knowledge of the sensor's trajectory. This is typically obtained from a separate motion capture system.

The trajectories estimated by our algorithm and the ground truth system must be synchronised and calibrated such that for each 3D pose estimated by our system, there exists a corresponding 3D pose estimated by the ground truth system, measured at the same time and within the same coordinate frame. In practice, one can achieve millisecond accurate synchronisation and millimetre accurate spatial calibration. The techniques used to perform synchronisation and calibration are described in Chapters 3 and 5.

Assuming this set-up, evaluation can be carried out as follows. We refer to ${}^{(est)}\boldsymbol{T}_{A_{t_i}\to B_{t_j}}$ as the transformation measured by the estimator est of frame B at time t_j relative to frame A at time t_i. We make use of the reference frames C, which represents the camera frame, and W, which represents the world frame (the origin).

We evaluate the performance of our proposed estimator est against the ground truth gt.

2.3.1 Absolute Trajectory Error

To compute the absolute trajectory, the ground truth and estimated trajectories must be aligned in the least-square sense. For this, we first establish pairs of corresponding poses between the trajectories based on the timestamps and then perform Procrustes analysis to recover the 3D transformation $\boldsymbol{Q}$ that maps the estimated trajectory onto the ground truth trajectory.

At time t, the error between corresponding poses is [26]:

$$\text{ATE}_t = {}^{(gt)}\boldsymbol{T}^{-1}_{W\to C_t}\boldsymbol{Q}\,{}^{(est)}\boldsymbol{T}_{W\to C_t}\ . \tag{2.33}$$

2.3.2 Relative Pose Error

To measure drift between two corresponding trajectories, we compute the relative pose error over a time interval Δ at each timestep t [26]:

$$\text{RPE}_t = {}^{(gt)}\boldsymbol{T}_{C_t \to C_t+\Delta}^{-1} \; {}^{(est)}\boldsymbol{T}_{C_t \to C_t+\Delta} \,. \tag{2.34}$$

2.3.3 Drift per Distance Travelled

We divide the relative pose error by the length of the path travelled to obtain the drift per metre travelled:

$$\text{DDT}_t = \frac{\text{RPE}_t}{\sum\limits_{k=t}^{t+\Delta} ||{}^{(gt)}\boldsymbol{t}_{C_k \to C_{k+1}}||} \,. \tag{2.35}$$

We compute these metrics over all timestamps and calculate the root mean square error (RMSE) over all poses for an entire trajectory.

2.4 Robotics

In this section we describe the fundamental robotics concepts used within this thesis.

2.4.1 Proprioceptive Sensors

We discuss some of the sensors used within this thesis and give a high level overview of their functionality.

- **Inertial Measurement Units** (IMUs) typically consist of 3-axis gyroscopes and 3-axis accelerometers.
  - **Gyroscopes** measure 3D angular velocity. By taking into account the amount of time between consecutive measurements, this sensor can be used to estimate the cumulative 3D rotation of a platform.
  - **Accelerometers** measure 3D linear acceleration. On a static platform, this sensor reports acceleration induced by gravity from which estimates of the roll and pitch rotations of the platform can be computed. As rotations about the gravity vector are not sensed, yaw cannot be computed.

In combination with vision, accelerometers can also be used to estimated linear velocity or positional quantities. Vision is required in this case to provide a prior velocity estimate which can subsequently be updated through acceleration integration. Such techniques also require an accurate estimate of the orientation of the sensor in order to decouple gravity from the motion of the platform.

For accurate results it is common in practice to also estimate a time-varying bias for both sensors.

- **Joint Encoders** are used to measure the position of an axle. They are typically placed in a robot's joints to measure the angle of each joint.
- **Force-Torque Sensors** report the forces and torques which are applied onto it on contact. On legged robots, force-torque sensors are typically placed in the feet and used to detect when they are in contact with the ground.

2.4.2 Forward Kinematics

Forward kinematics is a technique which enables estimating the location of each joint on a robot with respect to any other joint. It requires a known 3D model of the robot describing the dimensions of each link and it assumes that it is possible to sense the position of every joint (*e.g.* by having each joint equipped with an encoder).

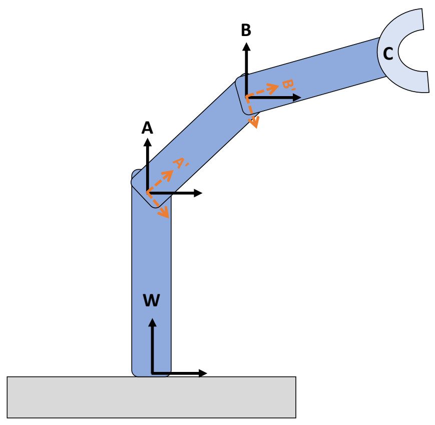


Figure 2.3: Assuming a known 3D model of the robot arm and known joint angles computed from encoders, the pose of the end-effector C can be computed with respect to the base W using forward kinematics.

Consider Figure 2.3, showing a robot arm. The known 3D model of the robot provides information about the relative transformations of the links, $\boldsymbol{T}_{W\to A}, \boldsymbol{T}_{A\to B}, \boldsymbol{T}_{B\to C}$, while the joint encoders measure the current angle of each joint, namely $\boldsymbol{T}_{A\to A'}$, $\boldsymbol{T}_{B\to B'}$. Thus, one can compute the pose of the end-effector, C, with respect to the base, W, as:

$$\boldsymbol{T}_{W\to C} = \boldsymbol{T}_{W\to A}\boldsymbol{T}_{A\to A'}\boldsymbol{T}_{A'\to B}\boldsymbol{T}_{B\to B'}\boldsymbol{T}_{B'\to C} \ . \tag{2.36}$$

Part II

Contributions

Chapter 3

Camera Pose Estimation with Motion Prior Integration Applied to Humanoid SLAM

In this chapter we present an approach for visual SLAM applied to the humanoid robot Valkyrie. Visual SLAM on robotic platforms can be subject to the following four challenges:

- The robot may point its camera towards areas in the environment which lack texture;
- The robot may perform fast motions or be shaking (due to instability) which can introduce significant motion blur into the images;
- Changes in illumination may occur;
- Moving objects may be present in the environment.

These situations pose significant difficulties to current state of the art visual SLAM systems. Indeed, in some cases it is not possible to operate successfully by relying only on visual sensors such as cameras.

Fortunately, robotic platforms are typically also equipped with other sensors, such as IMUs or joint encoders, which provide additional information typically used as input to the robot's locomotion system. We leverage this proprioceptive sensing to aid our visual SLAM system to achieve robust operation under the stated real-world challenges.

We propose an approach for camera pose estimation which fuses visual odometry from a stereo camera with a motion prior from a kinematic-inertial state estimator. The fusion is formulated as an optimisation problem and a non-linear least squares solver is used to compute a solution.

The proposed approach is using data from the Valkyrie robot. Performance was evaluated using ground truth of the robot's motion from a Vicon motion capture system. We find that fusion of visual odometry and a kinematic-inertial motion prior significantly improves the robustness of the visual SLAM system to the four challenges mentioned above. In addition, the visual SLAM system is used to correct drift within the kinematic-inertial state estimator, while the resulting 3D reconstruction can be used to execute collision-free motions.

3.1 Humanoid State Estimation and Visual SLAM

State estimation and visual SLAM are related techniques but have different purposes on humanoid robots.

Kinematic-Inertial State Estimation for Legged Robots State estimation is a core requirement for reactive or closed loop controlled robots where the robot's action is a function of its current state. Estimators used within closed loop applications must run at high frequency and low latency, in order to allow the controller to quickly correct the robot's motions. In addition, it is important that state estimates are locally very accurate and show a low level of noise, in order to prevent the controller from overcompensating and thus causing the robot to move erratically.

On humanoid robots, state estimation is typically implemented using a combination of inertial sensing (gyroscopes and accelerometers), kinematic sensing in the legs and force-torque sensing in the feet. These are low-latency sensors which enable estimation of the robot's position, orientation and velocity at high frequency (>200 Hz).

One group of approaches, including [88] and [89], use the inverted pendulum model to estimate the centre of mass (CoM) as this is the quantity of interest for control purposes. These approaches explicitly measure the deviation of the CoM value from the expected value, allowing for the detection of anomalies such as unexpected contact.

Other approaches estimate the motion of a specific link (typically the root link of the kinematic chain) by incorporating the individual sources of information within a filtering framework ([90, 91, 92]). These approaches were successfully demonstrated on the Boston Dynamics Atlas humanoid robot during the DARPA Robotics Challenge.

On quadrupeds, techniques have also been proposed for robots which are not equipped with contact sensors on the feet. Bloesch *et al.* [93] handle this by designing an Extended Kalman Filter where the state contains not only the body pose of the robot, but also the absolute positions of the foot contact points. Further work extends this and proposes a method which is more robust to slippage [94]. Camurri *et al.* [95] also propose a leg odometry module without requiring contact sensors on the feet, by proposing a probabilistic method to estimate reliable contact.

Visual SLAM for Humanoid Robots To a certain extent, a humanoid robot can operate only using a kinematic-inertial state estimator. A SLAM system is required when the robot must be aware of its location with respect to the environment. However, in most applications, pose estimates provided by SLAM systems are computed at lower frequency and with a higher level of local noise compared with state estimation systems. Nevertheless, they provide pose estimates which are globally accurate and consistent with the sensed environment.

There is a significant history of research in visual localisation and SLAM on humanoids. Initially, this focused on feature-based methods and Extended Kalman Filters (EKF). Stasse *et al.* [96] adapted MonoSLAM [21], a monocular EKF-based SLAM algorithm, to exploit knowledge about the HRP-2 robot's motion from its pattern generator and inertial sensing to improve the robustness of pose tracking. The work was a notable early example demonstrating loop closure on a humanoid.

The fusion of a visual SLAM method with proprioception was also used in the work of Ahn *et al.* [97] who implement loose coupling of forward kinematics, inertial estimates and visual odometry to improve the robustness of visual SLAM on their humanoid robot.

Oriolo *et al.* [98, 99] instead implemented a complementary strategy to fuse pose corrections from their sparse visual SLAM system within their EKF-based kinematic-inertial state estimator. Demonstrations were carried out on the Nao robot. Kwak *et al.* [100] proposed a particle filter-based SLAM method using a stereo camera. They attempted to build a 3D grid map for localisation but noise in the stereo data required them to only record camera data from stationary positions. They also mentioned that corruptions were introduced into their reconstructions by areas of the environment with no texture.

A common characteristic of these works is that they used sparse representations. These are useful for localisation but cannot be interpreted visually or be used for path planning. Recent approaches have also explored applying dense visual SLAM techniques to locomoting robots.

Wagner *et al*. [101] fused robot wheel odometry (*i.e.* not a bipedal robot) with a dense SLAM solution based on a pose-graph extension of KinectFusion. Their work combines the two modalities but as the robot's motion is planar and smooth it avoids the complexities of true humanoid SLAM.

Fallon *et al*. [102] integrated a dense SLAM approach on the Atlas humanoid robot. It provided a dense reconstruction as input to the robot's footstep planning system. However, it did not support loop closure for locally loopy trajectories, which has motivated this work.

We investigate the application of a direct semi-dense SLAM method and aim to achieve sufficient robustness during the walking and turning motions of the robot. We chose ElasticFusion for the current work as it is designed to handle locally loopy trajectories which are common in typical humanoid manipulation scenairos. Frame-by-frame fusion of 3D data results in an up-to-date environment model which can be used for collision-free motion planning.

3.2 The Valkyrie Humanoid Robot

In this section we discuss the functionality of the Valkyrie robot and configuration of the relevant sensors used. We explain our procedures for synchronising the sensor data and calibrating the robot's joints. Finally, we explain the pre-processing filters implemented for noise removal from the camera images.

3.2.1 Hardware Overview

The NASA Valkyrie is a 1.8 m tall, electrically actuated humanoid robot. It weighs 125kg, with a 32 degrees of freedom body and 6 degrees of freedom hands.

The robot contains a Carnegie Robotics Multisense SL global-shutter stereo camera installed in its head (Figure 3.1). The sensor provides 1024×1024 image pairs of colour and corresponding disparity at a rate of 15 Hz. The lenses have a field of view of 80°×80°. Disparity is computed by an implementation of Semi Global Matching [85] running on an FPGA on board the device.

Regarding proprioceptive sensing, it is equipped with a Microstrain GX4-25 MEMS IMU mounted in its pelvis, force-torque sensors in its feet and position encoders on each joint.

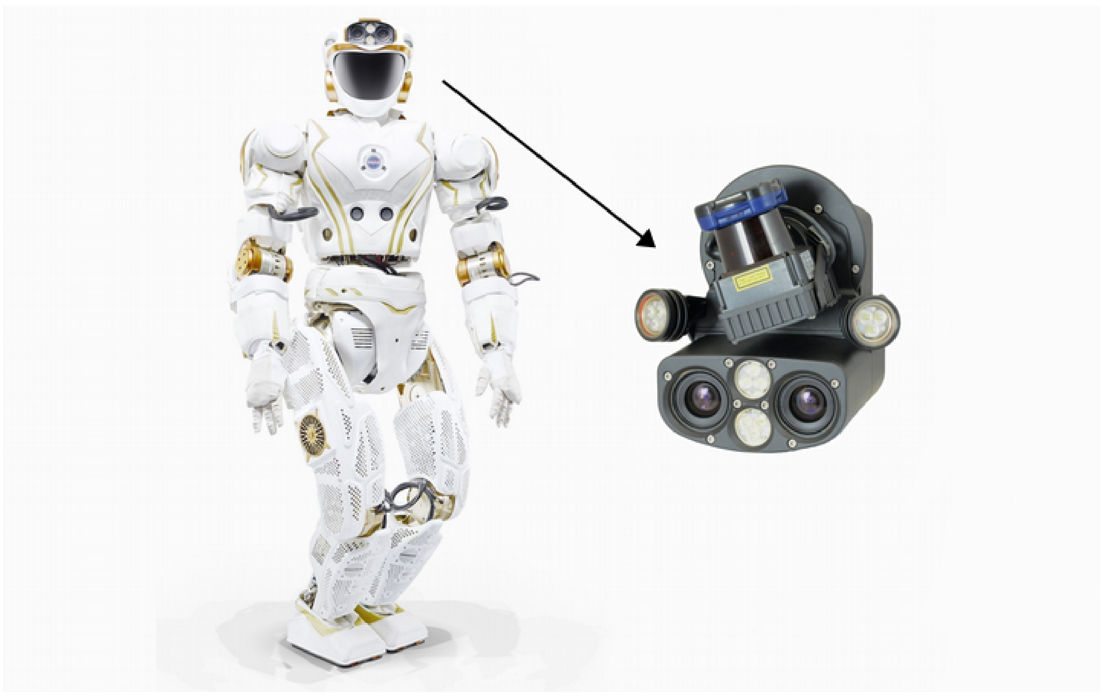


Figure 3.1: The NASA Valkyrie is a 1.8 m tall electrically actuated humanoid robot. Within its head is a Carnegie Robotics Multisense SL which combines a rotating LIDAR sensor and a stereo camera. The sensor is inverted on the robot. (photo credits: NASA and CRL)

3.2.2 Synchronisation and Calibration

The robot is operated through two on-board computers and one separate workstation equipped with a GPU, which is used to run our SLAM algorithm.

All computers are time synchronised through the network using chrony[1]. The MultiSense SL provides the camera image time stamps while the other sensor data is time stamped using the data arrival time on the computer which runs the sensor's driver. This type of synchronisation is accurate for the near real-time proprioceptive

[1]https://chrony.tuxfamily.org/

sensors (IMU, joint encoders and force-torque) which have a latency of less than 1ms.

We performed periodic calibration of the joint encoders (*i.e.* every few months) using manufactured jigs (*e.g.* Figure 3.2). The purpose of these jigs is to aid us in re-setting the encoders to an established origin value.

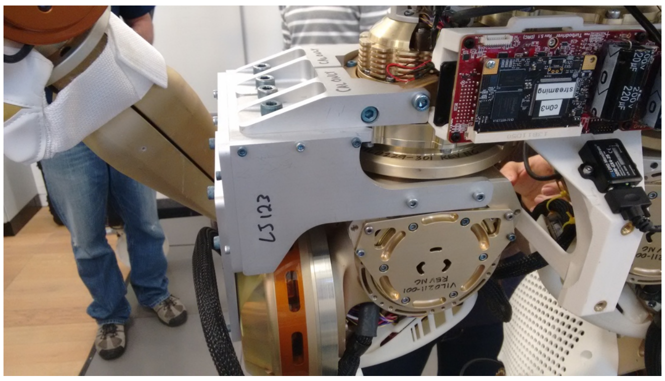


Figure 3.2: Example of an aluminium jig used to calibrate the hip joint position.

3.2.3 Stereo Pre-processing

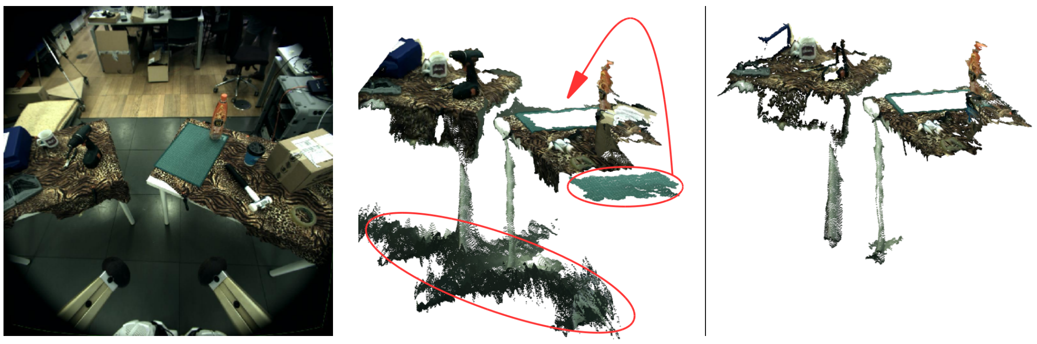
Our stereo camera produces dense disparity images at a rate of 15Hz. However, some of the disparity estimates are not accurate, particularly ones corresponding to areas in the image with low texture. These areas are problematic for the block matching algorithm because a particular low-texture patch within the left camera image will have many possible corresponding patches within the right image with similar appearance. It is thus very difficult to estimate correct correspondences within texture-less scenes, leading in turn to incorrect disparity estimates for such regions.

We implement pre-filtering procedures where we remove data from textureless regions by computing for each pixel the gradients in the vertical, horizontal and diagonal directions over a 5×5 pixel window. If these gradients are small, the pixel is considered to be in an area of low texture and is dropped.

The stereo cameras are set-up in a horizontal configuration, which makes estimating the disparity of horizontal edges unreliable. We discard data originating from edges oriented at an angle of less than 10 degrees from horizontal.

Finally, we remove small unconnected groups of points which could occur due to specular effects.

Figure 3.3 gives a qualitative impression of the effect of this filtering procedure.



(a) Colour Image (b) Raw Stereo Point Cloud (c) Filtered Stereo Point Cloud

Figure 3.3: The raw disparity images are filtered to remove unreliable data. (a) the original colour image. (b) the corresponding raw stereo point cloud - red circles highlight erroneous depth from areas of low texture, such as the floor and a green sheet reconstructed appart from its actual location (indicated with an arrow). (c) the result of our filtering procedure.

3.3 Methodology

In this section we describe our approach for visual SLAM applied to the Valkyrie humanoid robot.

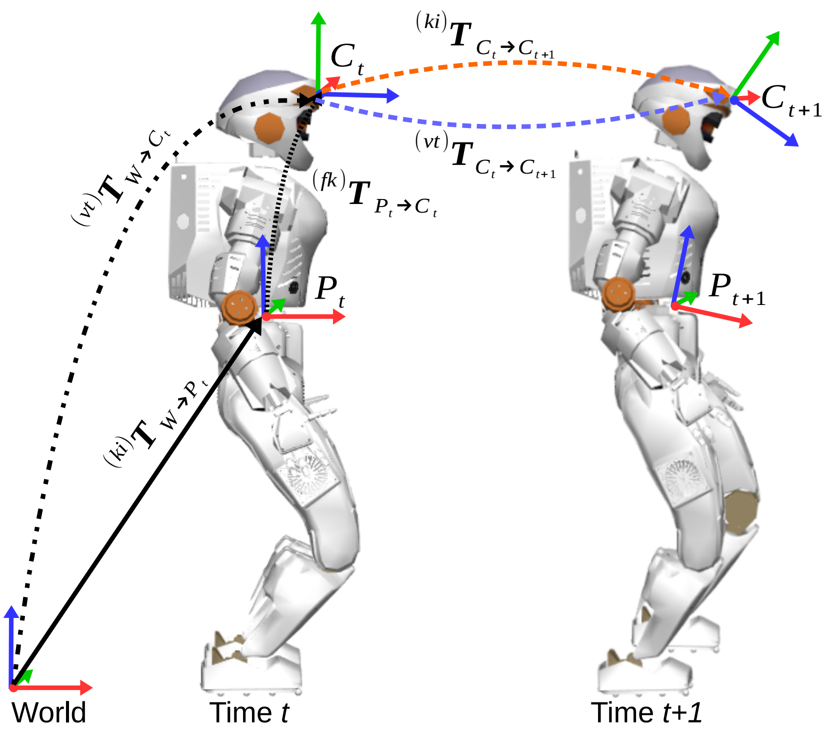
First, we discuss the kinematic-inertial state estimator running on the robot which is used to compute a motion prior of the camera pose estimate. Following this, we describe the visual SLAM system used as a baseline in our integration project. Finally, we propose our approach for robust camera pose estimation fusing vision and kinematic-inertial data, formulated as an optimisation problem.

3.3.1 Notation

A pose is denoted as a transformation matrix $\boldsymbol{T}$, within the class of rigid body motions forming the special Euclidean group $\mathbb{SE}(3)$, composed of a rotation matrix $\boldsymbol{R} \in \mathbb{SO}(3)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$. We refer to ${}^{(est)}\boldsymbol{T}_{A_{t_i} \to B_{t_j}}$ as the transformation measured by the estimator est of frame B at time t_j relative to frame A at time t_i.

We make use of a kinematic-inertial state estimator which tracks the pose of the pelvis in the world frame, ${}^{(ki)}\boldsymbol{T}_{W \to P_t}$. Our visual SLAM system tracks the pose of the camera in the world, ${}^{(vt)}\boldsymbol{T}_{W \to C_t}$ using consecutive pairs of images from the stereo camera.

The pelvis frame P and the camera frame C are connected through a non-rigid kinematic chain containing three back joints and three neck joints. Forward kinematics is used to relate measurements between the pelvis and camera frame at each time step



$^{(ki)}\boldsymbol{T}_{C_t \rightarrow C_{t+1}}$
C_t
C_{t+1}
$^{(vt)}\boldsymbol{T}_{W \rightarrow C_t}$
$^{(vt)}\boldsymbol{T}_{C_t \rightarrow C_{t+1}}$
$^{(fk)}\boldsymbol{T}_{P_t \rightarrow C_t}$
P_t
P_{t+1}
$^{(ki)}\boldsymbol{T}_{W \rightarrow P_t}$
World
Time *t*
Time *t+1*

Figure 3.4: The major coordinate frames of our system. These frames are connected via a time-varying kinematic tree.

t as $^{(fk)}\boldsymbol{T}_{P_t \rightarrow C_t}$.

Figure 3.4 shows the kinematic tree of the robot describing the coordinate frames and corresponding transforms between the different sensors.

Our system is based on the fusion of the kinematic-inertial state estimator and visual SLAM. We now describe these individual sources of information.

3.3.2 Kinematic-Inertial State Estimation

The kinematic-inertial state estimator used within our system is based on the approach of Koolen *et al.* [90].

This is an EKF-based state estimator which computes the 3D orientation, linear velocity and position of the pelvis root link. It makes use of sensor measurements from the six joints in each leg, force-torque sensors in each foot and an IMU rigidly attached to the pelvis.

The IMU implements an on-board EKF which fuses 3-axis accelerometer and gyroscope measurements to produce 3D orientation estimates that do not drift about the roll and pitch directions. This orientation estimate is used directly within our own

EKF.

The force torque sensors in the feet are used to estimate when the foot has made reliable contact with the ground. Once this is established, the joint encoders and known robot models are used to compute the 3D translation and velocity of the pelvis, through leg odometry.

This pose estimate is computed at high frequency (250 Hz), low latency (2-3 msec) and remains aligned to gravity. Its directions of drift are along yaw rotation and all linear degrees of freedom. Thus, the system overall has 4 degrees of freedom for drift.

This state estimator was originally implemented on the Atlas humanoid robot and demonstrated within the DARPA Robotics Challenge. Subsequently, it has been integrated within Valkyrie and provides direct input to the low-level control system.

Motion prior computation The state estimator computes the continuously evolving pelvis pose of the robot, ${}^{(ki)}\boldsymbol{T}_{W \to P_t}$. Using the pelvis pose estimates corresponding to the timestamps of consecutive images, t to $t+1$, and the pelvis-to-camera forward kinematics, the incremental motion of the camera can be computed as follows:

$$ {}^{(ki)}\boldsymbol{T}_{C_t \to C_{t+1}} = \left({}^{(ki)}\boldsymbol{T}_{W \to P_t} {}^{(fk)}\boldsymbol{T}_{P_t \to C_t} \right)^{-1} \left({}^{(ki)}\boldsymbol{T}_{W \to P_{t+1}} {}^{(fk)}\boldsymbol{T}_{P_{t+1} \to C_{t+1}} \right) . \tag{3.1} $$

In practice, state estimation measurements will not be computed for the exact same time stamps of the camera images. To account for this, we interpolate state estimation poses to generate poses for the camera time stamps.

The performance of this estimator is evaluated in Section 3.4.

3.3.3 Visual SLAM Baseline System

We implement our visual SLAM system by extending ElasticFusion [54], a real-time dense SLAM system for RGB-D cameras. This system was chosen as a baseline because it is designed to handle locally loopy trajectories typically performed by our robot. In addition, it builds dense 3D models of the environment which can be used for collision-free motion planning.

The visual odometry component of ElasticFusion is described in Section 2.2.3.1. The contribution of this chapter is a procedure to incorporate information from the robot's proprioceptive sensors. This is described next.

3.3.4 Visual Odometry and Motion Prior Integration

The fusion of a kinematic-inertial state estimator with visual SLAM is desirable because the modalities are complementary: the state estimator can handle degenerate cases where the vision system fails entirely, such as a lack of visual features or changes in illumination, while at the same time it provides information about global roll and pitch through the IMU. Through force-torque and joint encoder sensing, we can reliably estimate the linear velocity of the robot. Our goal is to limit the typical drift of the kinematic-inertial estimate through frame-to-model alignment and loop closures as performed by ElasticFusion.

Given the cumulative rigid body motion as sensed through kinematic-inertial measurements between two consecutive image frames $\boldsymbol{\xi}_{ki} = \log({}^{(ki)}\boldsymbol{T}_{C_t \to C_{t+1}})$ taken from Equation 3.1, we define a residual $\boldsymbol{r}_{ki}$ which computes the error between the current incremental estimate $\boldsymbol{\xi}$ and the kinematic-inertial estimate $\boldsymbol{\xi}_{ki}$:

$$\boldsymbol{r}_{ki}(\boldsymbol{\xi}) = \log(\exp(\boldsymbol{\xi}_{ki}) \exp(\boldsymbol{\xi})^{-1}) \; . \tag{3.2}$$

Note that as opposed to the scalar vision-related residuals r_{rgb}, r_{icp} (Equations 2.24, 2.26), $\boldsymbol{r}_{ki}$ is a 6×1 dimensional vector. The corresponding energy term is defined as:

$$E_{ki}(\boldsymbol{\xi}) = \boldsymbol{r}_{ki}(\boldsymbol{\xi})^\top \boldsymbol{r}_{ki}(\boldsymbol{\xi}) \; . \tag{3.3}$$

This term is added to the global energy function in Equation 2.28 with a corresponding weight q:

$$E(\boldsymbol{\xi}) = wE_{rgb}(\boldsymbol{\xi}) + E_{icp}(\boldsymbol{\xi}) + qE_{ki}(\boldsymbol{\xi}) \; . \tag{3.4}$$

The solver is initialised with a motion estimate $\boldsymbol{\xi} = \mathbf{0}_{6\times1}$ and incremental updates $\Delta\boldsymbol{\xi}$ are computed by solving the updated linear system of equations (corresponding to Equation 2.29):

$$(w\boldsymbol{J}_{rgb}^\top \boldsymbol{J}_{rgb} + \boldsymbol{J}_{icp}^\top \boldsymbol{J}_{icp} + q\boldsymbol{J}_{ki}^\top \boldsymbol{J}_{ki})\Delta\boldsymbol{\xi} = -(w\boldsymbol{J}_{rgb}^\top \boldsymbol{r}_{rgb} + \boldsymbol{J}_{icp}^\top \boldsymbol{r}_{icp} + q\boldsymbol{J}_{ki}^\top \boldsymbol{r}_{ki}) \; . \tag{3.5}$$

In future work we will explore using the kinematic-inertial estimate during initialisation (*i.e.* $\boldsymbol{\xi} = \boldsymbol{\xi}_{ki}$) as this should result in faster convergence for the solver.

Similarly to the vision-specific Jacobians in Equation 2.29, the kinematic-inertial Jacobian $\boldsymbol{J}_{ki}$ is defined as:

$$\boldsymbol{J}_{ki} = \left. \frac{\partial \boldsymbol{r}_{ki}}{\partial \boldsymbol{\xi}} \right|_{\boldsymbol{\xi}=\mathbf{0}} . \tag{3.6}$$

It is possible to still evaluate $\boldsymbol{J}_{ki}$ at $\boldsymbol{\xi} = \mathbf{0}$ even after $\boldsymbol{\xi}$ is updated. This can be done by an operation equivalent to image warping, but applied to the kinematic-inertial motion estimate $\boldsymbol{\xi}_{ki}$: after every iteration, one can compute an artificial kinematic-inertial motion estimate $\hat{\boldsymbol{\xi}}_{ki}$ by subtracting the current estimate of the camera motion until this point in time, $\boldsymbol{\xi}$:

$$\hat{\boldsymbol{\xi}}_{ki} = \log(\exp(\boldsymbol{\xi}_{ki}) \exp(\boldsymbol{\xi})^{-1}) . \tag{3.7}$$

This value is then used to substitute $\boldsymbol{\xi}_{ki}$ every time residuals $\boldsymbol{r}_{ki}$ are recomputed (Equation 3.2) and evaluated at $\boldsymbol{\xi} = \mathbf{0}$.

Afterwards, $\boldsymbol{\xi}$ is updated between consecutive iterations (k and $k+1$) as explained in Equation 2.31 and after the solver has converged, the global pose is updated according to Equation 2.32. For clarity, Algorithm 2 reproduces the procedure for frame-to-model alignment in ElasticFusion (described in Algorithm 1), while highlighting in blue the changes required for motion prior fusion.

It is worth noting that this system drifts along all 6 dimensions due to the motion prior being integrated only as a relative pose constraint. In future work we will investigate techniques to integrate the gravity-aligned orientation provided by the IMU to achieve similar drift-free estimates along the roll and pitch directions.

Weighting Terms in the Tracking Cost Function A point to consider is choosing how to balance the numerical contribution of each error term within the tracking cost function (Equation 3.4).

The kinematic-inertial term provides a single constraint between the previous pose estimate and the kinematic-inertial pose (Equation 3.2).

However, the ICP and RGB alignment procedures impose one constraint per pair of matched pixels. This results in an imbalanced number of constraints, and, if not considered, the kinematic-inertial term would have inconsequential influence.

We implement a simple heuristic for scaling the contribution of the kinematic-inertial term to have a sufficient influence on the combined motion estimate. Given

Algorithm 2: Frame-to-model alignment in Humanoid SLAM System

```
Input: $I_D, Z_D, I_M, Z_M, \boldsymbol{\xi}_{ki}$
compute image pyramids for $I_D, Z_D, I_M, Z_M$;
initialise $\boldsymbol{\xi} = \mathbf{0}_{6\times1}$;
initialise $\hat{\boldsymbol{\xi}}_{ki} = \boldsymbol{\xi}_{ki}$;
foreach level of the pyramid do
    foreach iteration do
        compute ICP correspondences given current $\boldsymbol{\xi}$;
        compute RGB correspondences given current $\boldsymbol{\xi}$;
        compute visual odometry residuals $\boldsymbol{r}_{icp}$ and $\boldsymbol{r}_{rgb}$;
        compute kinematic-inertial residual $\boldsymbol{r}_{ki}$;
            $\boldsymbol{r}_{ki}(\boldsymbol{\xi}) = \log(\exp(\hat{\boldsymbol{\xi}}_{ki}) \exp(\boldsymbol{\xi})^{-1})$;
        solve for $\Delta\boldsymbol{\xi}$;
            $(w\boldsymbol{J}_{rgb}^\top \boldsymbol{J}_{rgb} + \boldsymbol{J}_{icp}^\top \boldsymbol{J}_{icp} + q\boldsymbol{J}_{ki}^\top \boldsymbol{J}_{ki})\Delta\boldsymbol{\xi} =$
          $-(w\boldsymbol{J}_{rgb}^\top \boldsymbol{r}_{rgb} + \boldsymbol{J}_{icp}^\top \boldsymbol{r}_{icp} + q\boldsymbol{J}_{ki}^\top \boldsymbol{r}_{ki})$;
        update $\boldsymbol{\xi}$;
            $\boldsymbol{\xi} = \log(\exp(\boldsymbol{\xi}) \exp(\Delta\boldsymbol{\xi}))$;
        update $\hat{\boldsymbol{\xi}}$;
            $\hat{\boldsymbol{\xi}}_{ki} = \log(\exp(\boldsymbol{\xi}_{ki}) \exp(\boldsymbol{\xi})^{-1})$
```

the proportion of inliers for the ICP and RGB alignment procedures relative to the number of pixels in the image (*i.e.* the percentages ICP_p, RGB_p), we define a corresponding proportional term for the kinematic-inertial measurement:

1. In a degenerate situation where the proportion of inliers for both ICP and RGB alignment procedures is low ($ICP_p, RGB_p < 5\%$), we trust the kinematic-inertial estimate fully ($KI_p = 100\%$).

2. We observed that in well structured environments, the kinematic-inertial term should contribute slightly more than one third to the total pose error minimisation. Therefore, we set $KI_p = max(ICP_p, RGB_p) + \alpha$, where $\alpha = 10\%$ was a suitable value in our evaluation.

The contributions from the ICP, RGB and kinematic-inertial components are evenly balanced within a well structured scene. However, for sequences with strong lighting changes or where moving objects occupy a large portion of the field of view, the kinematic-inertial component dominates. The weight q (used in Eq. 3.4) is the result of applying the percentage KI_p onto the number of pixels in the image.

Although not addressed here, an additional strategy could be formulated to handle failures in the state estimator. For example, foot slippage could be detected from

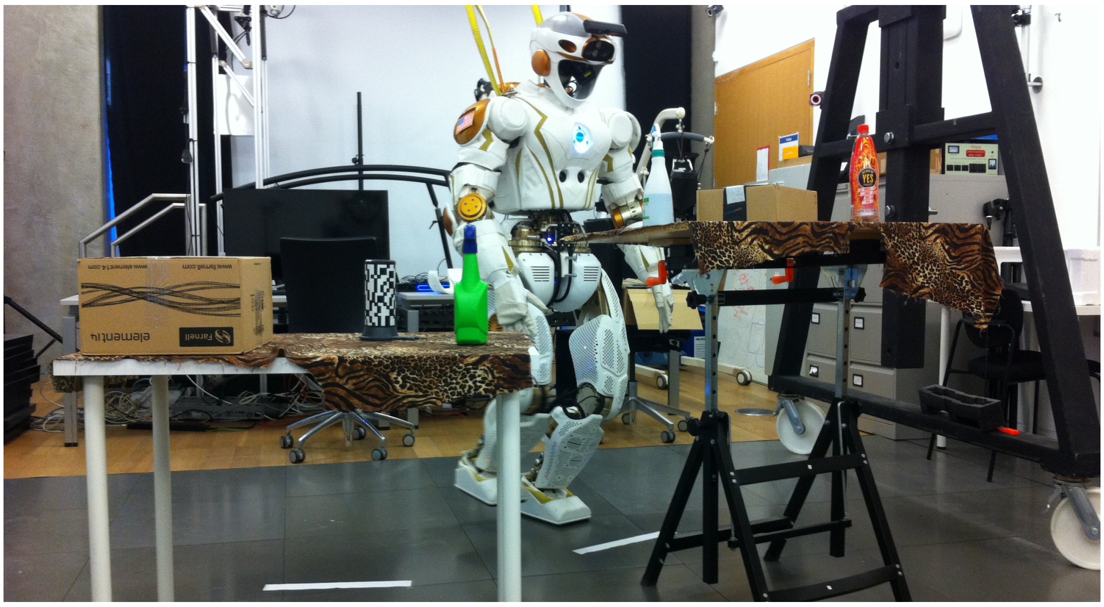


Figure 3.5: The NASA Valkyrie humanoid robot during operation in a laboratory environment. Note the markers on the ground, which are discussed in Section 3.4.3.

unexpected spikes in velocity and the influence of the kinematic-inertial term could be reduced during these sequences.

This aspect was not addressed within our work due to practical reasons: it is very difficult to provoke a foot slippage to our robot that does not result in a loss of balance and thus a fall of the robot.

3.4 Evaluation

In this section we present an evaluation of our method on a series of experiments with the Valkyrie humanoid robot. The test environment consists of a manipulation scene containing several tables with objects on them surrounding the robot (Figure 3.5).

Our laboratory is equipped with a Vicon motion capture system which provides ground truth trajectory measurements of the pelvis pose. Motion capture markers are placed on known locations on the pelvis, as shown in Figure 3.6. Also, the Vicon computer clock is synchronised with the robot workstation clock through the network.

The dataset used as part of this evaluation is described in Table 3.1. Log1 is a short walking sequence of a single loop trajectory within a static feature-rich environment (*i.e* the ideal operating scenario). During this log we also took care not to perform any fast motions of the neck. Log2 is a longer and locally loopy trajectory containing several visual challenges which typically cause baseline visual SLAM systems to fail:

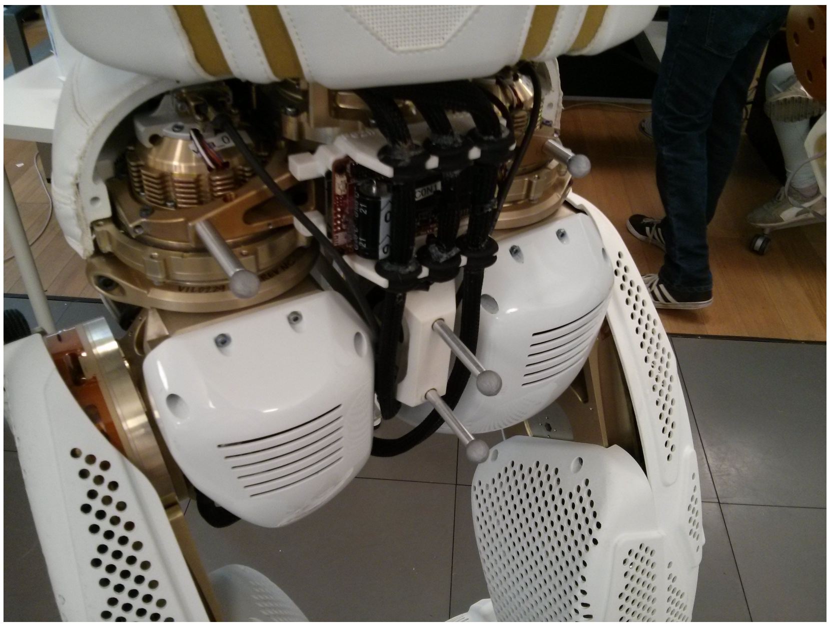


Figure 3.6: Vicon markers are placed on the pelvis to track the pose of the robot during walking.

Dataset Description							
Log	Distance (m)	Steps	Time (s)	Lack of Features	Lights Off	Motion Blur	Continuous Dynamics
Log1	18.5	34	485	✗	✗	✗	✗
Log2	62.96	102	1204	✓	✓	✓	✓

Table 3.1: Description of the dataset used in this evaluation. ✓/✗ indicates the presence/absence of a certain challenge.

- Lack of features in the camera view: the camera points at a blank wall while the robot turns ($\sim$ 20 sec per sequence).
- Changes in illumination: by turning the room's lighting on and off ($\sim$ 15 sec per sequence).
- Motion blur: by performing fast head motion ($\sim$ 5 sec per sequence).
- Continuous dynamics in the scene: by introducing moving objects and people covering more than 50% of the field of view of the camera ($\sim$ 5 sec per sequence).

We start by assessing the tracking performance of the proposed method against the original ElasticFusion system and the robot's kinematic-inertial state estimator for the two collected logs. We demonstrate that the proposed approach overcomes the typical limitations of visual tracking during challenging situations and corrects state estimation drift through the implementation of loop closures.

Following this, we evaluate the accuracy of the stereo reconstruction against LIDAR point clouds as produced by the Hokuyo UTM-30LX-EW spinning planar LIDAR contained within the MultiSense SL.

Finally, we demonstrate the integration of our algorithm within the closed-loop walking controller of the Valkyrie humanoid robot.

In our evaluation we refer to three different estimators:

- EF: ElasticFusion running on stereo data and with loop closures enabled (12 Hz).
- KI: The open-loop kinematic-inertial state estimator which is also used in the control loop (250 Hz).
- PEF: Proprioceptive ElasticFusion - our proposed system which fuses the visual and kinematic-inertial systems and with the default ElasticFusion loop closure capabilities enabled (12 Hz).

3.4.1 Quantitative Results (Trajectory Evaluation)

We evaluate the performance of these estimators by comparing trajectories against the Vicon ground truth measurements. Note that although the Vicon markers are mounted on the pelvis, the visual SLAM system estimates the pose of the camera in the head. Therefore, we use forward kinematics to transform all motion capture and kinematic-inertial pose estimates to the camera reference frame and evaluate all trajectories with respect to this frame.

For this evaluation, we use the established metrics proposed by Sturm *et al.* [26], namely Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). In addition, we report the Drift per Distance Travelled (DDT). These metrics are described in detail in Section 2.3. We compute these metrics over all timestamps and report the root mean square error (RMSE) for each trajectory.

Regarding DDT, we compute the total distance travelled by integrating the length of the path travelled by the camera for each time sample. As the typical robot walking gait involves oscillatory motion (left-right-left), this can overstate what is considered as 'distance travelled'.

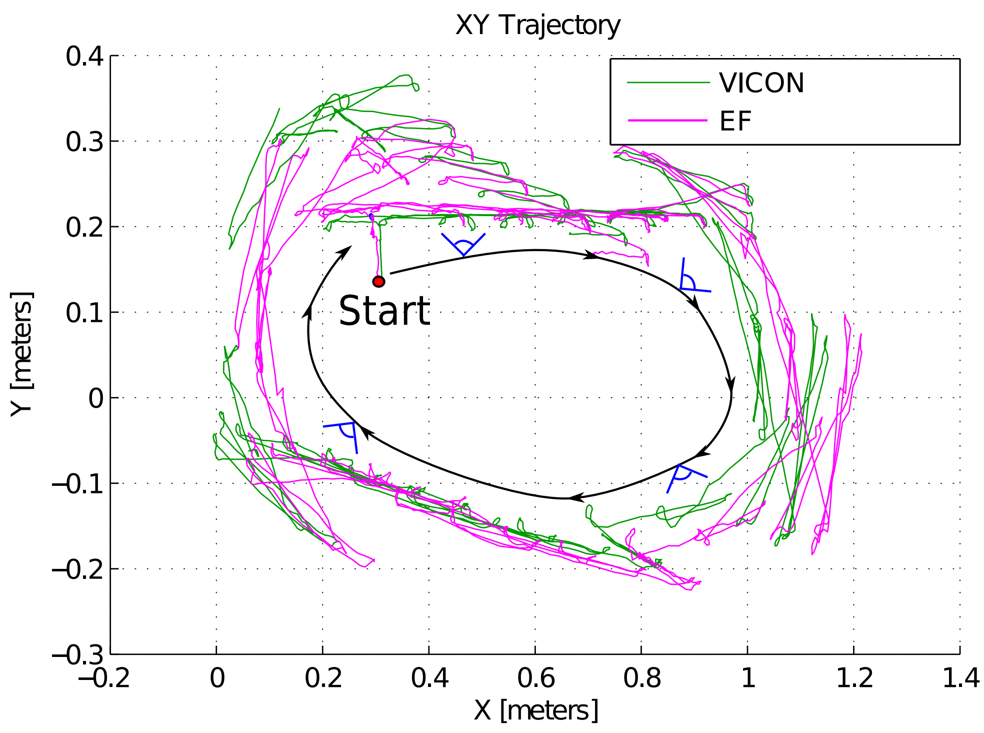


Figure 3.7: Overhead view of the camera trajectory as estimated by our ground truth Vicon system (green) and ElasticFusion using stereo data only (magenta) for Log1. The direction of motion is indicated by the black line and the blue frames indicate the point of view of the robot (facing outside).

3.4.1.1 Stereo ElasticFusion

We first explored the performance of ElasticFusion (EF) on stereo data preprocessed as described in Section 3.2.3. The robot was commanded to walk clockwise up to the point of completing a loop in Log1 and Figure 3.7 shows the trajectory as estimated by ElasticFusion. Despite vibrations due to foot impacts and some sharp rotations, the estimated motion closely matches the Vicon trajectory — in large part because the environment contained structure in all directions. This indicates that stereo-only EF can achieve acceptable tracking performance in feature rich environments assuming smooth camera motion.

Nevertheless, even in scenarios where visual SLAM can operate successfully, there are still benefits to performing sensor fusion. In this case, this is motivated by the high stability and temporal smoothness of estimates provided by the kinematic-inertial system. Figure 3.8 shows that the stability of PEF pose estimates are indeed higher when compared to EF. When tracking against the model, EF estimates display high frequency jitter because of independent per-frame geometric/photometric optimisa-

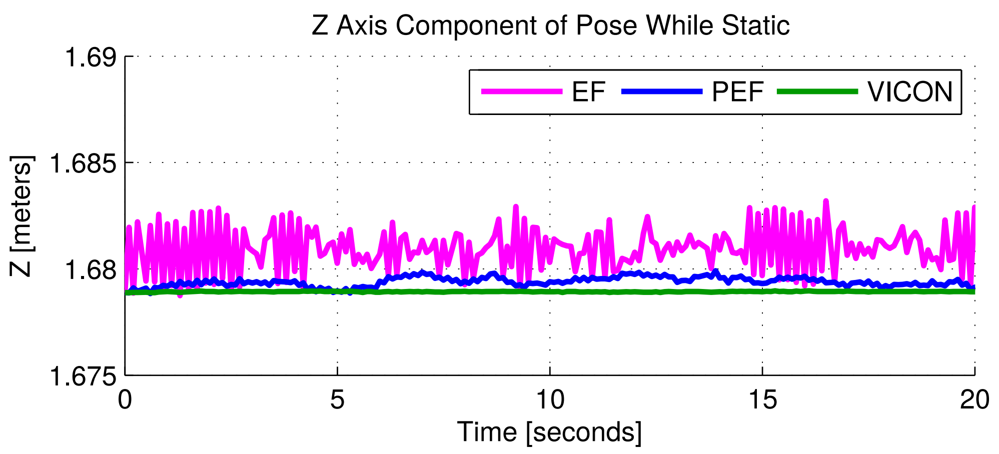


Figure 3.8: Z-component of the robot's pose for a sequence in Log1 when it is stationary: fusion of kinematic-inertial within visual tracking (PEF) results in more stable pose estimate than using only vision (EF).

tions. Note that the vision system could be simply turned off while the robot is standing, nevertheless this smoothness would still be observable when moving.

3.4.1.2 Evaluation of Accuracy

We analyse the drift characteristics of all estimators for the two sequences and report the DDT in Table 3.2. As EF uses frame-to-model tracking, we evaluate the drift of this system by limiting the size of the local tracking model to the previous 200 frames only.

	Log1			Log2		
DDT	EF	KI	PEF	EF	KI	PEF
XYZ (cm/m)	1.06	0.72	**0.61**	18.55*	0.78	**0.54**
XY (cm/m)	1.00	0.54	**0.53**	17.06*	0.56	**0.45**
Z (cm/m)	0.35	0.48	**0.30**	7.28*	0.54	**0.29**
Yaw (deg/m)	0.33	0.53	**0.16**	6.51*	0.38	**0.19**

Table 3.2: Drift per distance travelled averaged over 2 m to 10 m trajectory intervals. The proposed fusion approach, PEF, outperforms the individual sub-systems. Note * indicates algorithm failure.

For the kinematic-inertial state estimator, the main directions of drift are in the 3D linear axes and yaw rotation due to estimator unobservability in those directions. While roll and pitch are globally observable through the accelerometer, yaw is computed by integrating the rate gyroscope estimates and drifts over time. Linear Z-axis drift occurs due to small errors made whenever contact between the robot's feet and the ground is established.

PEF achieves the best performance for each portion of the state with an average translational drift of 0.54 cm/m for Log2. The baseline EF system is unable to operate on this sequence and fails.

A more detailed view of the rate of translational drift as a function of the path length for Log1 is shown in Figure 3.9. In this case, the relative pose error for PEF grows at the slowest rate.

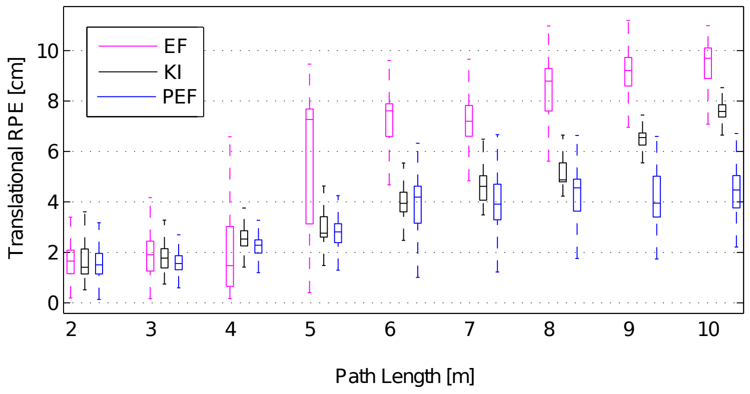


Figure 3.9: Translational RPE (cm) for increasing path lengths (m) for Log1, showing PEF achieving the smallest drift rate.

3.4.1.3 Evaluation of Robustness

Within this section we report the performance of our proposed approach on the challenging sequences within Log2 which cause the baseline EF system to fail.

Figure 3.10 shows examples of such challenges and the successful operation of PEF in these situations. For each example, the performance of PEF was as follows:

- Lack of features in the camera view (Figure 3.10a): the lack of suitable depth causes the kinematic-inertial tracking to dominate Equation 3.4.
- Motion blur (Figure 3.10b): while depth can be estimated in this case, the set of inliers is very small, meaning again kinematic-inertial tracking is preferred.
- Changes in illumination (Figure 3.10c): behaviour is similar to previous case.
- Continuous dynamics in the scene (Figure 3.10d): in this situation, our approach exploits the 3D fusion strategy of ElasticFusion, which only adds new objects in the map if these are observed repeatedly. Note that changes to the map are not supported.

The absolute trajectory error is shown in Table 3.3: our proposed system achieves low error for both sequences and successfully tracks the pose of the camera through all mentioned challenges.

	Trans. ATE RMSE (m)	
Logs	EF	PEF
Log1	0.048	**0.020**
Log2	FAIL	**0.025**

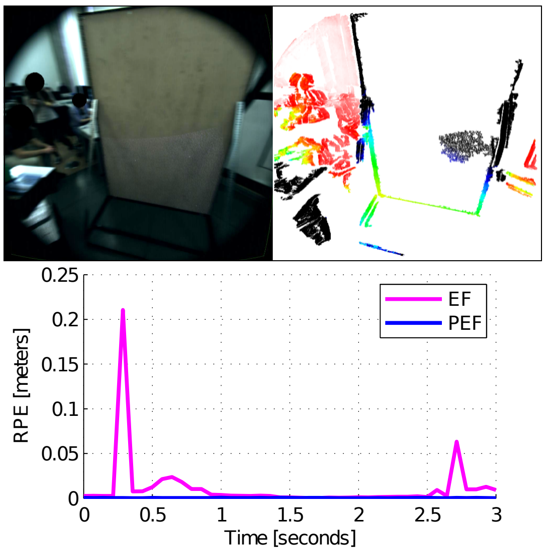
Table 3.3: Absolute Trajectory Error for the two collected sequences. ElasticFusion (EF) fails on Log2 because of these challenges.

3.4.2 Qualitative Results (Comparison of Stereo map against LIDAR map)

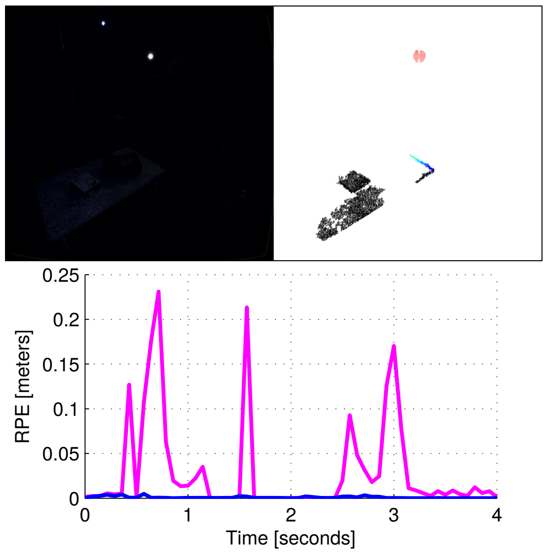
In Figure 3.11 (top) we present a 3D model showing the reconstruction obtained during Log2. We evaluate its accuracy against a model created using the spinning Hokuyo LIDAR sensor contained within the MultiSense SL. The accuracy of the LIDAR sensor is +/-10 mm within the range of 0.1 m - 10 m, which makes it appropriate for coarsely evaluating the stereo reconstruction. We manually align several LIDAR point clouds from Log2 to create a 3D model of the test environment. Due to imperfections in how the LIDAR map is produced, we point out that these results are indicative of reconstruction quality rather than fully quantitative.

For each point in the visual model, we compute the distance to the closest LIDAR point as a per-point error. A heat-map of this error is shown in Figure 3.11. The scale and structure of the stereo model closely matches the LIDAR model. The stereo model is aligned with gravity which is essential for it to be used in practical applications. This occurs due to the fact that the initial camera pose within the visual SLAM system is set to the initial pose of the kinematic-inertial estimate, whose roll and pitch estimates are computed from the accelerometer.

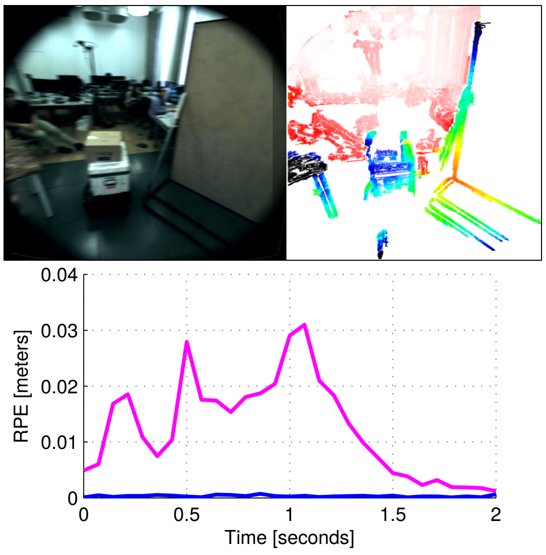
Figure 3.12 shows the distribution of per-point errors, with a median error value of 0.02 m. We conclude the reconstruction is of sufficient accuracy for tasks such as collision free motion planning.



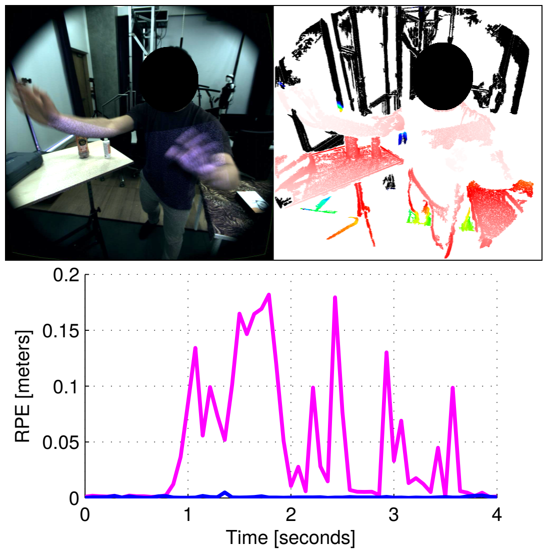
(a) Featureless Scene



(c) Turning off Lights



(b) Motion Blur



(d) Dynamics in the Scene

Figure 3.10: Top: Examples of colour-disparity image pairs during challenging sequences. Bottom: Corresponding frame-to-frame translational RPE for PEF and the baseline EF. In each case EF fails while PEF works well.

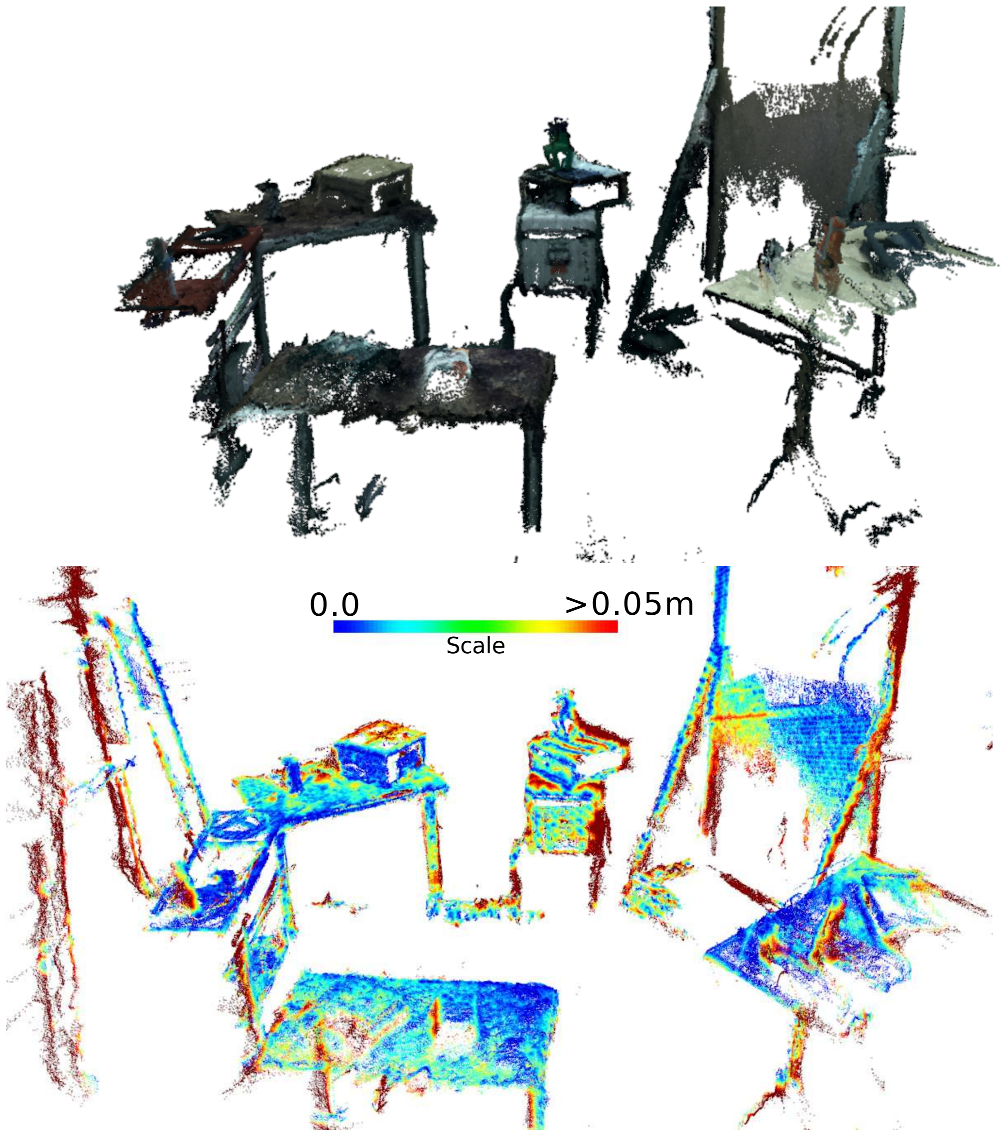


Figure 3.11: **Top**: Stereo 3D reconstruction of a manipulation scene covering a $12\,\text{m}^2$ area. **Bottom**: Heat map of stereo per-point error computed against the LIDAR point cloud.

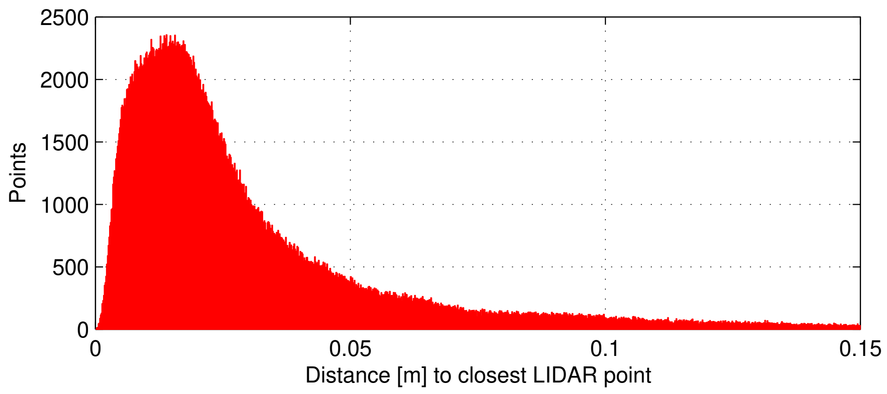


Figure 3.12: Distribution of stereo point errors.

3.4.3 Closed Loop Integration

In our final experiment we demonstrate the integration of PEF within the closed loop walking controller of the Valkyrie robot. We encourage the reader to watch the video of the experiment in Appendix A.3.1 for a better understanding.

The environment used for this experiment is depicted in Figure 3.5. It consists of two tables with objects on them and corresponding goal marks on the floor (white tape). The robot walks to each table in turn to reach these goals.

The kinematic-inertial state estimator (KI) represents the direct input to the walking control system of the robot. To prevent the state estimator from drifting, we transmit pose corrections on a regular interval from our method (PEF) which enables the robot to successfully reach the targets each time.

3.5 Discussion

In this chapter we proposed an approach for visual SLAM applied to the humanoid robot Valkyrie. We aimed for an approach that does not require control of how the camera moved.

This meant that we had to handle a number of challenges. For example, the robot could point its camera towards areas with few visual features or can perform fast motion, inducing motion blur and strong view changes within the images. We also analysed the problems posed by strong changes in illumination or the presence of moving objects in the image. These can affect the pose tracking component of visual SLAM systems, causing them to fail, and in turn lead to corrupted reconstructions of the scene.

To address these challenges, we extended the dense visual SLAM method ElasticFusion to integrate information from our high-rate low-drift kinematic-inertial state estimator. We use the state estimator to provide a camera motion prior which is integrated within the pose tracking component of ElasticFusion to handle the described degenerate cases. As many previous approaches made use of sparse point-based SLAM methods, our direct approach can produce a semi-dense reconstruction which can also be interpreted visually and used for tasks such as collision-free motion planning.

We evaluated our approach through a series of experiments in our laboratory. Our fusion method achieves lower drift rates than the tracking of the kinematic-inertial state estimator and ElasticFusion's visual tracking individually but more importantly it is robust to sequences containing the aforementioned visual challenges. We provided a qualitative evaluation of our stereo-produced reconstruction against LIDAR and described an online integration experiment of our method within the walking controller of Valkyrie.

3.5.1 Limitations and Future Work

The system could be further improved by integrating global roll and pitch information from the accelerometer to prevent drift along these dimensions. Laidlow *et al.* [69] implement this by constraining the graph deformation procedure to deform the map according to gravity. Alternatively, Puri *et al.* [103] embed global roll and pitch information within each surfel to enable accurate reconstructions of planar floors.

Also, we did not explore the performance of our fused system in cases where the proprioceptive state estimator might fail. For example, the state estimator reports excessively high velocity values when foot slippages occur, in turn leading to an incorrect kinematic-inertial motion prior. The influence of an incorrect motion prior within our fusion system was not investigated due to practical limitations, as it is difficult to induce controlled slippage which would not cause the robot to fall.

In future work, this problem could be addressed on a simpler robotic platform. Wheeled vehicles are also equipped with encoders and IMUs which can be used to compute proprioceptive odometry and provide motion priors. Such vehicles also encounter slippage, but are less prone to damage in these situations, particularly those with more stable four-wheel design.

Finally, our method is only implicitly robust to dynamics in the scene. We exploit the fact that dynamic objects are assigned low confidence and do not become a part of the model as long as they are continuously moving.

In the next chapters, we describe approaches taken to explicitly handle the presence of moving objects in the environment and support changes to the map.

Chapter 4

Joint Visual Odometry and Motion Segmentation for SLAM in Dynamic Environments

In this chapter we explore approaches to improve robustness of visual SLAM systems in dynamic environments. In the previous chapter we leveraged motion priors from other sensors and relied on simple outlier filtering to handle the presence of moving objects. Here we explore a vision-based approach to explicitly detect moving objects and remove them from our camera pose estimation process.

Robust operation in the presence of dynamic elements is an open problem in visual SLAM. For example, many visual odometry methods are based on the assumption that camera motion is responsible for the change between two consecutive images. Dynamic elements violate this assumption and can cause failures in pose tracking. In addition, if not actively detected and segmented, dynamic objects can be fused into the map which can lead to irreversible corruptions.

In this chapter we propose a method for robust dense RGB-D SLAM in dynamic environments which detects moving objects and reconstructs the background structure in real-time.

The novelty of our approach is a new formulation that simultaneously estimates the camera motion as well as a probabilistic static/dynamic segmentation of the current RGB-D image pair. This segmentation is then used to weight the dense RGB-D fusion to estimate a 3D model of only the static parts of the environment.

We demonstrate that by leveraging the 3D model for frame-to-model alignment, as well as static/dynamic segmentation, camera motion estimation has reduced overall drift — as well as being more robust to the presence of dynamics in the scene. We compare our approach to related state-of-the-art methods using both static and dynamic sequences and report that the proposed method achieves similar performance in static environments and improved accuracy and robustness in dynamic scenes.

4.1 Visual SLAM in Dynamic Environments

Improving the performance of SLAM in dynamic environments is an important goal particularly for mobile robots. It is rarely the case that robots operate in strictly static environments. Such a requirement would significantly limit the extent to which they could be successfully deployed. For example, co-bots such as the Rethink Baxter carry out assembly tasks among moving equipment and infrastructure. Similarly, mobile service robots accomplish their tasks in environments which are inhabited by people.

Efforts have been made to increase the robustness of visual SLAM in dynamic scenes. Among current approaches, we identify two main strategies:

- Outlier Filtering - these methods consider moving objects as outliers and filter them out using hard-coded thresholding techniques, semantic segmentation or robust cost functions.

- Spatial or Temporal Modelling - these methods actively detect moving objects and reconstruct and track them in time.

We review works implementing both approaches in the next section.

4.1.1 Outlier Filtering

Implicit Handling of Dynamic Elements It is common to use a robust cost function within visual odometry which penalises the contribution of high-residual points and implicitly increases the robustness of pose estimation to un-modelled effects. Gutierrez *et al*. [82] compare different robust functions focusing on the quality of the resulting pose estimate, while Kerl *et al*. [60][104] demonstrate robustness to the presence of small moving objects in the scene by using a robust penalty.

These solutions however are insufficient and fail when moving objects occupy a significant portion of the image.

Reconstruction-focused approaches, such as ElasticFusion [54] as well as the method of Keller *et al.* [53], require points be repeatedly observed through consecutive frames before becoming integrated within the 3D model.

In these methods dynamic elements are not explicitly detected and handled, resulting in robustness which is limited to small motions of the objects in the scene.

Outlier Rejection Strategies A common strategy is to treat dynamically moving objects as noise which must be detected and filtered out.

For Keller *et al.* [53], input points with no close model correspondence are used to seed a region-growing procedure to segment the current image into static and dynamic parts. Subsequently, model points which are matched with dynamic input points are removed from the reconstruction. This approach can only be demonstrated once a confident reconstruction of the scene is in place.

In DTAM [105], which is a monocular dense mapping system, the authors discard pixels with a photometric error higher than a specific threshold. For ORB-SLAM [106, 24] the authors enforce an effective survival-of-the-fittest strategy which judges the validity of keyframes and the points used for pose tracking. By being generous when spawning new keyframes and points within the system and by enforcing highly conservative culling strategies, they demonstrate impressive robustness and versatility.

Nevertheless, within these approaches, no spatial or temporal coherence is enforced among the detected dynamic points between consecutive frames.

4.1.2 Methods Enforcing Spatial or Temporal Coherence

Jaimez *et al.* [79] introduce a joint visual odometry and scene flow estimation method. Similarly, BaMVO [107] is an odometry method which reconstructs the environment over the previous 4 frames by temporal propagation. As both are frame-to-frame methods, they incur unbounded drift in the pose estimate over time and struggle when dynamics occupy a majority of the image.

Rünz *et al.* proposed Co-Fusion [108], a method to reconstruct and track each moving object with a separate 3D model, being one of the first real-time methods to perform dense tracking and fusion of multiple objects so as to explicitly handle dynamics and enforce both spatial and temporal coherence. Moving objects are detected through motion segmentation computed using ICP or semantic segmentation using the method of Pinheiro *et al.* [109]. This technique is extended in MaskFusion [110], where moving objects are detected using Mask-RCNN [111]. The limitation of these approaches is that camera pose estimation is computed through dense visual odometry which could fail when the moving objects occupy a large portion of the image.

Finally, methods such as DynamicFusion [112] and VolumeDeform [113] are impressive techniques which are specifically targeted towards dynamic environments. They estimate a non-rigid 3D model of the scene and track its motion using scene flow. However, they require that objects move smoothly and slowly to successfully track them.

By coupling visual odometry and motion segmentation, our approach achieves increased robustness in dynamic scenes compared with state-of-the-art techniques. While effective detection and segmentation of moving objects typically require temporal feedback or a multi-frame formulation, we demonstrate that background 3D reconstruction is an efficient way to propagate this temporal information without incurring significant run-time costs. Also, the resulting map is more meaningful in the sense that it only contains structural elements and the static objects present in the scene.

4.2 Methodology

4.2.1 System Overview

In this section we provide a general description of our proposed SLAM system and Figure 4.1 illustrates its main components. Each of these components will be described in detail in the following sections.

The input to our system is a stream of registered RGB-D images. An RGB-D pair is represented as a colour image $C_D : \Omega \to \mathbb{R}^3$ and a depth image $Z_D : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{N}^2$ is the image plane. We also compute an intensity image $I_D : \Omega \to \mathbb{R}$ from C_D for use in the algorithm.

To achieve real-time operation, we segment every incoming pair (I_D, Z_D) into K geometric clusters and we solve the static/dynamic segmentation problem cluster-

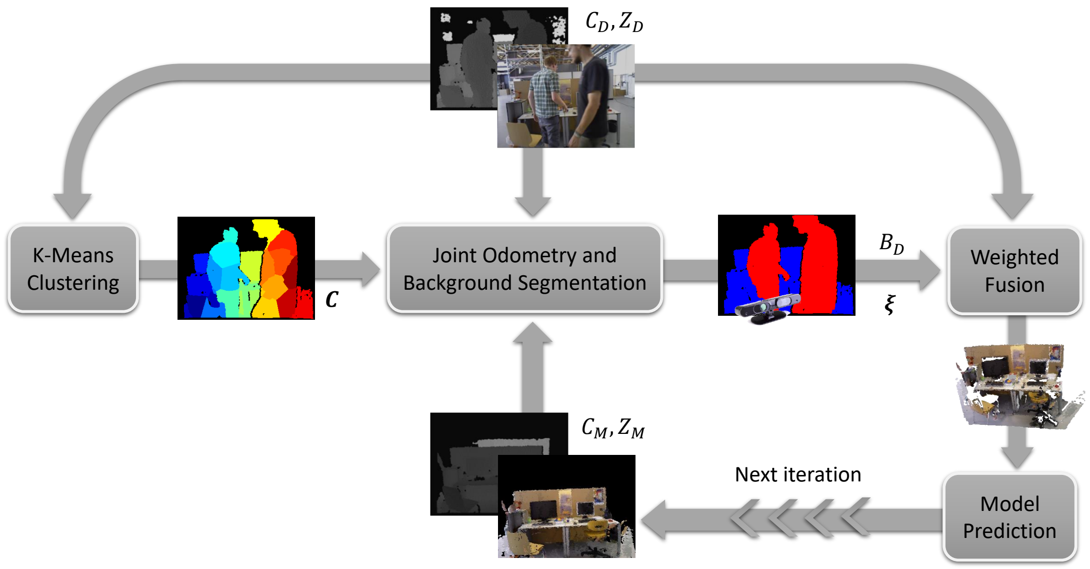


Figure 4.1: System architecture: the process starts by receiving a new RGB-D image (C_D, Z_D) and grouping its pixels into geometric clusters $\boldsymbol{C}$. A prediction (C_M, Z_M) is rendered from the model and the last pose estimate $\boldsymbol{T}$ and used for joint alignment and background segmentation. Both results are then exploited for weighted fusion of the static clusters of C_D, Z_D with the map.

wise as opposed to pixel-wise. The clusters $\boldsymbol{C} = \{C_i,\ i = 1, ..., K\}$ are computed using K-Means on the 3D coordinates of the scene points (as described in [79]). This is an acceptable approximation because we are not interested in estimating accurate motions of moving objects, but are rather focused on building a *conservative* reconstruction of the static structures in the scene.

Then, an artificial image pair $(I_M,\ Z_M)$ is rendered by placing a virtual camera at the previous camera pose estimate within the current map of the static scene constructed up to that point.

Given the current images (I_D, Z_D), the cluster segmentations and the last prediction (I_M, Z_M), our novel step is to jointly obtain the camera motion $\boldsymbol{\xi} \in \mathfrak{se}(3)$ and a motion-based segmentation of the scene between the two time instances. Each cluster i is assigned a score $b_i \in [0, 1]$ which corresponds to the level of dynamism:

- $b \simeq 1$ corresponds to static clusters;
- $b \simeq 0$ to moving clusters;
- $0 < b < 1$ to intermediate levels of uncertainty.

After the solution to the joint estimation problem is calculated, the clusters and scores are used to compute a per-pixel segmentation image B_D, which, together with the current colour and depth images (C_D, Z_D), is used for weighted 3D fusion.

4.2.2 Formulation of Joint Visual Odometry and Motion Segmentation

To estimate these two joint properties, we propose a new formulation based on the minimisation of two energy terms:

$$\min_{\boldsymbol{\xi},\boldsymbol{b}} \{D(\boldsymbol{\xi},\boldsymbol{b}) + S(\boldsymbol{b})\} \quad s.t.\ b_i \in [0,1] \quad \forall i, \tag{4.1}$$

where $\boldsymbol{b}$ represents the full set of scores over all clusters.

The term $D(\boldsymbol{\xi},\boldsymbol{b})$ encodes direct image alignment by enforcing photometric and geometric consistency only for pixels that belong to static clusters.

The second term $S(\boldsymbol{b})$ complements $D(\boldsymbol{\xi},\boldsymbol{b})$ by forcing clusters to be segmented as dynamic when their residuals are very high, and vice versa. It also includes spatial regularisation to encourage a smooth segmentation of the clusters, and exploits prior geometric knowledge to help the optimisation converge to the correct minimum.

Next, we present the formulation of $D(\boldsymbol{\xi},\boldsymbol{b})$ and $S(\boldsymbol{b})$ and describe how the overall minimisation problem is tackled.

4.2.2.1 Camera Motion Estimation

For every new RGB-D pair, the incremental motion of the camera is computed by minimising the geometric and photometric reprojection errors between the current RGB-D image and the last prediction obtained from the map. The respective reprojection errors (or residuals) are defined as:

$$r_Z^p(\boldsymbol{\xi}) = Z_M(\mathcal{W}(\boldsymbol{x}^p,\boldsymbol{\xi})) - \left|\exp(\boldsymbol{\xi})\,\pi^{-1}(\boldsymbol{x}^p, Z_D(\boldsymbol{x}^p))\right|_z \tag{4.2}$$

$$r_I^p(\boldsymbol{\xi}) = I_M(\mathcal{W}(\boldsymbol{x}^p,\boldsymbol{\xi})) - I_D(\boldsymbol{x}^p)\ , \tag{4.3}$$

where $\boldsymbol{x}^p \in \Omega$ represents the coordinates of a given pixel p and $|\bullet|_z$ denotes the z-coordinate of a 3D point. The function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ projects 3D points onto the image plane according to the camera's pinhole model. $\exp(\boldsymbol{\xi}) \in \mathbb{SE}(3)$ computes the homogeneous transformation associated to the twist $\boldsymbol{\xi}$. The warping function is given by:

$$\mathcal{W}(\boldsymbol{x}^p,\boldsymbol{\xi}) = \pi(\exp(\boldsymbol{\xi})\,\pi^{-1}(\boldsymbol{x}^p, Z_D(\boldsymbol{x}^p)))\ . \tag{4.4}$$

These residual formulations are similar to those in the visual odometry component of ElasticFusion discussed in Section 2.2.3.1, however there are some notable differences:

- ElasticFusion aligns depth images using ICP in a point-to-plane formulation. We instead implement direct image alignment over depth images, which is a more efficient formulation that does not require computing normals.

- ElasticFusion filters outlier correspondences using hard-coded distance thresholds. Instead we follow the approaches of Kerl *et al*. [60] and Jaimez *et al*. [64] who use M-estimators to reduce the influence of high residuals which could originate from outlier correspondences. We make use of the Cauchy robust penalty, described in Section 2.1.1.3 and visualised in Figure 4.2.

Our direct image alignment formulation takes the form:

$$D(\boldsymbol{\xi}, \boldsymbol{b}) = \sum_{p=1}^{N} b_{i(p)} \left[F(w_Z^p r_Z^p(\boldsymbol{\xi})) + F(\alpha_I w_I^p r_I^p(\boldsymbol{\xi})) \right] , \tag{4.5}$$

where N is the overall number of pixels and $b_{i(p)}$ refers to the score of the cluster i containing p. We weight the residuals using $b_{i(p)}$ to ensure that those associated to static parts of the scene have a high contribution.

The function $F(r)$ is the Cauchy robust penalty:

$$F(r) = \frac{c^2}{2} \log \left(1 + \left(\frac{r}{c} \right)^2 \right) , \tag{4.6}$$

where c represents the inflection point of $F(r)$ and controls how robustly residuals are minimised.

As the geometric and intensity terms compute errors with different units, the parameter α_I re-scales the intensity term so that it has a comparable effect in scale as the geometric term.

Lastly, we follow the approach of Jaimez *et al*. [79] and implement weights w_Z and w_I which penalise the photometric and geometric residuals according to the noise of the measurements (k^Z and k^I) and the occlusions and discontinuities observed through high spatial or temporal gradients:

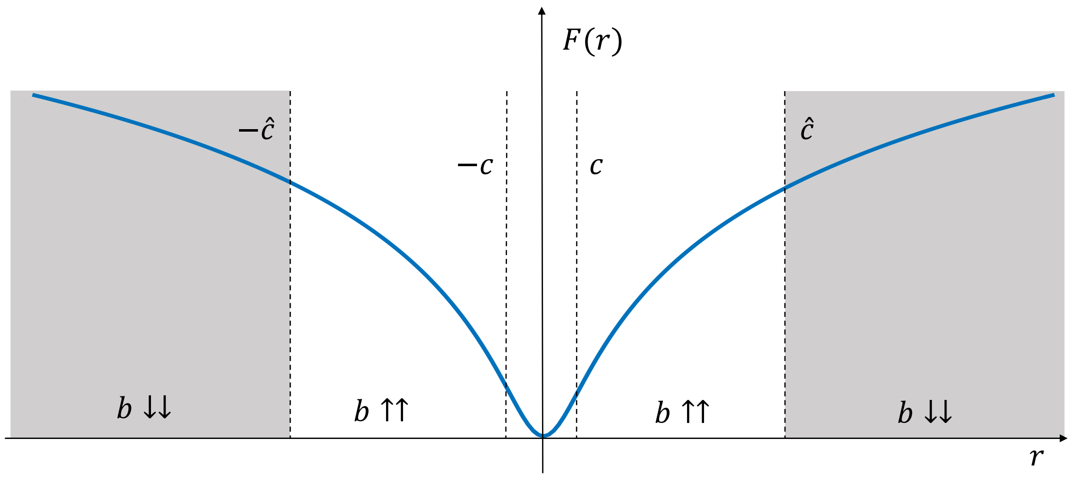


Figure 4.2: Cauchy robust penalty and the different regions defined to distinguish between the clusters that are likely to be static ($b \uparrow\uparrow$) or dynamic ($b \downarrow\downarrow$).

$$w_Z = \frac{1}{k^Z + |\nabla_x Z_D| + |Z_D - Z_M|} \;, \tag{4.7}$$

$$w_I = \frac{1}{k^I + |\nabla_x I_D| + |I_D - I_M|} \;. \tag{4.8}$$

In Equations 4.7 and 4.8, the parameters k^Z and k^I control the relative importance of the noise against the derivatives.

4.2.2.2 Static / Dynamic Segmentation

The objective of the second term in 4.1 is to classify clusters with high average residuals as dynamic and those with low residuals as static. The underlying idea is that clusters with high residuals are the ones whose relative motions with respect to the camera do not coincide with the camera motion itself. In order to implement this concept we must quantify what a 'high residual' is.

Our assumption is that large residuals correspond to those significantly higher than the parameter c, i.e. those lying on the flatter sides of the function $F(r)$ (see Fig. 4.2). The following term sets this threshold within the overall minimisation problem:

$$S_D(\boldsymbol{b}) = 2\sum_{i=1}^{K}(1 - b_i)K_i F(\hat{c}) \;. \tag{4.9}$$

The total number of pixels in each cluster i is represented by K_i, and $\hat{c} > c$ is a heuristically selected threshold which defines the frontier between low and high residuals.

The combination of this term with Eq. 4.5 encourages b_i to be as low as possible (to a minimum of 0) when the average residual of cluster i is higher than $\hat{c}$; otherwise it favours high values of b_i (to a maximum of 1).

Furthermore, we include a regularisation term that encourages contiguous clusters to have a similar score:

$$S_R(\boldsymbol{b}) = \lambda_R \sum_{i=1}^{K} \sum_{j=i+1}^{K} G_{ij} \left(b_i - b_j\right)^2 \ . \tag{4.10}$$

In Eq. 4.10, G_{ij} is a connectivity map: it is equal to 1 when clusters i and j are contiguous in space and it is 0 otherwise. Contiguity is established by first projecting the clusters onto the image plane and then checking, for each pair of clusters that share a border on the image plane, if their corresponding 3D points along that border are sufficiently close. The parameter λ_R weights $S_R(\boldsymbol{b})$ with respect to the other terms.

Lastly, we add a geometric constraint that exploits the fact that moving objects do not appear in our map and therefore the depth differences between Z_D and Z_M will be high for moving clusters. This constraint is expressed as a segmentation prior:

$$S_P(\boldsymbol{b}) = \lambda_P \sum_{i=1}^{K} \left(b_i - b_i^P\right)^2 \tag{4.11}$$

with

$$b_i^P = 1 - k_p \frac{\sum_{k=1}^{K_i} |Z_D(\boldsymbol{x}^k) - Z_M(\boldsymbol{x}^k)|}{K_i} \ , \tag{4.12}$$

where k_p controls how high these depth differences should be to enforce a dynamic scoring and λ_P is the parameter that weights this constraint within the overall optimisation. Admittedly, Eq. 4.11 has some degree of redundancy with Eq. 4.5, however, Eq. 4.12 computes depth differences directly without any pre-weighting as do Equations 4.7 and 4.8. This provides additional evidence of the presence of moving objects.

The three terms described above only depend on $\boldsymbol{b}$. For the sake of clarity we group them into the combined term $S(\boldsymbol{b})$ which is used in Equation 4.1:

$$S(\boldsymbol{b}) = S_D(\boldsymbol{b}) + S_R(\boldsymbol{b}) + S_P(\boldsymbol{b}) \ . \tag{4.13}$$

4.2.2.3 Cost Function Overview and Intuition

In this section we provide an overview of the whole cost function with additional explanation regarding how the per-cluster segmentation is solved for.

For a cluster C_i, the cost function will take the form:

$$\min_{\boldsymbol{\xi},b} \left\{ \underbrace{b_i \sum_{p=1}^{K_i} \left[F(w_Z^p r_Z^p(\boldsymbol{\xi})) + F(\alpha_I w_I^p r_I^p(\boldsymbol{\xi})) \right]}_{\substack{D \\ \text{Image alignment}}} + \underbrace{(1-b_i) 2 K_i F(\hat{c})}_{\substack{S_D \\ \text{Segmentation}}} + \underbrace{S_r(b_i) + S_p(b_i)}_{\text{Regularisation}} \right\}, \tag{4.14}$$

where K_i is the cardinality of cluster C_i. The term D contains all per-pixel residuals while the term S_D simulates a high residual for every pixel p in cluster C_i.

When minimising this cost function with respect to the segmentation, b_i will converge towards static (*i.e.* $b_i \to 1$) if $D < S_D$, which means the direct alignment residuals are smaller than those set through the threshold value $\hat{c}$. However, if the residuals of the current cluster are high (*i.e.* $D > S_D$), the segmentation b_i will converge towards dynamic (*i.e.* $b_i \to 0$). Our assumption is that high residual values originate from dynamic objects which move independently from the camera motion.

4.2.2.4 Solver Description

The solver is listed in Algorithm 3 and explained next.

Equation 4.1 involves direct image alignment and to solve it we use a similar coarse-to-fine registration strategy as described in Section 2.2.3.1. Namely, we build a pyramid of images and align these from the coarsest to the finest level.

At each level, the term $D(\boldsymbol{\xi}, \boldsymbol{b})$ is nonlinear and non-convex with respect to $\boldsymbol{\xi}$. However, the combined optimisation problem is convex and can be solved analytically with respect to $\boldsymbol{b}$. This motivates us to estimate $\boldsymbol{\xi}$ and $\boldsymbol{b}$ separately, using iteratively re-weighted least squares (IRLS) to minimise Eq. 4.1 with respect to the camera motion $\boldsymbol{\xi}$ and then obtain the close-form solution for $\boldsymbol{b}$ after every iteration of the IRLS algorithm.

The segmentation computed at every level of the pyramid is used to initialise the solver at the next deeper level, thus allowing the algorithm to converge to the right segmentation at the different levels.

Decoupling $\boldsymbol{\xi}$ from $\boldsymbol{b}$ within the solver allows us to compute the solution for each efficiently, while the tight alternation of those two steps leads to good rate of convergence. In future work, we intend to investigate if simultaneous optimisation of both quantities would lead to a faster rate of convergence.

Algorithm 3: Frame-to-model alignment and motion segmentation

```
Input: $I_D, Z_D, I_M, Z_M$
compute image pyramids for $I_D, Z_D, I_M, Z_M$;
initialise $\boldsymbol{\xi} = \mathbf{0}_{6\times1}$;
initialise $\boldsymbol{b} = \mathbf{0}_{K\times1}$;
foreach level of the pyramid do
    foreach iteration do
        warp $I_D$ towards $I_M$ given current $\boldsymbol{\xi}$;
        warp $Z_D$ towards $Z_M$ given current $\boldsymbol{\xi}$;
        compute segmentation prior $\boldsymbol{b}_P$ (4.12);
        if $\boldsymbol{b}$ not initialised then
            $\boldsymbol{b} = \boldsymbol{b}_P$
        compute weights $w_I, w_Z$ (4.7, 4.8);
        compute visual odometry residuals $\boldsymbol{r}_I$ and $\boldsymbol{r}_Z$ (4.3 ,4.2);
        foreach IRLS iteration do
            compute Cauchy weights $W_I, W_Z$ using segmentation $\boldsymbol{b}$;
            solve for $\Delta\boldsymbol{\xi}$ (4.5);
                $(\boldsymbol{J}_I^\top W_I \boldsymbol{J}_I + \boldsymbol{J}_Z^\top W_Z \boldsymbol{J}_Z)\Delta\boldsymbol{\xi} = -(\boldsymbol{J}_I^\top W_Z \boldsymbol{r}_I + \boldsymbol{J}_Z^\top W_Z \boldsymbol{r}_Z)$;
            update residuals $\boldsymbol{r}_I$ and $\boldsymbol{r}_Z$;
            solve for segmentation $\boldsymbol{b}$ (4.13);
        update $\boldsymbol{\xi}$;
            $\boldsymbol{\xi} = \log(\exp(\boldsymbol{\xi})\exp(\Delta\boldsymbol{\xi}))$;
```

Our main contribution in this chapter regards computing the segmentation, so we describe its analytical solution next.

Solving for Static/Dynamic Segmentation For a particular cluster C_i, the error term Eq. 4.1 with respect to b_i takes the expanded form:

$$\begin{aligned}
E(b_i) = D(\boldsymbol{\xi}, b_i) + S_D(b_i) + S_P(b_i) + S_R(b_i) = \\
= b_i \sum_{p \in C_i} \left[F(w_Z^p r_Z^p(\boldsymbol{\xi})) + F(\alpha_I w_I^p r_I^p(\boldsymbol{\xi})) \right] \\
+ 2(1-b_i) K_i F(\hat{c}) \\
+ \lambda_P (b_i - b_i^P)^2 \\
+ \lambda_R \sum_{j=i+1}^{K} G_{ij} (b_i - b_j)^2 \ .
\end{aligned} \tag{4.15}$$

where $\boldsymbol{\xi}$ is the velocity estimated through IRLS and pixels p are all pixels in cluster C_i. To solve for b_i, we take the derivative of $E(b_i)$ and set this to zero:

$$\frac{\partial E}{\partial b_i} = \sum_{p \in C_i} \Big[F(w_Z^p r_Z^p(\boldsymbol{\xi})) + F(\alpha_I w_I^p r_I^p(\boldsymbol{\xi}))\Big] - 2K_i F(\hat{c}) + 2\lambda_P (b_i - b_i^P) + 2\lambda_R \sum_{j=i+1}^{K} G_{ij}\,(b_i - b_j) = 0\,. \tag{4.16}$$

From Eq. 4.16 we formulate constraints for b_i depending on whether its value is coupled with those of neighbouring clusters:

$$2\lambda_P b_i + \underbrace{\sum_{p \in C_i} \Big[F(w_Z^p r_Z^p(\boldsymbol{\xi})) + F(\alpha_I w_I^p r_I^p(\boldsymbol{\xi}))\Big] - 2K_i F(\hat{c}) - 2\lambda_P b_i^P}_{r_{b_i}} = 0\,, \tag{4.17}$$

$$2\lambda_R \sum_{j=i+1}^{K} G_{ij}\,(b_i - b_j) = 0\,. \tag{4.18}$$

To solve for the full segmentation $\boldsymbol{b}$, we stack these constraints for all clusters in a system of equations of the form:

$$
\begin{array}{r}
K\left\{\vphantom{\begin{array}{c}0\\0\\\vdots\\0\\0\end{array}}\right.\\
n\left\{\vphantom{\begin{array}{c}0\\0\\\vdots\\0\end{array}}\right.
\end{array}
\overbrace{\left[\begin{array}{ccccc}
2\lambda_P & 0 & \cdots & 0 & 0\\
0 & 2\lambda_P & \cdots & 0 & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots\\
0 & 0 & \cdots & 2\lambda_P & 0\\
0 & 0 & \cdots & 0 & 2\lambda_P\\
\hline
2\lambda_R & -2\lambda_R & \cdots & 0 & 0\\
0 & 2\lambda_R & \cdots & -2\lambda_R & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots\\
0 & 0 & \cdots & 2\lambda_R & -2\lambda_R
\end{array}\right]}^{K}
\times
\begin{bmatrix} b_1\\ b_2\\ \vdots\\ b_{K-1}\\ b_K\end{bmatrix}
=
\left[\begin{array}{c}
-r_{b_1}\\ -r_{b_2}\\ \vdots\\ -r_{b_{K-1}}\\ -r_{b_K}\\
\hline
0\\ 0\\ \vdots\\ 0
\end{array}\right]
\begin{array}{l}
\left.\vphantom{\begin{array}{c}0\\0\\\vdots\\0\\0\end{array}}\right\}K\\
\left.\vphantom{\begin{array}{c}0\\0\\\vdots\\0\end{array}}\right\}n
\end{array}
.
$$

Where K is the number of clusters and n is the number of connections, *i.e.* the number of neighbouring clusters as described by the connectivity map G. The first K terms corresponds to all equations of the form of Eq. 4.17 for all clusters, while the following n terms represent the neighbourhood regularisation equations of the form of Eq. 4.18 for all clusters. We solve this system of equations using the Sparse Cholesky decomposition (from the C++ Eigen library [1]) to yield $\boldsymbol{b}$.

[1] http://eigen.tuxfamily.org

4.2.3 Surfel-based 3D Reconstruction

The 3D model is represented as an unordered list of surfels. A surfel is a 3D disk, with associated position and normal $\boldsymbol{p}, \boldsymbol{n} \in \mathbb{R}^3$, colour $\boldsymbol{c} \in \mathbb{N}^3$, radius $r \in \mathbb{R}$, initialisation timestamp t_0, timestamp of latest update t and counter $h \in \mathbb{N}$ representing the number of times the surfel has been updated. We propose an additional characteristic for each surfel which we term *viability*, $w \in [0, 1]$, with $w \to 1$ meaning a surfel has been repeatedly fused with points segmented as static.

At every timestep, our system takes as input the last RGB-D pair (C_D, Z_D) as well as the per-pixel segmentation image B_D of each point belonging to the background. We maintain a fused coloured model for visualisation purposes but convert to intensity when rendering an image prediction.

We follow the original approach of ElasticFusion [54, 53] for pre-processing and data association of new points to existing surfels for 3D fusion. The difference is that we propose a strategy which judges the viability of each surfel in order to enable the model to remove those surfels that are matched with dynamic input points. Our fusion approach is listed in Algorithm 4 and explained below.

4.2.3.1 Surfel Viability

A surfel i is considered *viable* if it is repeatedly observed (h is large) and matched with static input points ($B_D^i > 0.5$). These conditions ensure that a viable surfel is neither spurious nor dynamic. Only in these conditions do we impose that $w \to 1$.

To achieve this, each new surfel is introduced into the model with low viability $w \to 0$. On subsequent observations, w is updated through a running sum of the log-odds probabilities of input points which are considered correspondences.

This strategy is suitable for our application as viability only increases through repeated matches with static points ($B_D^i > 0.5$) and automatically decreases with dynamic matches ($B_D^i < 0.5$):

$$\operatorname{sign}\left(\ln\left(\frac{w}{1-w}\right)\right) = \begin{cases} -1 & \text{if } w < 0.5 \\ 0 & \text{if } w = 0.5 \\ 1 & \text{otherwise} \end{cases} . \tag{4.19}$$

4.2.3.2 Fusion

During fusion, a weighted average scheme is used to update a surfel's position, colour and normal as described in ElasticFusion [54]. In addition to B_D, we employ two additional weights to represent the quality of each input point:

1. For each surfel i with pixel coordinates $\boldsymbol{x}^i \in \Omega$, a Gaussian weight $\gamma_D^i \in [0,1]$ biasing in favour of pixels close to the image centre $\boldsymbol{x}^c$ [53]:

$$\gamma_D^i = \exp\left(-\frac{||\boldsymbol{x}^i - \boldsymbol{x}^c||}{2\sigma_\gamma^2||\boldsymbol{x}^c||}\right) . \tag{4.20}$$

2. A velocity weight $v_D \in [0,1]$ biasing in favour of surfels seen during slow motion:

$$v_D = \max\left(1 - \frac{1 - \max(||\boldsymbol{\xi}||, 0.15)}{0.15}, 0.5\right) . \tag{4.21}$$

While the first term weights surfels based on the assumption that measurements closer to the camera centre are more accurate, the second penalises the influence of surfels recorded during fast motion which would introduce blur within the model.

4.2.3.3 Surfels Removal

Finally, a cleaning stage removes surfels for which $w < 0.5$ for more than 10 consecutive frames. We also perform free-space violation checks to remove points remaining in front of viable surfels. This ensures that dynamic objects and noisy measurements are removed from the map and a clean representation of the environment structure is maintained in the long term.

Algorithm 4: Weighted Surfel Fusion

```
Input: $C_D, Z_D, B_D$
$s_D \leftarrow generate\_input\_surfels()$;
foreach $s_D^i$ do
    $s_M^j \leftarrow search\_for\_model\_correspondence()$;
    if $s_M^j$ found then
        $compute\_weights(\gamma_D^i, v_D)$;
        //Compute input viability
        $w_D^i \leftarrow \min(B_D^i, \gamma_D^i, v_D^i)$;

        //Truncate to avoid early saturation
        $w_M^j \leftarrow \max(0.01, \min(0.99, w_M^j))$;
        $w_D^i \leftarrow \max(0.01, \min(0.53, w_D^i))$;

        //Update position, colour and normal through weighted average
         scheme
        $\boldsymbol{k}_M^j \leftarrow \frac{h_M^j w_M^j \boldsymbol{k}_M^j + w_D^i \boldsymbol{k}_D^i}{h_M^j w_M^j + w_D^i}$;
        $\forall \boldsymbol{k} \in \{\boldsymbol{p}, \boldsymbol{c}, \boldsymbol{n}\}$;

        //Update viability by sum of log-odds
        $l \leftarrow \ln\left(\frac{w_M^j}{1 - w_M^j}\right) + \ln\left(\frac{w_D^i}{1 - w_D^i}\right)$;
        $w_M^j \leftarrow 1 - \frac{1}{1 + \exp(l)}$;

        //Update history counter
        $h_M^j \leftarrow h_M^j + 1$;
    else
        //Add new surfels
        $w_D^i \leftarrow \alpha$ where $\alpha \rightarrow 0$;
        $h_D^i \leftarrow 1$;
        $add\_surfel\_to\_model()$;
```

4.2.4 Implementation Details

4.2.4.1 Initialisation

As we rely on a map for both odometry and static/dynamic segmentation of the scene, we require an initialisation stream to generate the first reliable map. The initial frames (first 1-2 seconds) observed by the system should contain no more than 20-30% of moving elements in order to allow for a successful initialisation of the map.

Our current formulation starts by aligning the first two RGB-D pairs read from the camera following the same procedure described in Section 4.2.2. By solving Eq. 4.1 we also obtain a segmentation of the last image (B_D) that we use to generate the first instance of the map. After this first step the algorithm always aligns the incoming RGB-D pairs with the last prediction obtained from the map.

4.2.4.2 Analysing and processing residuals

Among the parameters presented in Section 4.2.2, the most important ones are c and $\hat{c}$ (see Fig. 4.2). The parameter c is commonly chosen as a linear function of the median or the median absolute deviation (MAD) of the residuals [82, 114]. Since this metric is computationally expensive, we sacrifice accuracy and compute the mean ($\bar{r}$) of the residuals instead:

$$\bar{r} = \frac{1}{2N} \sum_{p=1}^{N} |w_Z^p r_Z^p(\boldsymbol{\xi})| + |\alpha_I w_I^p r_I^p(\boldsymbol{\xi})| \ . \tag{4.22}$$

Note that $\bar{r}$ actually represents the mean of the pre-weighted residuals. This computation is performed before each iteration of the IRLS solver described in Section 4.2.2.4 (for the first iteration $\boldsymbol{\xi}$ is assumed to be null). Afterwards, to ensure robust estimation, we set $c = 0.5\,\bar{r}$.

For $\hat{c}$ we choose a threshold above which points are regarded dynamic, considering that false positives (static regions segmented as dynamic) are preferable to false negatives (dynamics are segmented as static). The lower the value of $\hat{c}$, the more aggressively are dynamics segmented. During initialisation this value is set lower to ensure that the initial map contains only static parts:

$$\hat{c} = \begin{cases} \max(r_{min}, \bar{r}) & \text{During initialisation} \\ 1.5 \max(r_{min}, \bar{r}) & \text{Otherwise} \end{cases} \ . \tag{4.23}$$

The variable r_{min} sets the minimum residual value below which clusters should always be segmented as static. When images are perfectly aligned the mean residual $\bar{r}$ is very low and the threshold $1.5\ \bar{r}$ would also be very low, which would lead to having static clusters segmented as being dynamic — irrespective of how precise the alignment is. The introduction of r_{min} solves this problem.

The values of the other parameters are as follows:

- K, the total number of clusters, is 24.
- α_I, the weighting applied to the photometric term (Eq. 4.5), is 0.15.
- The weights of the regularisation terms S_R (Eq. 4.10) and S_P (Eq. 4.11), namely λ_R and λ_P, are set to 0.35 and 0.5 respectively.
- The pre-weighting constants, k^I (Eq. 4.8) and k^Z (Eq. 4.7) are set to 0.1 and 10^{-4} respectively.

4.3 Evaluation

We evaluate our approach in static and dynamic environments. First, results are presented for several sequences of the Freiburg dataset [26], which has ground truth camera trajectory from a motion capture system and allows us to measure relative and absolute drift. This dataset contains only a small number of sequences with high dynamics and, for that reason, we recorded by ourselves additional sequences with a hand-held camera to provide a more general evaluation with varied scenes.

We compare the accuracy of our method, StaticFusion (SF), against related state-of-the art approaches:

1. The joint visual odometry and scene flow of Jaimez *et al.* [79] (VO-SF). As an odometry method designed for dynamic scenes, a comparison is useful to investigate the benefits of reconstructing a 3D model of the static scene.
2. ElasticFusion [54] (EF) in order to investigate the performance of a state-of-the-art method designed for static environments within dynamic scenes.
3. Co-Fusion [108] (CF), as it is a state-of-the-art approach for tracking and reconstructing multiple moving objects.

4. The background model-based visual odometry of Kim *et al*. [107] (BaMVO), which is conceptually related to our approach but uses a multi-frame strategy instead of frame-to-model alignment. Since we were unable to replicate the published results using the open-source release of this method, we only include numerical results for the sequences evaluated in the original publication.

The experiments were performed on a workstation with an Intel(R) Core(TM) i7-3770 CPU at 3.40GHz and a GeForce GTX 1070 GPU using the Ubuntu 16.04 operating system. We used registered RGB-D images and maintained default parameters for the methods we compare against.

Regarding the image resolution, to ensure real-time operation we use QVGA (320 $\times$ 240) and keep the default resolution of the other methods included in the comparison (VO-SF, BaMVO also use QVGA, the remaining ones work with VGA).

Because we are mainly focused on evaluating the robustness of the visual odometry component of StaticFusion, loop closure capabilities are disabled during these experiments.

4.3.1 Quantitative Evaluation

Regarding the accuracy of pose estimation, evaluation is performed through the metrics proposed by Sturm *et al*. [26]:

- Translational and rotational relative pose error (RPE) to measure the average local drift per second.
- Translational absolute trajectory error (ATE) to measure the global quality of the trajectory.

For completeness, sequences recorded in static, low dynamic and high dynamic scenes are included within this evaluation.

4.3.1.1 Trajectory Evaluation and Comparison Against Related State-of-the-Art

Results are listed in Tables 4.1, 4.2 for RPE and in Table 4.3 for ATE.

It can be seen that StaticFusion's performance in *static environments* is comparable to that of ElasticFusion for both ATE and RPE criteria. Table 4.3 demonstrates the advantage of frame-to-model alignment strategies over odometry methods in reducing overall drift.

Environment	Sequence	Trans. RPE RMSE (cm/s) VO-SF	EF	CF	BaMVO	SF
Static Env.	fr1/xyz	2.1	**1.9**	2.3	–	2.4
	fr1/desk	3.7	**2.9**	9.0	–	3.0
	fr1/desk2	5.4	7.2	9.2	–	**4.79**
	fr1/plant	6.1	**5.0**	8.9	–	9.14
Low Dynamic Env.	fr3/sit_static	2.4	**0.9**	1.1	2.4	1.1
	fr3/sit_xyz	5.7	**1.6**	2.7	4.8	2.81
	fr3/sit_halfsphere	7.5	17.2	**3.0**	5.8	3.17
High Dynamic Env.	fr3/walk_static	10.1	26.0	22.4	13.3	**1.3**
	fr3/walk_xyz	27.7	24.0	32.9	23.2	**9.25**
	fr3/walk_halfsphere*	24.8	16.3	31.1	–	**5.5**
	fr3/walk_halfsphere	33.5	20.5	40.0	**17.3**	20.7

Table 4.1: Relative Translation Error.

Environment	Sequence	Rot. RPE RMSE (deg/s) VO-SF	EF	CF	BaMVO	SF
Static Env.	fr1/xyz	1.00	**0.91**	1.34	–	1.43
	fr1/desk	1.77	**1.48**	4.49	–	2.17
	fr1/desk2	**2.45**	4.07	4.79	–	3.28
	fr1/plant	2.00	**1.58**	3.02	–	2.92
Low Dynamic Env.	fr3/sit_static	0.71	**0.30**	0.44	0.69	0.42
	fr3/sit_xyz	1.44	**0.59**	1.00	1.38	0.90
	fr3/sit_halfsphere	2.98	4.56	**1.92**	2.88	2.07
High Dynamic Env.	fr3/walk_static	1.68	4.77	4.01	2.08	**0.39**
	fr3/walk_xyz	5.11	4.79	5.55	4.39	**2.26**
	fr3/walk_halfsphere*	5.49	5.70	8.45	–	**2.17**
	fr3/walk_halfsphere	6.69	6.41	13.02	**4.28**	5.04

Table 4.2: Relative Rotation Error.

In highly *dynamic sequences*, our system outperforms the other approaches for the following reasons:

- EF is not designed to handle dynamics in the scene. Thus, moving objects corrupt its 3D reconstruction and, in turn, its pose estimate.
- VO-SF and BaMVO are odometry-based methods which cannot handle sequences where more than 50% of the image contains dynamics
- CF tracks and reconstructs rigid objects, and its performance deteriorates when non-rigid elements such as people are present in the scene.

As mentioned in Section 4.2.4, StaticFusion requires sequences with limited dynamics during the initialisation stage of the model and for this reason it produces high

Environment	Sequence	Trans. ATE RMSE (cm)			
		VO-SF	EF	CF	SF
Static Env.	fr1/xyz	5.1	**1.2**	1.4	1.5
	fr1/desk	5.6	**2.1**	17.7	2.4
	fr1/desk2	17.4	5.7	16.8	**4.1**
	fr1/plant	7.8	**5.3**	12.6	9.0
Low Dynamic Env.	fr3/sit_static	2.9	**0.8**	1.1	1.3
	fr3/sit_xyz	11.1	**2.2**	2.7	3.9
	fr3/sit_halfsphere	18.0	42.8	**3.6**	4.0
High Dynamic Env.	fr3/walk_static	32.7	29.3	55.1	**1.5**
	fr3/walk_xyz	87.4	90.6	69.6	**9.3**
	fr3/walk_half*	48.2	48.6	75.6	**5.8**
	fr3/walk_half	73.9	63.8	80.3	**39.1**

Table 4.3: Absolute Translation Error.

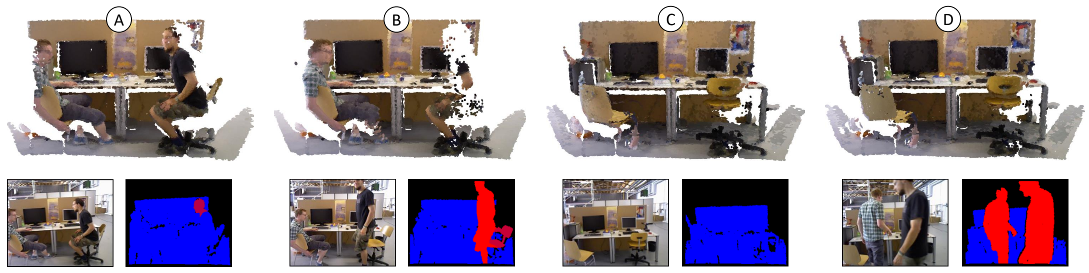


Figure 4.3: **Top**: The background reconstruction evolves to reflect changes in the scene. **Bottom**: Corresponding RGB image (left) and computed segmentation (right) for each instance, where red means dynamic and blue means static. Between (A) and (C) the model adapts to remove dynamic objects based on a correct segmentation. The sequence at the instance of (D) contains significant dynamics and the model provides a valuable prior knowledge of the scene, allowing for correct segmentation and pose estimation.

errors on *fr3/walking_halfsphere*. We include the additional sequence *fr3/walking_halfsphere**, which skips the initial 5 seconds which have high dynamics, in order to illustrate the ability of the method to perform well for that same scene when the initial frames are not so challenging (note that the other methods still fail in this case).

In order to illustrate how our background model handles moving objects, Fig. 4.3 shows the temporal evolution of the map built during the sequence *fri3/walking_static*. It can be seen that the map evolves to reflect changes in the scene. It efficiently removes moving parts, keeping only the structure which represents a valuable prior for accurate segmentation of dynamics.

4.3.1.2 Importance of Regularisation Terms

In this section we examine the importance of the two regularisation terms described in Section 4.2.2.2, namely S_R, which encourages neighbouring clusters to have a similar score, and S_P, which is a segmentation prior. Table 4.4 reports results regarding ATE and RPE of the performance of the system with and without these priors.

In static scenes including the priors improves accuracy by a small extent as they enable the algorithm to compute accurate estimates of the motion segmentation. Their contribution becomes more evident in dynamic scenes where they significantly improve the robustness of the method.

Sequence	Trans. ATE RMSE (cm)		Trans. RPE RMSE (cm/s)		Rot. RPE RMSE (deg/s)	
	None	$S_P + S_R$	None	$S_P + S_R$	None	$S_P + S_R$
fr1/xyz	**1.5**	**1.5**	**2.4**	**2.4**	1.50	**1.43**
fr1/desk	2.9	**2.4**	3.5	**3.0**	2.85	**2.17**
fr1/desk2	6.9	**4.1**	7.8	**4.79**	4.70	**3.28**
fr1/plant	9.6	**9.0**	10.0	**9.14**	3.42	**2.92**
fr3/sit_static	1.4	**1.3**	1.2	**1.1**	4.40	**0.42**
fr3/sit_xyz	4.5	**3.9**	3.1	**2.81**	0.94	**0.90**
fr3/sit_halfsphere	6.2	**4.0**	5.1	**3.17**	2.44	**2.07**
fr3/walk_static	12.08	**1.5**	14.54	**1.3**	2.63	**0.39**
fr3/walk_xyz	13.2	**9.3**	12.4	**9.25**	3.05	**2.26**
fr3/walk_half*	11.7	**5.8**	16.1	**5.50**	4.86	**2.17**

Table 4.4: Absolute and Relative Translational and Rotational Errors for our approach with and without the inclusion of priors. The priors aid the method with regards to accuracy and robustness in all cases.

4.3.1.3 Importance of Image Resolution

In this experiment we analyse the performance and run-time of the algorithm on decreasing image resolutions. The maximum resolution of the image pyramid is set in decreasing order to 640×480, 320×240, 160×120 and 80×60. The lowest level of the image pyramid has a resolution of 20×15 in each case, therefore the number of layers in the pyramid are: 6, 5, 4 and 3 respectively.

Table 4.5 summarises the ATE results obtained using the different image resolutions and also lists their respective run-times. While the best results are achieved with the highest resolution, it is also worth noting that decreasing the resolution from 640×480 to 320×240 leads to only marginally worse accuracy results while the run-time is reduced to approximately a quarter.

Environment	Sequence	Trans. ATE RMSE (cm) 640 × 480	320 × 240	160 × 120	80 × 60
Static Env.	fr1/xyz	**1.4**	1.5	1.5	1.5
	fr1/desk	2.5	**2.4**	2.5	3.5
	fr1/desk2	4.5	**4.1**	4.2	5.0
	fr1/plant	**8.2**	9.0	11.6	17.9
Low Dynamic Env.	fr3/sit_static	**1.1**	1.3	1.5	2.0
	fr3/sit_xyz	**3.6**	3.9	4.5	6.3
	fr3/sit_halfsphere	**3.9**	4.0	4.1	9.2
High Dynamic Env.	fr3/walk_static	**1.3**	1.5	2.3	4.7
	fr3/walk_xyz	10.0	**9.3**	11.5	139
	fr3/walk_half*	**5.5**	5.8	6.4	10.1
Runtime (ms/frame)		116	30	13	8

Table 4.5: Absolute Translation Error on Decreasing Image Resolution.

4.3.2 Qualitative Evaluation (3D Maps)

In this section we evaluate our approach in scenes with varying levels of dynamism. To this end, we have recorded two sequences with a hand-held RGB-D camera. In the first sequence the camera observes a person interacting with objects and performing varied motions at different speeds. Occasionally the person remains still and close to the camera, which poses a challenge to the tracking system because it must be able to segment out the person as soon as they move in order not to lose track of the sensor. For the second test we have recorded a 'selfie sequence' during which a person carries the camera while it points at them. This is a very complex test because the person often occupies more than 50% of the image and looks quasi-static with respect to the camera (which represents a strong local minimum for the motion estimate of any method which does not segment moving parts explicitly). We encourage the reader to inspect the video in Appendix A.3.2 in order to see these experiments.

Results for the first sequence are shown in Fig. 4.4. It can be seen that the person is initially inserted in the map because he does not move, but is removed as soon as he moves. The segmentations estimated by StaticFusion are almost perfect during the whole sequence: the person is always segmented correctly, as well as the ball and the door when it gets closed.

The second sequence starts and finishes with the camera at the exact same position (see Fig. 4.5), which allows us to measure the overall drift for the full trajectory. It can be observed that our algorithm is able to provide good predictions (C_M, Z_M) and accurate camera pose estimates even when the person covers most of the view of the

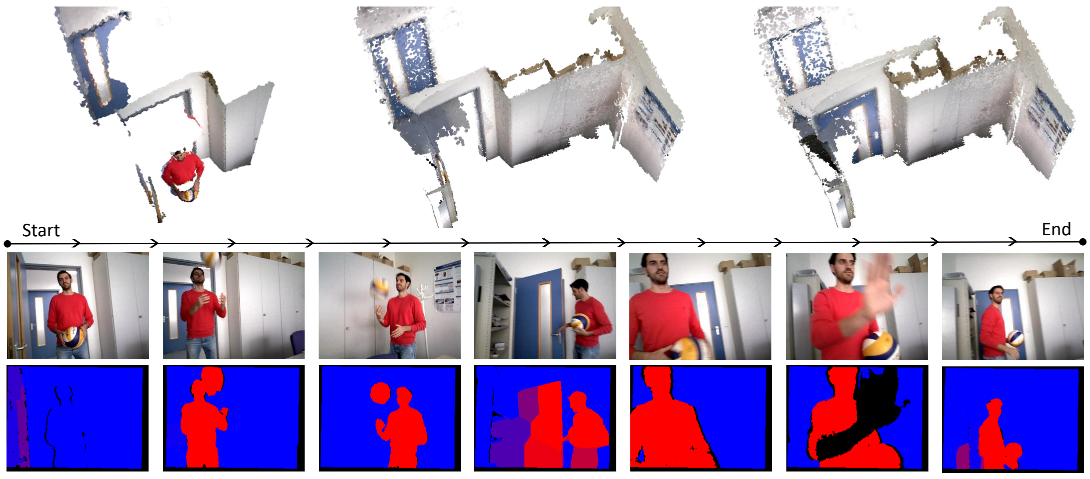


Figure 4.4: **Top**: Evolution of the map built during the first sequence of Section 4.3.2. **Bottom**: Some of the sample input images and the segmentations that StaticFusion creates for them.

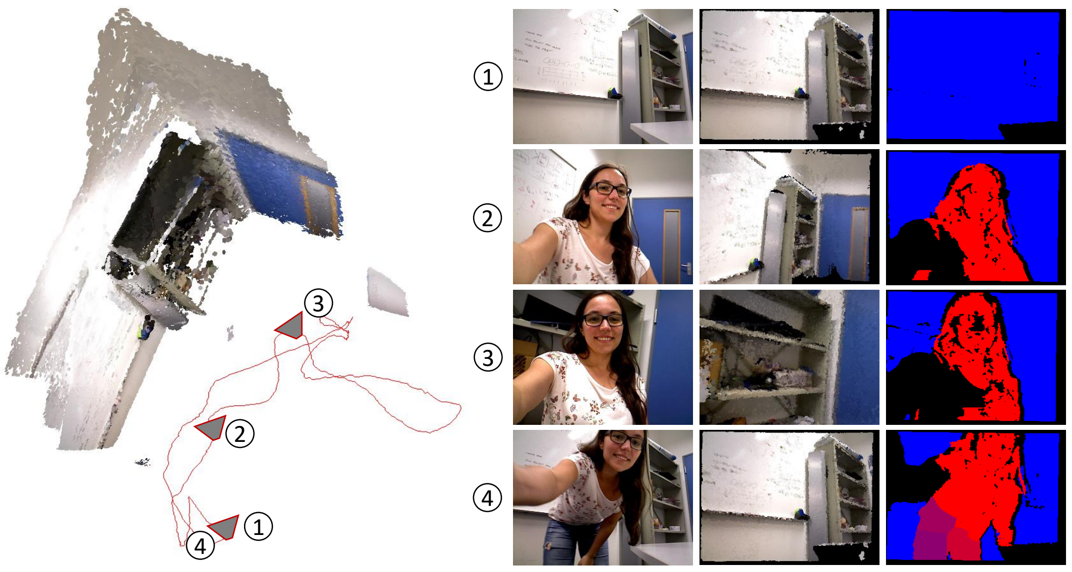


Figure 4.5: **Left:** trajectory estimated by StaticFusion for the second experiment in Section 4.3.2. **Right:** Some of input images of this sequence (first column), together with our predictions for those same views (second column) and the estimated segmentations (third column).

camera.

ElasticFusion [54] and Co-Fusion [108] were also tested with these sequences, and they both fail to provide good pose estimates. Figure 4.6 illustrates some of these failures. Figure 4.6 (A) shows the map reconstructed by ElasticFusion for the first sequence before it loses track of the camera (note the double chair and the misaligned walls). Figure 4.6 (B) shows how Co-Fusion finds too many moving objects during the second sequence and randomly keeps track of those parts of the scene, destroying the overall map. For that sequence, with an estimated trajectory length of 9.5 metres, the

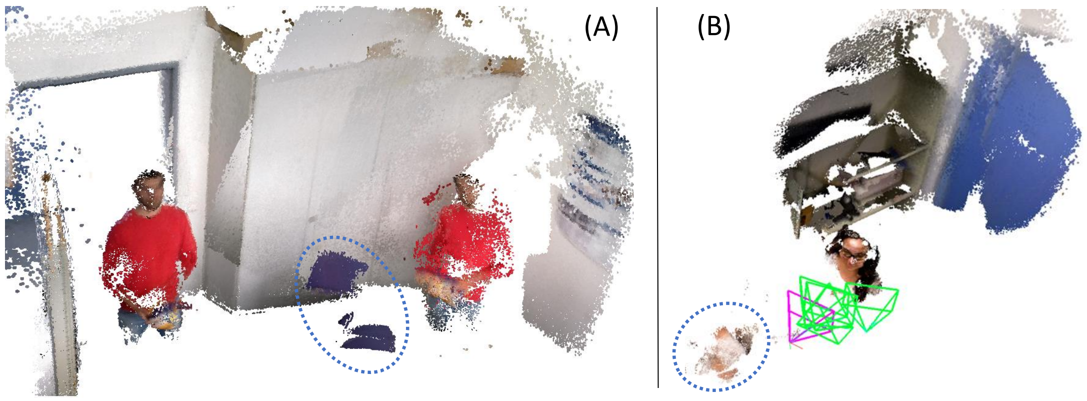


Figure 4.6: (A) shows how ElasticFusion adds the person into the map multiple times and eventually fails to estimate the camera motion. In (B) Co-Fusion finds and tracks multiple independent phantom objects (note that the region surrounded by the blue circle is part of the person as well).

final drift of our method is just 1.5 centimetres, whereas the total drift for ElasticFusion and Co-Fusion was 1.03 metres and 0.88 metres respectively.

4.3.3 Failure Cases

In this section we discuss some of the problems and failure cases of our method.

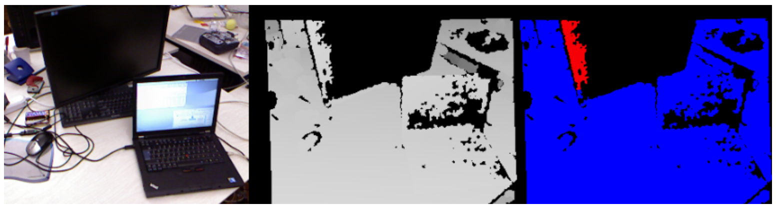
4.3.3.1 Segmentation Estimation

Motion segmentation is computed based on residuals, with the intuition that data which agrees with the model is likely part of the static background, while points which do not have a model correspondence are part of moving objects.

This design makes our approach sensitive to the quality of the sensor and the data it provides.

Noisy data from specular objects or objects which are far from the sensor may generate high residuals and thus be classified as dynamic even though they are not moving. Figure 4.7 shows examples of false-positive segmentation estimates, where flickering in the depth estimation due to the monitor (4.7a) and from an object located at a distance from the camera (4.7b) are segmented as dynamic.

For pose estimation this may not be a severe limitation as we would only want to use the data of best quality when estimating the motion the sensor. However, the 3D reconstruction will be effected and may contain holes in the places where the data is noisy.


(a) The monitor is incorrectly segmented as dynamic because depth estimation on the black matte monitor is flickery, which results in large residuals.

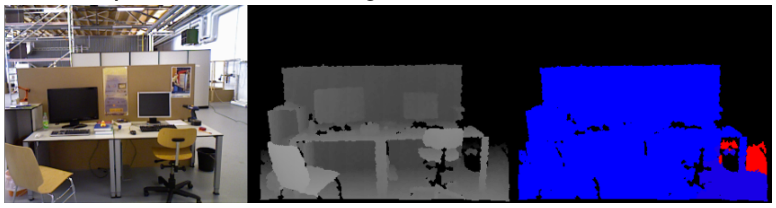

(b) The floor in the bottom right area of the image is incorrectly segmented as dynamic because it generates large residuals as it is far away from the camera.

Figure 4.7: Examples of motion segmentation false-positive detections. RGB image (left), depth image (centre), computed segmentation (right), where blue means static and red means dynamic.

This design also means that, for moving objects to be detected, their residuals must be larger than those corresponding to data of average quality. Thus, small objects may not be detected as dynamic as their residuals may be small and comparable to those of points considered inliers.

4.3.3.2 3D Reconstruction

The fusion technique removes existing surfels from the 3D model if it is possible to see behind this surface in the current sensor data. However, this strategy does not result in a fully clean reconstruction, as some surfels may remain in places where no structure is sensed behind, for example when the background is too far away to be detected by the sensor. This results in reconstructions which contain some ”flying points” and an example of this is shown in Figure 4.8.

A solution to this problem could be to compute connected components of the map and remove those which are small and isolated. Connected component analysis however is computationally expensive and it may be necessary to run this procedure on a separate thread. Thus, the main thread would continuously perform 3D fusion while the secondary thread would occasionally process the map and clean it using connected components analysis.

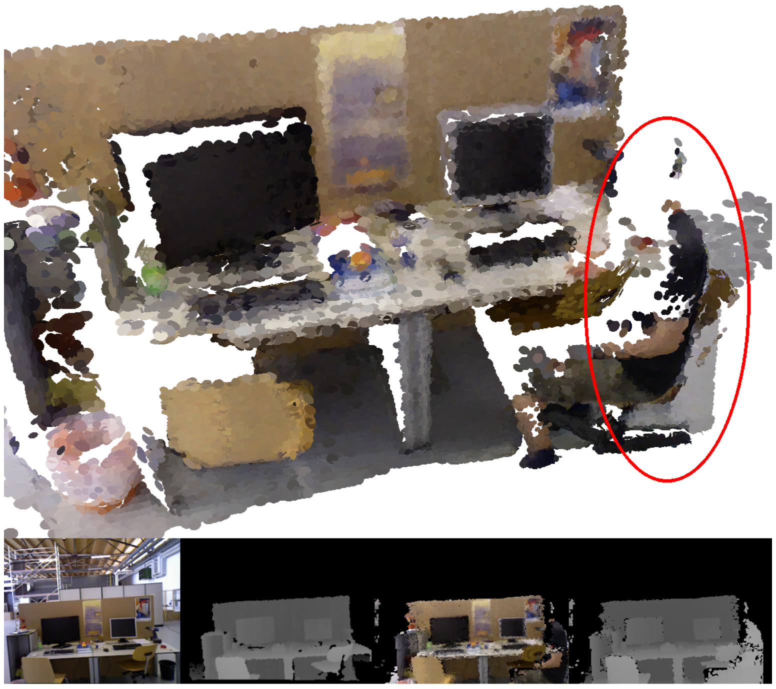


Figure 4.8: **Top**: An example of a reconstruction which still contains a person even after they have moved (encircled in red). **Bottom left**: current RGB-D image pair. **Bottom right**: RGB-D model prediction image pair. The person is not removed from the reconstruction because the current depth image does not contain data in the area where they were sitting.

4.4 Discussion

In this chapter we proposed an approach for RGB-D SLAM in dynamic environments. It is based on building and exploiting a 3D background model that only contains static parts of the scene and using this to accurately estimate the motion of the RGB-D camera in the presence of moving objects.

We evaluated our approach on a standardised dataset including sequences with several moving objects and compared it against state-of-the-art techniques. Our qualitative and quantitative results demonstrate that it outperforms related approaches and it achieves a competitive runtime of 30 milliseconds/frame.

Since publication, this approach has been considered state-of-the-art and used as a baseline for robust RGB-D visual odometry in dynamic environments in several new research works [110, 115, 116, 117].

While our approach is robust to high levels of dynamics if a confident model of the environment is in place, it can fail if the initial stream of images contains more than $\sim 30\%$ of moving elements. Also, the approach struggles if significant portions of the estimated static scene begin moving ($> 50\%$ of the image).

The method could be made more robust through several approaches:

- Implementation of multi-frame optimisation to ensure that the background and moving objects are consistently detected within consecutive frames.
- Sensor fusion using a motion prior as proposed in the previous chapter is also a reasonable extension direction. The motion prior would improve robustness of camera pose estimation and aid in the computation of motion segmentation.
- Related approaches leverage progress in Deep Learning techniques for semantic segmentation and object classification [110, 115]. These approaches track and reconstruct multiple classes of objects and use semantics to reason about which ones are dynamic. Whilst our approach attempts to be as general as possible by only reasoning about motion, further robustness can be achieved by fusing semantic information within the motion segmentation procedure.

We explore the second option in the next chapter by combining this visual SLAM system with an inertial sensor in order to analyse the importance of both techniques for handling dynamic environments.

Chapter 5

Dense RGB-D SLAM in Dynamic Environments using Inertial Sensing and Motion Segmentation

As discussed in previous chapters, moving objects are challenging for visual SLAM because they can cause failures in camera pose estimation or be wrongly fused into the map.

Typically, this problem is handled either by integrating a motion prior from a separate sensor within visual odometry to enable the system to ignore these objects or by segmenting moving objects (using semantics or motion) and removing them before visual odometry.

This has also been the approach of this thesis, as in Chapter 3 we used a kinematic-inertial motion prior to aid with robustness, while in Chapter 4 we implemented residual-based motion segmentation to eliminate moving objects during visual odometry.

In this chapter we aim to further examine and evaluate the performance of StaticFusion (the method introduced in the previous chapter) by extending it with rotational prior information from a gyroscope. Through this extension we study and compare the use of a motion prior against motion segmentation, in order to understand their contributions and weaknesses for handling dynamic scenes.

We start from the simple baseline of frame-to-model alignment where all the pixels in the image are used to compute the camera motion. We then formulate a simple extension to integrate a motion prior from a gyroscope within the visual odometry process, without otherwise modifying it. Finally, we include the motion segmentation formulation proposed in the previous chapter and use it to filter out the pixels which belong to moving objects.

We evaluated all these configurations separately using a dataset which we collected with the Intel RealSense ZR300, a sensor that contains an RGB-D camera and an IMU, in a Vicon motion capture arena used to collect ground truth trajectory estimates. We record sequences in increasingly challenging scenarios, starting from completely static scenes, to scenes with small moving objects covering 20-30% of the field-of-view and ending with scenes where moving objects temporarily occupy $> 50\%$ of the field-of-view and also transition between being in motion and being static.

We find that using both a motion prior and motion segmentation improve performance and reliability. The rotational prior provides increased robustness against the baseline vision-only configuration and good performance in scenes where moving objects occupy a small portion of the image. Its influence is diminished if the majority of the image contains moving objects. Here, motion segmentation is necessary to detect and remove dynamic objects, thus reducing drift during visual odometry.

5.1 Handling Dynamic Environments in Dense RGB-D SLAM

As mentioned, there are two main strategies to tackle the presence of moving objects in visual SLAM:

- Integrating a motion prior from an external sensor. Motion priors are used in visual SLAM for handling fast motions and textureless scenes, but their performance in dynamic environments is not well understood. In particular, moving objects induce *motion conflict* in sensor fusion systems: visual odometry can produce very different and potentially contradictory motion estimates compared to proprioceptive sensors in this case, rendering fusion difficult or less effective than in static environments.

- Computing image segmentation of the camera feed to remove dynamic objects before visual odometry, through techniques based on motion or semantic detection.

Next, we discuss approaches for both strategies.

5.1.1 Motion Prior Integration

Visual-inertial odometry and SLAM systems have been researched significantly in recent years for several reasons: cameras and IMUs are now available at low cost and the combined sensors are lightweight and of small size which makes them applicable to small mobile platforms [118].

Visual-inertial approaches display a trade-off between resource requirements and accuracy. High accuracy is achieved with batch optimisation approaches which process all image frames and IMU measurements simultaneously to produce compelling results in offline implementations [119, 120]. At the other extreme, filtering approaches are very computationally efficient but accumulate more error as they only process the most recent sensor measurements [121, 122]. Fix-lag smoothing methods sit in between. They maintain a short history of images and IMU measurements and perform estimation using these [123, 124, 69].

The IMU has several limitations as a motion prior source. While gyroscopes can be used to compute an estimate of the inter-frame rotation of the camera, estimating linear velocity and translation from the accelerometer is non-trivial due to the complex coupling of bias, noise and gravity. Translational estimates computed purely from inertial data can be unreliable for any duration longer than 1-2 seconds. Laidlow *et al*. [69] report that in some cases more accurate results can be obtained with vision-gyroscope fusion rather than full inertial integration (Section VI.B.3 of their paper).

For this reason and also because we use the same sensor as Laidlow *et al*. (Intel RealSense ZR300), we have omitted accelerometer fusion in the system developed in this chapter.

On robotic platforms, wheel or leg odometry provide more reliable estimates of translational motion, as this is sensed directly [125]. Unlike accelerometry fusion, the odometry from wheels or legs can differentiate between a stationary robot and a robot moving at constant velocity. However, translational estimates can be incorrect when slippage or dynamic foot impacts occur.

Approaches fusing RGB-D vision and motion priors have not been evaluated in dynamic environments and we are interested to understand if sensor fusion can provide sufficient robustness in this case.

5.1.2 Image Segmentation

This approach detects dynamic objects in images and segments them such that only background pixels are used to compute the camera motion.

Segmentation can be done either by detecting the motion of different objects or by semantic labelling. In the case of semantics, segmentation is mostly limited to people [110] or cars in traffic environments [126]. This is effective in eliminating the main potential causes of motion in a scene, but is often insufficient. For example, in indoor environments, many types of objects, such as furniture or household objects, can be moved.

Motion-based segmentation is a more general approach, but is difficult to compute when the camera is also moving. During visual odometry, it is impossible to compute segmentation if more than 50% of the current image contains moving objects [79].

Once detected, moving objects can be removed from the visual odometry procedure and either discarded [127] or reconstructed and tracked [108]. The reconstruction of dynamic objects is challenging as they are non-rigid and one must estimate deformation fields to track their motion [112, 113].

To avoid complexity and maintain a real-time operation, we treat dynamic objects as noise and do not track or reconstruct them.

5.2 System Overview

We formulate our approach as an extension of the system described in Chapter 4, StaticFusion. For simplicity, we maintain the same notation as used in that chapter.

To reiterate, StaticFusion is a real-time dense RGB-D SLAM system with a two step architecture, alternating between tracking and mapping. During tracking we simultaneously estimate the camera motion $\boldsymbol{\xi}$ and a static/dynamic segmentation of the current input image $\boldsymbol{b}$, in a combined energy function which takes the form:

$$E(\boldsymbol{\xi}, \boldsymbol{b}) = D(\boldsymbol{\xi}, \boldsymbol{b}) + S(\boldsymbol{b}) \; . \tag{5.1}$$

The mapping component builds a dense surfel-based 3D model of the environment, based on [54].

We extend our system with additional information from a gyroscope in order to analyse its importance in dynamic scenes. As previously mentioned, accelerometry is omitted.

5.2.1 Computation of Rotational Prior from Gyroscope

We refer to ${}^{(est)}\boldsymbol{T}_{A_{t_i} \to B_{t_j}}$ as the transformation measured by the estimator est of frame B at time t_j relative to frame A at time t_i. The transformation $\boldsymbol{T} \in \mathbb{SE}(3)$ is composed of a rotation matrix $\boldsymbol{R} \in \mathbb{SO}(3)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$. We make use of the gyroscope estimator g with coordinate frame G and the visual odometry estimator vo with camera coordinate frame C.

5.2.1.1 Incremental Rotation Estimation

The gyroscope produces estimates at a much higher rate compared to the camera (approximately 200Hz, as opposed to 30Hz for the camera). In order to compute a rotational prior to be used in visual odometry, we must cumulate all M gyroscope measurements sensed between two consecutive images recorded at time instances t and $t+1$:

$$ {}^{(g)}\boldsymbol{R}_{G_t \to G_{t+1}} = \prod_{i=1}^{M} \exp((\boldsymbol{\omega}_i - \boldsymbol{g})\Delta t_i) \ , \tag{5.2} $$

where $\boldsymbol{\omega}_i \in \mathbb{R}^3$ is an individual angular velocity estimate, $\boldsymbol{g} \in \mathbb{R}^3$ is the gyroscope bias that we estimate at the start of the experiment and Δt_i is the amount of time between consecutive gyroscope measurements.

This rotational increment is estimated with respect to the gyroscope body frame, G, however we make use of it during visual odometry and must convert it to the camera coordinate frame, C. We perform offline extrinsic calibration of the RGB camera and the IMU using the Kalibr system [73] to estimate the pose of the camera with respect to the gyroscope, $\boldsymbol{T}_{G \to C}$, which is used to convert all rotational estimates to the camera coordinate frame:

$$ {}^{(g)}\boldsymbol{R}_{C_t \to C_{t+1}} = \boldsymbol{R}_{C \to G}\, {}^{(g)}\boldsymbol{R}_{G_t \to G_{t+1}} \boldsymbol{R}_{G \to C} \ . \tag{5.3} $$

5.2.1.2 Initial Gyroscope Bias Estimation

We estimate the gyroscope bias $\boldsymbol{g} \in \mathbb{R}^3$ at the start of the experiment by assuming a number of known visual odometry estimates. We start each experiment by facing the sensor towards a static scene that contains sufficient 3D information to constrain visual odometry and rotate it on all 3 axes. We compute a series of rotational visual odometry increments, ${}^{(vo)}\boldsymbol{R}_{C_t \to C_{t+1}}$, and convert them to the gyroscope coordinate frame, ${}^{(vo)}\boldsymbol{R}_{G_t \to G_{t+1}}$, using the same technique as described above (Eq. 5.3).

We formulate one gyroscope bias $\boldsymbol{g}$ residual for each pair of corresponding vision and gyroscope motion estimates, where Δt is the amount of time during which these motions were estimated:

$$\boldsymbol{r}_{bias}(\boldsymbol{g}) = \log({}^{(g)}\boldsymbol{R}_{G_t \to G_{t+1}} {}^{(vo)}\boldsymbol{R}^{-1}_{G_t \to G_{t+1}})/\Delta t - \boldsymbol{g} \ . \tag{5.4}$$

One pair of corresponding estimates is sufficient to solve for the value of the bias vector, however, we make use of multiple such pairs in order to reduce the influence of outliers:

$$E_{bias}(\boldsymbol{g}) = \sum_{i=1}^{N} \boldsymbol{r}^i_{bias}(\boldsymbol{g})^T \boldsymbol{r}^i_{bias}(\boldsymbol{g}) \ . \tag{5.5}$$

E_{bias} is quadratic and we solve it analytically to yield the gyroscope bias $\boldsymbol{g}$ which, for simplicity, is maintained constant for the duration of the experiment. The accuracy of our proposed system could be further improved through the implementation of online gyroscope bias estimation, as described in the systems of Forster *et al.* [120] and Mur-Artal *et al.* [68].

5.2.1.3 Rotational Prior Formulation

The ${}^{(g)}\boldsymbol{R}_{C_t \to C_{t+1}}$ quantity (estimated through Eq. 5.2 and Eq. 5.3) is used to formulate a residual prior for the rotation component of the current incremental motion estimate:

$$\boldsymbol{r}_g(\boldsymbol{\xi}) = \log({}^{(g)}\boldsymbol{R}_{C_t \to C_{t+1}} \exp(\boldsymbol{\xi}_{rot})^{-1}) \ , \tag{5.6}$$

where $\boldsymbol{\xi}_{rot}$ refers to the rotational component of the current pose increment.

We now formulate an energy term, G, for this residual, to be integrated within the original formulation (Eq. 5.1).

As opposed to dense image alignment, which computes one residual for each valid pixel, the gyroscope provides a single residual in total. We must allocate a weight to this term to enable it to contribute during optimisation.

As mentioned, we evaluate the relative importance of motion prior integration and motion segmentation separately, regarding robust camera pose estimation in dynamic scenes. In order to assess both scenarios, we compute two forms for the energy term G, depending on the configuration of the system.

First, we describe our naive baseline configuration for dense visual odometry and gyroscope integration. Second, we describe our strategy for combining gyroscope and motion segmentation to explicitly tackle motion conflict.

5.2.2 Visual Odometry and Rotational Prior Integration

Baseline Configuration: Pure Gyroscope Integration When information about motion segmentation is not available, the gyroscope term is weighted by the inverse covariance of the rotational increment:

$$G(\boldsymbol{\xi}) = \boldsymbol{r}_g(\boldsymbol{\xi})^T \boldsymbol{\Sigma}_g^{-1} \boldsymbol{r}_g(\boldsymbol{\xi}) \ , \tag{5.7}$$

where the covariance ${}_i\boldsymbol{\Sigma}_g$ corresponding to gyroscope measurement i is estimated by forward propagation [120]:

$${}_i\boldsymbol{\Sigma}_g = \exp((\boldsymbol{\omega}_i - \boldsymbol{g})\Delta t)_{i-1}\boldsymbol{\Sigma}_g \exp((\boldsymbol{\omega}_i - \boldsymbol{g})\Delta t)^T + \boldsymbol{B}\Delta t \ , \tag{5.8}$$

for $i \in [1, M]$, and where $\boldsymbol{B}$ is the covariance of the gyroscope process noise. ${}_0\boldsymbol{\Sigma}_g$ is reset to $\boldsymbol{O}_{\boldsymbol{3}\times\boldsymbol{3}}$ before every rotational prior computation. In our implementation, $\boldsymbol{B}$ is set to:

$$\boldsymbol{B} = \boldsymbol{I}_{3\times3}\sigma_g^2, \tag{5.9}$$

where $\sigma_g = 1.1e^{-2}$.

Within this configuration motion segmentation is not estimated, which is equivalent to having a segmentation of $b_i = 1$ for all clusters. As a consequence, all pixels contribute within the visual odometry procedure, including those representing moving objects.

Proposed Configuration: Joint Gyroscope and Motion Segmentation Integration We propose a fusion approach that directly tackles motion conflict by computing an adaptive energy term G which favours frame-to-model alignment in static scenes and allocates a stronger influence to the gyroscope in dynamic scenes. To achieve this effect, we weigh the rotational prior inversely proportional to the estimated segmentation $\boldsymbol{b}$:

$$G(\boldsymbol{\xi}, \boldsymbol{b}) = \sum_{i=1}^{K} K_i (1 - b_i)^2 \boldsymbol{r}_g(\boldsymbol{\xi})^T \boldsymbol{r}_g(\boldsymbol{\xi}) \;, \tag{5.10}$$

where K_i represent the number of points with valid depth in cluster i. This results in strong reliance on the gyroscope in dynamic scenes, as we limit the potential of moving objects within images to counteract the contribution of the motion prior.

In static scenes, however, vision is given more influence. This strategy is useful when performing sensor fusion with low-cost MEMS gyroscopes, which can introduce noise in the estimation system. In addition, frame-to-model alignment does not drift if the camera is continuously exploring the same environment.

5.2.3 Segmentation Prior

The original StaticFusion approach initialises the static/dynamic segmentation $\boldsymbol{b}$ by computing per-cluster residuals between the current frame and the model prediction assuming zero motion (Eq. 4.11).

However, given the estimated rotational increment from the gyroscope, ${}^{(g)}\boldsymbol{R}_{C_t \to C_{t+1}}$, we compute a more accurate initialisation by first warping the current image using this increment before computing residuals. For this computation, the rotational estimate ${}^{(g)}\boldsymbol{R}_{C_t \to C_{t+1}}$ is converted to a homogeneous transformation ${}^{(g)}\boldsymbol{T}_{C_t \to C_{t+1}}$ where the translational component is set to zero.

$$\bar{r}_i = \frac{\sum_{k=1}^{K_i} |Z_M(\mathcal{W}_T(\boldsymbol{x}^k, {}^{(g)}\boldsymbol{T}_{C_t \to C_{t+1}})) - |{}^{(g)}\boldsymbol{T}_{C_t \to C_{t+1}}\, \pi^{-1}(\boldsymbol{x}^p, Z_D(\boldsymbol{x}^p))|_z|}{K_i} \;, \tag{5.11}$$

where the warping function $\mathcal{W}_T$ takes the form:

$$\mathcal{W}_T(\boldsymbol{x}^p, \boldsymbol{T}) = \pi(\boldsymbol{T}\, \pi^{-1}(\boldsymbol{x}^p, Z_D(\boldsymbol{x}^p))) \;. \tag{5.12}$$

This quantity is used to compute a more accurate prior for the segmentation of cluster i:

$$b_i^P = 1 - k_p \bar{r}_i \;, \tag{5.13}$$

which takes the place of the original segmentation prior formulation as described in the previous chapter in Eq. 4.11.

5.2.4 Formulations

Finally, we can specify two novel configurations which we will evaluate:

1. Baseline FTM-G: Frame-to-model alignment with a rotational prior (with $b_i = 1, \forall i$):
$$E(\boldsymbol{\xi}) = D(\boldsymbol{\xi}) + G(\boldsymbol{\xi}) \ . \tag{5.14}$$

2. Proposed FTM-MS-G: Frame-to-model alignment with motion segmentation and a rotational prior:
$$E(\boldsymbol{\xi}, \boldsymbol{b}) = D(\boldsymbol{\xi}, \boldsymbol{b}) + S(\boldsymbol{b}) + G(\boldsymbol{\xi}, \boldsymbol{b}) \ . \tag{5.15}$$

These cost functions, as well as the original formulation from Eq. 5.1, are solved using the same iteratively re-weighted least squares procedure described in the previous chapter.

Algorithm 5 lists the procedure for implementing the FTM-MS-G formulation. It reproduces the frame-to-model alignment in StaticFusion (described in Algorithm 3), with highlights in blue representing the changes required for gyroscope rotational prior integration.

Next, we evaluate all configurations against our dynamic scenes dataset.

5.3 Evaluation

To evaluate the approach we collected a dataset containing a variety of degeneracies which cause typical RGB-D SLAM systems to fail such as (1) sequences with motion present in $> 50\%$ of the image, (2) changes in the scene between observation and re-observation of a particular location and (3) aggressive camera motion.

5.3.1 Dataset Collection

We used the Intel RealSense ZR300 camera which consists of an RGB-D camera synchronised with an IMU containing gyroscopes and accelerometers. We used the Intel-provided ROS driver to collect our dataset, which generates RGB-D data at 30Hz and gyroscope data at 200Hz. We performed offline extrinsic calibration between the colour camera and the IMU using the Kalibr system [73].

Algorithm 5: Frame-to-model alignment using inertial sensing and motion segmentation

```
Input: $I_D, Z_D, I_M, Z_M, \boldsymbol{\xi}_g$
compute image pyramids for $I_D, Z_D, I_M, Z_M$;
initialise $\boldsymbol{\xi} = \mathbf{0}_{6\times1}$;
initialise $\boldsymbol{b} = \mathbf{0}_{K\times1}$;
initialise $\hat{\boldsymbol{\xi}}_g = \boldsymbol{\xi}_g$;
foreach level of the pyramid do
    foreach iteration do
        warp $I_D$ towards $I_M$ given current $\boldsymbol{\xi}$;
        warp $Z_D$ towards $Z_M$ given current $\boldsymbol{\xi}$;
        compute segmentation prior $\boldsymbol{b}_P$ (5.13);
        if $\boldsymbol{b}$ not initialised then
            $\boldsymbol{b} = \boldsymbol{b}_P$
        compute weights $w_I, w_Z$ (4.7, 4.8);
        compute visual odometry residuals $\boldsymbol{r}_I$ and $\boldsymbol{r}_Z$ (4.3 ,4.2);
        compute motion prior residual $r_g$ (5.6);
        foreach IRLS iteration do
            compute Cauchy weights $W_I, W_Z$ using segmentation $\boldsymbol{b}$;
            solve for $\Delta\boldsymbol{\xi}$ (4.5);
                  $(\boldsymbol{J}_I^\top W_I \boldsymbol{J}_I + \boldsymbol{J}_Z^\top W_Z \boldsymbol{J}_Z + w_g \boldsymbol{J}_g^\top \boldsymbol{J}_g)\Delta\boldsymbol{\xi} =$
              $-(\boldsymbol{J}_I^\top W_Z \boldsymbol{r}_I + \boldsymbol{J}_Z^\top W_Z \boldsymbol{r}_Z + w_g \boldsymbol{J}_g^\top \boldsymbol{r}_g)$;
            update residuals $\boldsymbol{r}_I$ and $\boldsymbol{r}_Z$;
            solve for segmentation $\boldsymbol{b}$ (4.13);
        update $\boldsymbol{\xi}$;
              $\boldsymbol{\xi} = \log(\exp(\boldsymbol{\xi})\exp(\Delta\boldsymbol{\xi}))$;
        update $\hat{\boldsymbol{\xi}}$;
              $\hat{\boldsymbol{\xi}}_g = \log(\exp(\boldsymbol{\xi}_g)\exp(\boldsymbol{\xi})^{-1})$
```

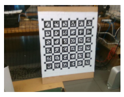
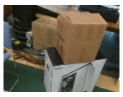
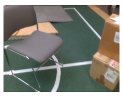
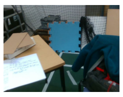
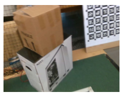
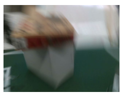
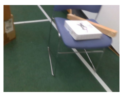
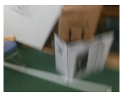
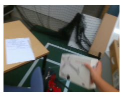
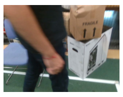
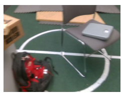
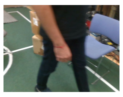
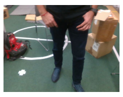
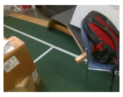
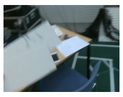
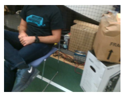
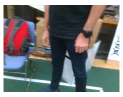
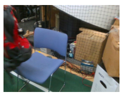
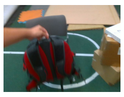
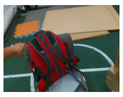
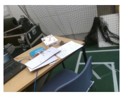
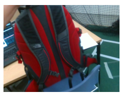
Type	No.	Selected Images						Time[s]	Len[m]
slow camera motion	1							83	24
slow camera motion loop closure	2							50	12
aggressive camera motion loop closure	3							81	22
minimal scene dynamics	4							70	18
medium scene dynamics	5							74	29
medium scene dynamics loop closure	6							76	19
strong scene dynamics environment changes	7							101	25
strong scene dynamics environment changes	8							86	24

Figure 5.1: A visual description of our collected dataset. Each sequence is progressively more complicated as moving objects appear more frequently and occupy an increasing portion of the camera field of view. We include the duration and length of each trajectory.

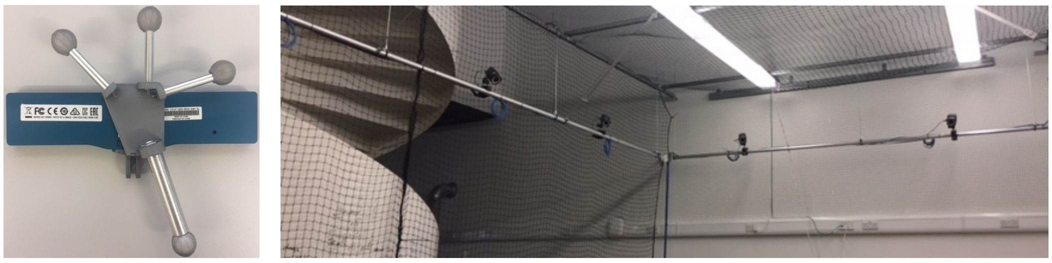


Figure 5.2: Our evaluation set-up. **Left:** Our Intel RealSense ZR300 camera with mounted motion capture markers. **Right:** Our Vicon system.

The dataset was acquired within a Vicon motion capture arena which allows us to quantify the effect that these challenges have on our proposed approach.

We designed a frame in AutoDesk to connect a set of Vicon markers to the Intel ZR300. The camera URDF provides coordinate frames for the colour camera optical frame as well as the bottom screw frame and we use these to perform spatial calibration between the motion capture system and the camera. Figure 5.2 illustrates our set-up. We also synchronise the Vicon computer and the data collection laptop clocks through the network.

An overview of the dataset can be seen in Figure 5.1. Sequences are ordered in increasing complexity from top to bottom. For each individual experiment, a batch of images are shown in chronological order. These sequences can be characterised as follows:

- Sequences 1 - 3 are recorded in static scenes and only vary regarding the length of the trajectory and the speed of the camera motion.
- Sequence 4 contains mostly a static scene and occasionally a dynamic object of small size moving in front of the camera on 2 or 3 occasions. This object would never occupy $> 30\%$ of the image.
- Sequences 5 and 6 contain large objects which move in and out of the camera view. When they appear, the objects move continuously and occasionally occupy $\sim 50\%$ of the image.
- Sequences 7 and 8 contain significantly larger objects which alternate between being in motion and being stationary. These objects would frequently occupy $> 50\%$ of the image.

The significant differences between Sequences 5-6 and 7-8 are the larger sizes of the objects present in the latter sequences. Because the moving objects transition between being in motion and being stationary, the visual SLAM system is required to repeatedly fuse and remove them from the 3D model it is creating.

5.3.2 Quantitative Evaluation

When evaluating trajectory accuracy, we use the established metrics defined by Sturm *et al.* [26], namely Absolute Trajectory Error (ATE) to evaluate the error over the whole trajectory estimate and Relative Pose Error (RPE) to analyse the local drift. For both metrics, we compute the Root Mean Square Error (RMSE).

To understand the importance of both motion segmentation and rotational prior, we evaluate and compare the configurations of our system, as stated in Section 5.2.4. In addition, we compare our system with state-of-the-art approaches:

Sequence	Translational ATE RMSE (cm)				
	VOSF	EF	SF	FTM-G	FTM-MS-G (Proposed)
1	16.67	14.77	5.86	**3.88**	4.69
2	7.15	**4.11**	5.77	4.16	4.56
3	58.53	**6.06**	8.87	11.49	9.42
4	12.93**	48.38*	36.41*	16.17**	**12.28**
5	43.25*	37.41*	11.28	17.63	**6.23**
6	17.64**	47.44*	10.84	**7.15**	8.36
7	32.30*	43.51*	71.48*	27.30**	**9.62**
8	65.85*	49.81*	209.33*	50.90*	**6.12**

Table 5.1: Absolute Trajectory Error. FTM-MS-G achieves the most accurate results through the integration of motion segmentation and gyroscope rotational prior. * marks that tracking failures occurred during this sequence and ** marks significant drift without failure.

- The system for joint visual odometry and scene flow (VOSF) by Jaimez *et al*. [79], which is designed to handle dynamic scenes. We investigate the performance improvement that can be expected from building a 3D model of the static environment.

- ElasticFusion (EF) [54], a dense RGB-D SLAM system designed for static scenes. In this case, we investigate the benefits of motion segmentation or gyroscope integration for handling dynamic environments.

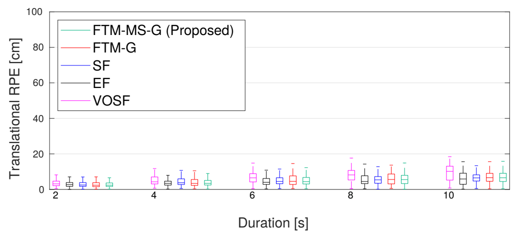
- StaticFusion (SF) [125], our baseline system. As mentioned, we want to compare the performance of gyroscope integration against motion segmentation when handling moving objects.

Table 5.1 reports ATE results for all trajectories and configurations. Figure 5.3 contains RPE results over increasing time intervals of selected sequences with increasing amounts of dynamics present (Sequences 2, 4, 5 and 7).

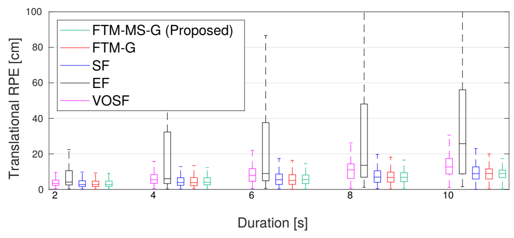
We note that in static scenes (Sequences 1 and 2), the four map-based approaches (SF, EF, FTM-G and FTM-MS-G) report similar performance, while VOSF suffers from increased drift due to the lack of a 3D model.

Starting with the presence of minimal scene dynamics (Figure 5.3c - Sequence 4), the performance of VOSF and EF degrade. This happens because VOSF performs frame-to-frame alignment and cannot recover from an incorrectly computed pose or segmentation, while EF uses all pixels within the image during visual odometry, which causes it to fail when moving objects occupy a significant portion of the image.

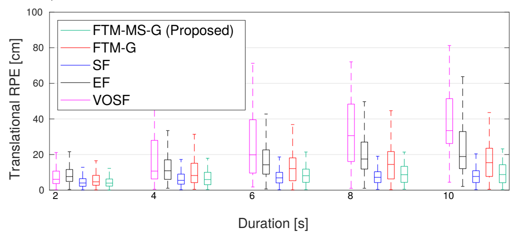
For minimal to medium scene dynamics (Sequences 5 and 6), the performance of



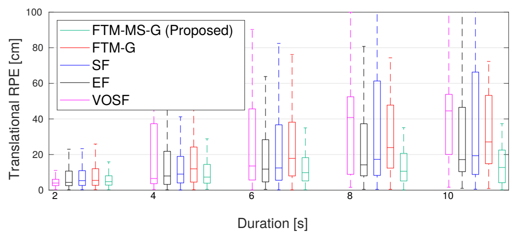
(a) Sequence 2 (slow camera motion, loop closure)



(c) Sequence 4 (minimal scene dynamics)



(b) Sequence 5 (medium scene dynamics)



(d) Sequence 7 (strong scene dynamics, environment changes)

Figure 5.3: RPE for increasing time intervals. In static scenes containing rich texture, all approaches are robust and accurate. However, as the scene becomes more challenging, the contribution of the gyroscope becomes evident and bounds the drift of the estimate. Finally, in scenes with strong dynamics, both motion segmentation and motion prior are necessary to achieve a low drift estimate.

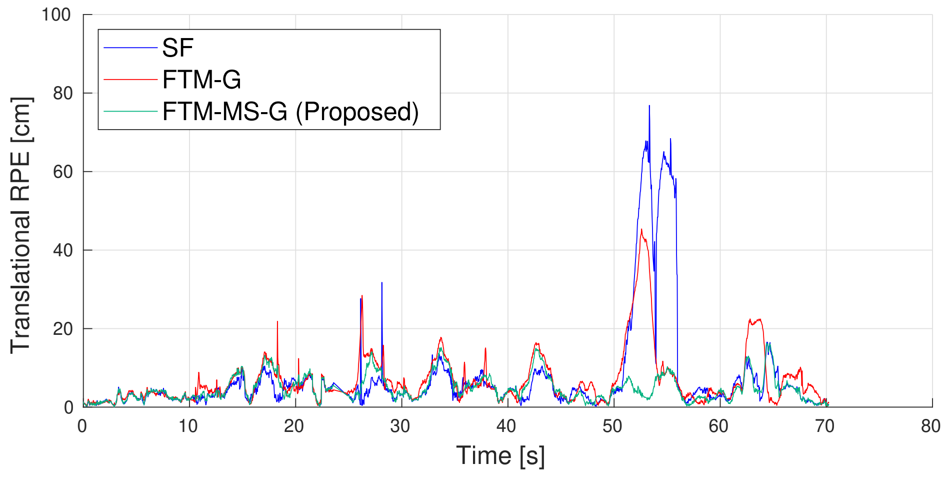


Figure 5.4: RPE over one second intervals for Sequence 5 (medium scene dynamics). FTM-MS-G provides the most accurate results. FTM-G suffers from significant drift, while SF has a catastrophic failure after 50 seconds.

SF and FTM-G are comparable, implying that either the integration of motion segmentation or rotational prior is sufficient for good performance in this case.

In highly dynamic environments (Sequences 7 and 8), FTM-G outperforms the pure vision approach, SF. This can also be seen in Figure 5.4, which shows the translational RPE for Sequence 5: the gyroscope prevents the solver from converging to a completely incorrect estimate, as it can happen for SF when moving objects occupy the majority of the image. However, FTM-G accumulates significant drift as the amount of points belonging to dynamic objects overwhelms the rotational prior within the optimisation.

Importantly, across all scenarios, the most accurate results are achieved with FTM-MS-G, which demonstrates that both motion prior and motion segmentation are necessary to achieve robustness in dynamic scenes. Additionally, as our fusion strategy attributes a larger weight to the gyroscope in dynamic scenes and a larger weight to vision in static scenes, we can achieve accurate results in both scenarios.

We ran our experiments on a workstation with an Intel(R) Core(TM) i7-6700K CPU at 4.00GHz and a GeForce GTX 970 GPU using the Ubuntu 16.04 operating system and we achieve a run-time of 53milliseconds/frame with loop closures enabled.

5.3.3 Qualitative Evaluation

We present illustrations of the performance of FTM-MS-G for the most difficult sequences we recorded, namely Sequences 7 and 8. The reader is also encouraged to inspect the videos of these experiments in Appendix A.3.3.1.

Figure 5.5 illustrates the performance of FTM-MS-G for Sequence 7. During this sequence, a person is initially sitting on a chair and becomes fused within the reconstruction. Afterwards, the person leaves their chair (as the camera is facing them) and moves a backpack across the field of vision. The figure demonstrates the capability of our approach to robustly track the pose of the camera when moving objects occupy a large portion of the image, and also to handle large changes in the environment and update the reconstruction accordingly.

During Sequence 8, a backpack is moved between multiple chairs while the camera is following the backpack's motion. The performance of our algorithm is shown in Figure 5.6, where the upper part illustrates the map changing in time reflecting the new location of the backpack.

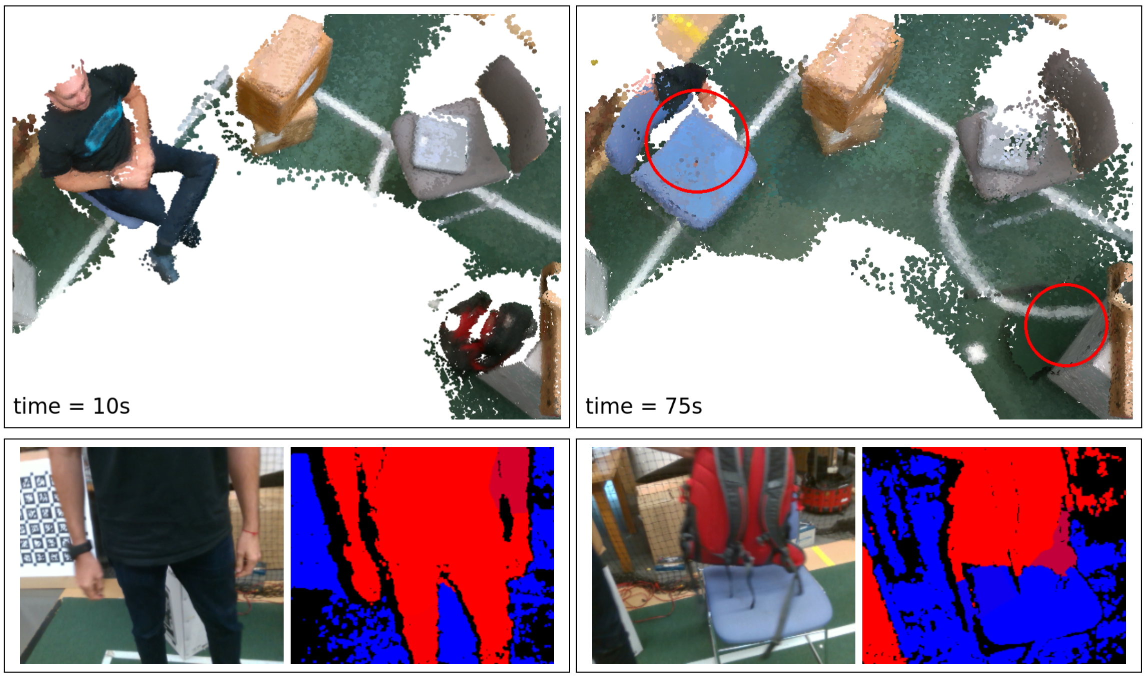


Figure 5.5: **Top**: Dense 3D reconstructions produced by our approach for a sequence where a person stands up, walks into the camera view and moves a bag. Red circles indicate changes in the map due to dynamics (or movement) during the experiment. **Bottom**: Two image pairs of input colour images (left) and corresponding segmentation images (right), where dynamic objects are coloured red and static structure is blue.

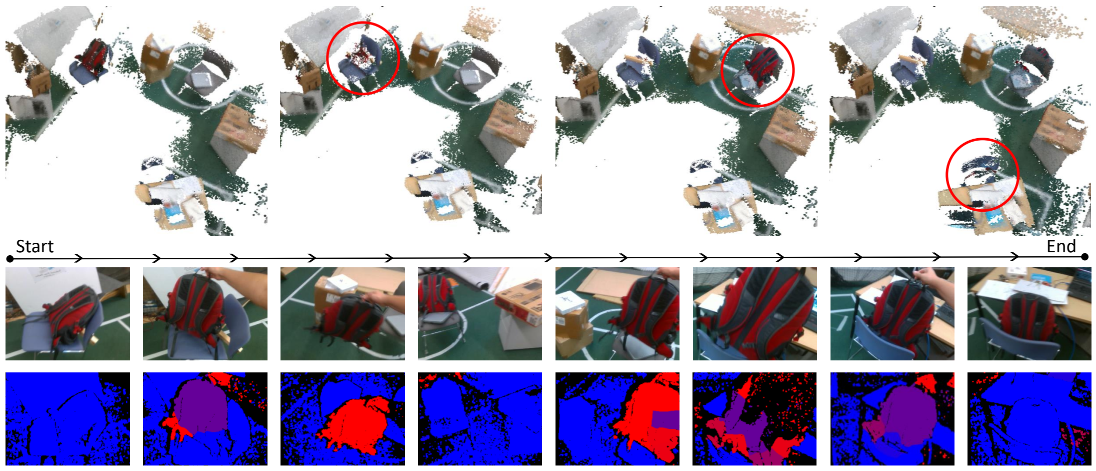


Figure 5.6: Qualitative evaluation of the performance of the algorithm during Sequence 8 (strong scene dynamics, changes in the environment). During this sequence, a bag is moved between different chairs, while the camera is recording its motion. **Top**: Evolution of the map, where red circles highlight changes. **Bottom**: Sample input images and computed segmentations (red - dynamic object, blue - static object).

The lower rows show corresponding colour and static/dynamic segmentation results (where moving objects are segmented as red and static objects are segmented as blue). While the backpack occupies a large portion of the field of view, FTM-MS-G succeeds in computing a correct motion segmentation and camera pose estimate thanks to the integration of the rotational prior.

A positive feature of our system is that we are able to detect and segment general moving objects by simple reasoning about residuals and without requiring semantic information.

For completeness, the videos illustrating the performance of FTM-MS-G for the remaining Sequences 1-6 can be seen in Appendix A.3.3.2.

5.3.4 Limitations

The Intel RealSense ZR300 is equipped with a consumer-grade MEMS IMU and we find that the rotational prior from our gyroscope is only reliable for short durations. If dynamic objects cover the majority of the image for more than $\sim$10 seconds, the rotational prior will accumulate too much noise, causing the pose estimate to diverge. This could be improved through the use of a high-end gyroscope.

Another limitation of our approach is that we did not explore the impact of a full motion prior which would include translation. We expect translational or linear velocity priors originating from wheel odometry or leg odometry to be much more reliable than those estimated using accelerometry and believe that future research in this type of fusion would lead to very robust and reliable systems.

Finally, we discuss the performance of the loop closure detection systems of ElasticFusion in dynamic environments. The global loop closure detection system is an appearance based technique which builds a database of renderings of the 3D map from the different poses of the camera. It attempts to match renderings of the map from the current pose against those stored in the database, which can suffer when the map changes significantly. We observe experimentally that loop closures are detected mostly in areas of the environment which have not changed.

While in this thesis we have mostly focused on techniques for robust visual odometry, achieving an overall reliable SLAM system also requires further research in developing place recognition techniques which can support environmental changes.

5.4 Discussion

The purpose of this chapter was to perform further evaluation of the StaticFusion system introduced in Chapter 4 by extending it with a motion prior computed from a gyroscope and comparing the impact of this additional prior against the originally proposed motion segmentation strategy.

For this, we have recorded a dataset with the Intel RealSense ZR300 sensor which is equipped with an RGB-D camera and an IMU in a motion capture arena to record ground truth trajectory estimates. The sequences in this dataset range from simple static environments in which current visual SLAM systems work well, to sequences with moving objects that occupy a large portion of the image. In some sequences, these moving objects also alternate between being static and in motion, in order to force the visual SLAM system to continuously update its map of the static scenes.

In our evaluation, we were mainly interested in comparing baseline StaticFusion against a configuration that combines frame-to-model alignment with gyroscope information as well as the configuration which combines both gyroscope information and motion segmentation.

We found that in scenes with minor to moderate levels of motion, motion segmentation and gyroscope integration achieve similar accuracy while outperforming related approaches. In highly dynamic scenes, the gyroscope provides increased robustness and outperforms StaticFusion, although significant drift is observed in the pose estimate due to the unfiltered moving objects in the image.

The approach which combines both gyroscope motion prior and motion segmentation was found to perform best, as the gyroscope aids with robustness and the motion segmentation is useful for removing moving objects during visual odometry, resulting in the most accurate results for the proposed full fusion system.

Part III

Conclusions

Chapter 6

Conclusions

In this thesis we have tackled the problem of visual SLAM and focused on techniques for robust pose estimation in challenging real-world environments and applications.

In designing our solutions, we focused on stereo/RGB-D visual sensors because in our opinion these are better than monocular cameras with respect to achieving robust performances. Stereo/RGB-D visual odometry has several advantages compared with monocular, as it does not require specific initialisation procedures, does not suffer from scale drift and can handle pure rotational motions which constitute degeneracies for monocular visual odometry.

Nevertheless, stereo/RGB-D odometry is fragile when used in unconstrained environments and its limitations could impede its deployment in real-world applications such as Robotics. Its failure cases include:

- Textureless scenes could lead to unconstrained motion estimates.
- Moving objects contradict the static-scene assumptions in most state-of-the art visual odometry approaches and lead to significant drift or failure.
- Illumination change can lead to high residuals for valid pixel correspondences which can cause the visual odometry solver to converge to the wrong solution.
- Limited operational range and field of view of the sensors. In the case of the stereo set-up, this limit is defined by the baseline, while for RGB-D sensors this range is between 0.5m and 5m. In addition, the sensors we used also had a small field-of-view, with $80^\circ \times 80^\circ$ for the MultiSense SL and $58^\circ \times 45^\circ$ for the Asus Xtion Pro Live and Intel RealSense ZR300.

6.1 Contributions

We have tackled these problems though a combination of motion priors and motion segmentation. In summary, our contributions are:

- Chapter 3 researched a visual SLAM system which would be sufficiently robust and reliable to operate on our humanoid robot Valkyrie. The requirements for this system were that it should provide accurate pose estimates and produce a map of the environment that is dense and accurate enough to enable collision-free motion planning. To achieve this, we proposed a system which fuses dense stereo visual odometry with a motion prior from the kinematic-inertial state estimator of the robot. This system was successful in producing pose estimates which were robust to the challenges specified above and also produced a reconstruction of sufficient accuracy for collision-free motion planning. However, this system had limited support for handling moving objects and it did not support changes to the created map.

- Chapter 4 focused on the problem of handling dynamic objects during visual odometry and proposed a technique for simultaneous estimation of camera motion and motion segmentation of the current image into static and dynamic components. Given those estimated quantities, the approach then built a 3D model of the environment which only contained the static components. The approach also provided a simple heuristic for removing objects from the 3D model if they started to move. It showed superior performance against related state-of-the-art dense visual odometry techniques when handling dynamic environments. However, as it was a purely vision-based approach, it was susceptible to failure when the majority of the image contained moving objects.

- Chapter 5 built on the system proposed in Chapter 4 and extended it using a motion prior from a gyroscope, improving performance significantly in sequences where moving objects occupy the majority of the image. This work also allowed for evaluating and comparing the importance of motion segmentation and motion prior integration for handling moving objects. We found that in moderately dynamic scenes, the inclusion of either motion segmentation or the rotational prior had similar effects. For highly dynamic scenes, the rotational prior out-performed the motion segmentation and was thus more important. Overall, the proposed system which combined both priors yielded the best results.

6.2 Limitations and Future Directions

We found significant general benefits in the use of motion priors for handling all mentioned challenges, in particular leg odometry for linear velocity and gyroscopes for angular velocity. One aspect which is not addressed in our work is how to handle errors within motion priors. For example, leg or wheel slippage may induce high velocity errors which in turn would corrupt the odometry estimates. Modelling and handling such errors is a crucial component within a robust multi-sensor fusion SLAM system.

Regarding sensor fusion, it is not clear in practice how well the sensors should be synchronised. Many visual-inertial techniques advise to use a sensor where the camera and IMU are hardware synchronised for best results. However, in a multi-sensor fusion system such as the one we developed for the humanoid robot, it is not possible to hardware synchronise the camera, joint encoders, force-torque sensors and IMU. Instead, the synchronisation system for our humanoid robot is based on time-of-arrival, with errors originating from the varying latencies of the sensors. Although not perfect, this seemed sufficient for our application, however one reason for this could be that the robot is not moving at high velocities, as compared with a drone. It would be interesting to understand if techniques such as time-of-arrival are sufficient for fast robots such as drones, or if more accurate sensor synchronisation is required in this case.

Another aspect is to model and track the moving objects within the scene, as currently these are being regarded as noise and discarded within our approaches. Obtaining accurate models of moving objects would be useful for visual odometry as it would provide prior information of what pixels in the current image are reliable and should be used for motion estimation.

The use of semantic information is a very important research direction, with the goal of building semantically annotated 3D models. Robots could then use these models not only as a source of collision information, but also to reason about which objects they could interact with. Regarding our problem of robust visual odometry for dynamic scenes, semantic information would be useful for detecting people or animals and automatically labelling those pixels as moving objects, excluding them from motion estimation.

Regarding StaticFusion, currently this approach requires an initialisation sequence with minimal dynamics in the scene to build the first reliable 3D model. Following this, we compute independent motion estimates for each frame and no consistency is enforced between moving objects segmented in consecutive frames. An interesting research direction would be to perform multi-frame joint estimation of camera motion and motion segmentation which would enforce that moving objects are detected consistently between consecutive frames and thus produce better results than the current systems.

Finally, in our work we have not considered the problem of place recognition for implementing loop closures. As we are interested in supporting dynamic environments, it is also very important to research techniques for robust place recognition that could support changes in the environment.

To conclude, we hope that our work is a step towards solving these problems and supports progress in the field of visual SLAM.

Part IV

Appendix

Appendix A

Source Code, Dataset and Videos

In this appendix we provide links to the software and datasets developed and collected for this PhD. In addition we provide videos describing some of the experiments conducted for this research.

A.1 Source Code

The open-source C++ implementation for StaticFusion, the method proposed in Chapter 4, can be found here:

`https://github.com/raluca-scona/staticfusion`

A.2 Dataset

The dataset used for the evaluation in Chapter 5 is available here:

`http://conferences.inf.ed.ac.uk/rgbd-imu-dynamic-dataset/`

It consists of sequences collected with the Intel RealSense ZR300 (containing an RGB-D sensor and an IMU) in a Vicon motion capture arena. The sequences contain little to high levels of dynamic motion and enable the evaluation of visual odometry robustness in dynamic scenes.

A.3 Videos

In this section we list videos of experiments performed for each chapter within this thesis.

A.3.1 Camera Pose Estimation with Motion Prior Integration Applied to Humanoid SLAM

Video describing experiments with the humanoid robot Valkyrie from Chapter 3:

https://youtu.be/dv0ADFQkU3E

This includes:

- A demonstration of closed-loop visual SLAM running on Valkyrie.
- A demonstration of the robustness of the proposed system to visual challenges including illumination changes, textureless scenes, motion blur and moving objects in the scene.
- A visualisation of the 3D stereo reconstruction compared to LIDAR point clouds.
- An experiment demonstrating collision-free motion planning, where the robot plans a reaching motion using the 3D reconstruction to avoid colliding with the environment.

A.3.2 Joint Visual Odometry and Motion Segmentation for SLAM in Dynamic Environments

Video describing experiments from Chapter 4:

https://youtu.be/UVsqIgxHoBM

This includes:

- A demonstration of the robustness of StaticFusion to moving objects in the scene on a self-recorded sequence.
- A demonstration of StaticFusion compared to state-of-the-art approaches ElasticFusion [54], Co-Fusion [108] and Joint Visual Odometry & Scene Flow [79] on sequence *fr3/walking_static* from the TUM/Freiburg dataset [26].
- The performance of StaticFusion on a self-recorded 'selfie sequence' where the moving person occasionally occupies more than 50% of the image.
- The performance of StaticFusion on the static sequence *fr1/desk* from the TUM/Freiburg dataset.

A.3.3 Dense RGB-D SLAM in Dynamic Environments using Inertial Sensing and Motion Segmentation

For Chapter 5 we recorded a dataset of RGB-D-gyroscope sequences in scenes with increasing numbers of moving objects and with camera trajectory ground-truth estimates from a Vicon motion capture system.

For the video recordings of the experiments, the estimated trajectory is shown in red and the ground-truth trajectory in black.

A.3.3.1 Notable Results

The following video provides an overview of the notable results:

https://youtu.be/VTCYLFUGJnk

This includes:

- A demonstration of the performance of the proposed system for Sequence 8, where an object is moved between different locations while the camera is tracking its motion. This experiment contains strong scene dynamics and environment changes.
- A comparison of the effects of moving objects on (1) simple frame-to-model alignment, (2) StaticFusion and (3) gyroscope fusion neglecting motion segmentation.
- A demonstration of the performance of the proposed system for Sequence 7, where a person moves in front of the camera and also moves an object. Similarly to Sequence 8, this sequence contains large objects which frequently occupy more than 50% of the image and alternate between being in motion and being stationary.

A.3.3.2 Results on Entire Dataset

As this dataset is new, we list in this section videos of the performance of our system for each recorded sequence.

- Sequence 1 – the camera is moved back and forth in a static scene:

  https://youtu.be/kRTxvcglJsE

- Sequence 2 – the camera is moved in a loop in a static scene:

  https://youtu.be/QVQNaQaDheM

- Sequence 3 – the camera is moved in a loop and it is also shaken during this trajectory. The scene is static:

  https://youtu.be/r1LO3ccLn4w

- Sequence 4 – objects occupying around 30% of the image appear occasionally in view:

  https://youtu.be/pa-G0pWGqaU

- Sequence 5 – objects which occasionally occupy at most 50% of the image appear in view:

  https://youtu.be/EF2paCaj0yU

- Sequence 6 – the camera is moved in a loop and occasionally there are objects appearing which occupy less than 50% of the field of view:

  https://youtu.be/Tj801xQzmpI

Appendix B

Other Contributions

This PhD contributed to another research project which does not form a part of this thesis:

> **Overlap-based ICP Tuning for Robust Localization of a Humanoid Robot** [128]
> Simona Nobili, Raluca Scona, Marco Caravagna, Maurice Fallon. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2017*

Personal contributions include:

- Assisting with data collection using the Valkyrie robot.
- Assisting with running closed-loop experiments using the Valkyrie robot.
- Contributing to the paper content and figures.

Shortened Abstract In this paper we focus on the localisation of a humanoid robot with limited LIDAR field-of-view in a semi-structured environment. We analyse the effect of overlap variations on point cloud registration performance and demonstrate that where overlap varies, outlier filtering needs to be tuned accordingly. We define a novel parameter which gives a measure of this overlap. In this context, we propose a strategy for robust non-incremental registration. The pre-filtering module selects planar macro-features from the input clouds, discarding clutter. Outlier filtering is automatically tuned at run-time to allow registration to a common reference in conditions of non-uniform overlap. An extensive experimental demonstration is presented which characterises the performance of the algorithm using two humanoids:

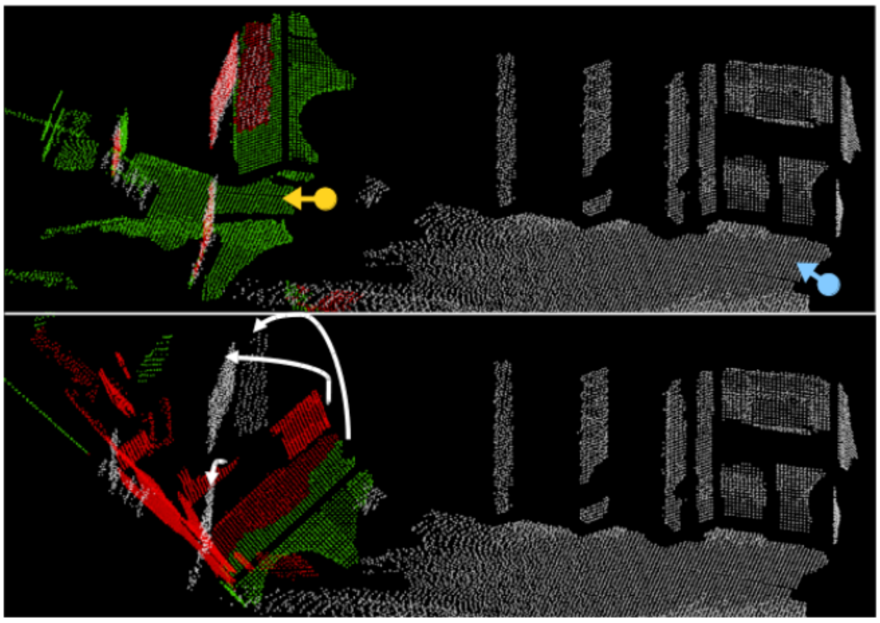


Figure B.1: White points belong to the reference cloud captured with respect to the blue pose. The red points are accepted inlier matches and the green points are rejected outliers, all belonging to the reading cloud, captured from the yellow pose. Bottom: overlap variation is not considered, leading in turn to incorrect point correspondences and registration failure. Top: overlap variation is taken into account and outlier correspondences are filtered out, leading to a successful registration. Photo credit Nobili *et al*. [128]

the NASA Valkyrie, in a laboratory environment, and the Boston Dynamics Atlas, during the DARPA Robotics Challenge Finals.

Results showing the effect of outlier filtering based on point cloud overlap analysis are shown in Figure B.1.

Bibliography

[1] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars exploration rovers," *J. of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[2] C. F. Olson, L. H. Matthies, J. R. Wright, R. Li, and K. Di, "Visual terrain mapping for Mars exploration," *Computer Vision and Image Understanding*, vol. 105, no. 1, pp. 73–85, 2007.

[3] M. Johnson-Roberson, O. Pizarro, S. B. Williams, and I. Mahon, "Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys," *J. of Field Robotics*, vol. 27, no. 1, pp. 21–51, 2010.

[4] S. Rahman, A. Q. Li, and I. Rekleitis, "Sonar visual inertial SLAM of underwater structures," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 1–7.

[5] N. Weidner, S. Rahman, A. Q. Li, and I. Rekleitis, "Underwater cave mapping using stereo vision," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 5709–5715.

[6] S. Rahman, A. Q. Li, and I. Rekleitis, "SVIn2: An underwater SLAM system using sonar, visual, inertial, and depth sensor," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 1861–1868.

[7] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *J. of Robotics and Autonomous Systems*, vol. 62, no. 11, pp. 1646–1656, 2014.

[8] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular SLAM with multiple micro aerial vehicles," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 3962–3970.

[9] P. Schmuck and M. Chli, "Multi-UAV collaborative monocular SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 3863–3870.

[10] L. Teixeira and M. Chli, "Real-time mesh-based scene estimation for aerial inspection," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 4863–4869.

[11] M. Karrer, M. Kamel, R. Siegwart, and M. Chli, "Real-time dense surface reconstruction for aerial manipulation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 1601–1608.

[12] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[13] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *J. of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[14] C. Urmson, C. Ragusa, D. Ray, J. Anhalt, D. Bartz, T. Galatali, A. Gutierrez, J. Johnston, S. Harbaugh, H. Yu Kato *et al.*, "A robust approach to high-speed navigation for unrehearsed desert terrain," *J. of Field Robotics*, vol. 23, no. 8, pp. 467–508, 2006.

[15] P. G. Trepagnier, J. Nagel, P. M. Kinney, C. Koutsougeras, and M. Dooner, "Kat-5: Robust systems for autonomous vehicle navigation in challenging and unknown terrain," *J. of Field Robotics*, vol. 23, no. 8, pp. 509–526, 2006.

[16] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, J. Carff, W. Rifenburgh, P. Kaveti, W. Straatman, J. Smith, M. Griffioen, B. Layton, T. de Boer, T. Koolen, P. Neuhaus, and J. Pratt, "Team IHMC's lessons learned from the DARPA robotics challenge trials," *J. of Field Robotics*, vol. 32, no. 2, pp. 192–208, 2015.

[17] S. Kuindersma, R. Deits, M. F. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion plan-

ning, estimation, and control design for Atlas," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[18] O. G. Grasa, J. Civera, and J. Montiel, "EKF monocular SLAM with relocalization for laparoscopic sequences," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 4816–4821.

[19] D. Burschka, M. Li, M. Ishii, R. H. Taylor, and G. D. Hager, "Scale-invariant registration of monocular endoscopic images to CT-scans for sinus surgery," *Medical Image Analysis*, vol. 9, no. 5, pp. 413–426, 2005.

[20] D. Stoyanov, A. Darzi, and G. Z. Yang, "A practical approach towards accurate dense 3D depth recovery for robotic laparoscopic surgery," *Computer Aided Surgery*, vol. 10, no. 4, pp. 199–208, 2005.

[21] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.

[22] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building Rome in a day," *Communications of the ACM*, vol. 54, no. 10, pp. 105–112, 2011.

[23] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE/ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 1–10.

[24] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[25] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 1935–1942.

[26] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[27] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 40, no. 3, pp. 611–625, 2018.

[28] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Intl. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[29] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, “ORB: An efficient alternative to SIFT or SURF.” in *Intl. Conf. on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[30] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Eur. Conf. on Computer Vision (ECCV)*, 2006, pp. 430–443.

[31] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry,” in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2004.

[32] A. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, “Visual odometry and mapping for autonomous flight using an RGB-D camera,” in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, 2011, pp. 235–252.

[33] J. Engel, J. Sturm, and D. Cremers, “Semi-dense visual odometry for a monocular camera,” in *Intl. Conf. on Computer Vision (ICCV)*, 2013, pp. 1449–1456.

[34] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in *Eur. Conf. on Computer Vision (ECCV)*, 2014, pp. 834–849.

[35] R. C. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *Intl. J. of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[36] J. J. Leonard and H. F. Durrant-Whyte, “Simultaneous map building and localization for an autonomous mobile robot,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 1991, pp. 1442–1447.

[37] G. Klein and D. Murray, “Improving the agility of keyframe-based SLAM,” in *Eur. Conf. on Computer Vision (ECCV)*, 2008, pp. 802–815.

[38] H. Strasdat, J. Montiel, and A. J. Davison, “Real-time monocular SLAM: Why filter?” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 2657–2664.

[39] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Intl. J. of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[40] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[41] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 4628–4635.

[42] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2013, pp. 1352–1359.

[43] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level SLAM," in *2018 International Conference on 3D Vision (3DV)*, 2018, pp. 32–41.

[44] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[45] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," in *Sensor devices and systems for robotics*, 1989, pp. 253–276.

[46] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[47] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[48] J. Wilhelms and A. Van Gelder, "Octrees for faster isosurface generation," *ACM Transactions on Graphics (TOG)*, vol. 11, no. 3, pp. 201–227, 1992.

[49] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[50] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, “BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, p. 24, 2017.

[51] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, “The Replica dataset: A digital replica of indoor spaces,” *arXiv preprint arXiv:1906.05797*, 2019.

[52] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross, “Surfels: Surface elements as rendering primitives,” in *Proc. of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 335–342.

[53] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, “Real-time 3D reconstruction in dynamic scenes using point-based fusion,” in *International Conference on 3D Vision (3DV)*, 2013, pp. 1–8.

[54] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, “ElasticFusion: Dense SLAM without a pose graph,” *Robotics: Science and Systems (RSS)*, 2015.

[55] W. Gao and R. Tedrake, “SurfelWarp: Efficient non-volumetric single view dynamic reconstruction,” *Robotics: Science and Systems (RSS)*, 2019.

[56] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, “KinectFusion: Real-time dense surface mapping and tracking,” in *IEEE/ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[57] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. Leonard, and J. McDonald, “Real-time large scale dense RGB-D SLAM with volumetric fusion,” *Intl. J. of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[58] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, “Real-time 3D reconstruction at scale using voxel hashing,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 169, 2013.

[59] R. Maier, K. Kim, D. Cremers, J. Kautz, and M. Nießner, “Intrinsic3D: High-quality 3D reconstruction by joint appearance and geometry optimization with

spatially-varying lighting," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3114–3122.

[60] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 3748–3754.

[61] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Intl. Conf. on Computer Vision (ICCV) Workshops*, 2011, pp. 719–722.

[62] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, 1992, pp. 586–606.

[63] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D model acquisition," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, 2002, pp. 438–446.

[64] M. Jaimez and J. Gonzalez-Jimenez, "Fast visual odometry for 3-D range sensors," *IEEE Trans. Robotics*, vol. 31, no. 4, pp. 809–822, 2015.

[65] T. Whelan, M. Kaess, J. J. Leonard, and J. McDonald, "Deformation-based loop closure for large scale dense RGB-D SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 548–555.

[66] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[67] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 1885–1892.

[68] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[69] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense RGB-D-inertial SLAM with map deformations," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 6741–6748.

[70] J. Li, M. Kaess, R. M. Eustice, and M. Johnson-Roberson, "Pose-graph SLAM using forward-looking sonar," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2330–2337, 2018.

[71] C. Houseago, M. Bloesch, and S. Leutenegger, "KO-Fusion: Dense visual SLAM with tightly-coupled kinematic and odometric tracking," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 4054–4060.

[72] D. Wisth, M. Camurri, and M. Fallon, "Robust legged robot state estimation using factor graph optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4507–4514, 2019.

[73] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 1280–1286.

[74] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The TUM VI benchmark for evaluating visual-inertial odometry," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 1680–1687.

[75] T. Tykkälä, A. I. Comport, J.-K. Kämäräinen, and H. Hartikainen, "Live RGB-D camera tracking for television production studios," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 207–217, 2014.

[76] J. Czarnowski, S. Leutenegger, and A. J. Davison, "Semantic texture for robust dense tracking," in *Intl. Conf. on Computer Vision (ICCV) Workshops*, 2017, pp. 860–868.

[77] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2878–2885, 2018.

[78] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 809–816.

[79] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from RGB-D cameras based on geometric clustering," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 3992–3999.

[80] A. Croeze, L. Pittman, and W. Reynolds, "Solving nonlinear least-squares problems with the Gauss-Newton and Levenberg-Marquardt methods," *Technical Report, Louisiana State University*, 2012.

[81] P. E. Gill and W. Murray, "Algorithms for the solution of the nonlinear least-squares problem," *SIAM Journal on Numerical Analysis*, vol. 15, no. 5, pp. 977–992, 1978.

[82] D. Gutiérrez-Gómez, W. Mayol-Cuevas, and J. J. Guerrero, "Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 83–89.

[83] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[84] J.-L. Blanco, "A tutorial on SE(3) transformation parameterizations and on-manifold optimization," *Technical Report, University of Malaga*, vol. 3, 2010.

[85] H. Hirschmüller, "Stereo processing by semi-global matching and mutual information," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 30, no. 2, pp. 328–341, 2008.

[86] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[87] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi, "Real-time RGB-D camera relocalization via randomized ferns for keyframe encoding," *IEEE transactions on visualization and computer graphics*, vol. 21, no. 5, pp. 571–583, 2014.

[88] X. Xinjilefu, S. Feng, and C. Atkeson, "Center of mass estimator for humanoids and its application in modelling error compensation, fall detection and prevention," in *IEEE/RSJ Int. Conf. on Humanoid Robots*, 2015, pp. 67–73.

[89] B. J. Stephens, "State estimation for force-controlled humanoid balance using simple models in the presence of modeling error," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 3994–3999.

[90] T. Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a Momentum-Based Control Framework and Application to

the Humanoid Robot Atlas," *Intl. J. of Humanoid Robotics*, vol. 13, no. 1, pp. 1–34, 2016.

[91] M. F. Fallon, M. Antone, N. Roy, and S. Teller, "Drift-free humanoid state estimation fusing kinematic, inertial and LIDAR sensing," in *IEEE/RSJ Int. Conf. on Humanoid Robots*, 2014, pp. 112–119.

[92] N. Rotella, M. Bloesch, L. Righetti, and S. Schaal, "State estimation for a humanoid robot," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 952–958.

[93] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots - consistent fusion of leg kinematics and IMU," in *Robotics: Science and Systems (RSS)*, 2012.

[94] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart, "State estimation for legged robots on unstable and slippery terrain," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 6058–6064.

[95] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini, "Probabilistic contact estimation and impact detection for state estimation of quadruped robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1023–1030, 2017.

[96] O. Stasse, D. Andrew J, R. Sellaouti, and K. Yokoi, "Real-time 3D SLAM for humanoid robot considering pattern generator information," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006, pp. 348–355.

[97] S. Ahn, S. Yoon, S. Hyung, N. Kwak, and K. S. Roh, "On-board odometry estimation for 3D vision-based SLAM of humanoid robot," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 4006–4012.

[98] G. Oriolo, A. Paolillo, L. Rosa, and M. Vendittelli, "Vision-based odometric localization for humanoids using a kinematic EKF," in *IEEE/RSJ Int. Conf. on Humanoid Robots*, 2012, pp. 153–158.

[99] ——, "Humanoid odometric localization integrating kinematic, inertial and visual information," *Autonomous Robots*, vol. 40, no. 5, pp. 867–879, 2016.

[100] N. Kwak, O. Stasse, T. Foissotte, and K. Yokoi, “3D grid and particle based SLAM for a humanoid robot,” in *IEEE/RSJ Int. Conf. on Humanoid Robots*, 2009, pp. 62–67.

[101] R. Wagner, U. Frese, and B. Bäuml, “Graph SLAM with signed distance function maps on a humanoid robot,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 2691–2698.

[102] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, “Continuous humanoid locomotion over uneven terrain using stereo fusion,” in *IEEE/RSJ Int. Conf. on Humanoid Robots*, 2015, pp. 881–888.

[103] P. Puri, D. Jia, and M. Kaess, “GravityFusion: Real-time dense mapping without pose graph using deformation and orientation,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 6506–6513.

[104] C. Kerl, J. Sturm, and D. Cremers, “Dense visual SLAM for RGB-D cameras,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106.

[105] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “DTAM: Dense tracking and mapping in real-time,” in *Intl. Conf. on Computer Vision (ICCV)*, 2011, pp. 2320–2327.

[106] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: a versatile and accurate monocular SLAM system,” *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[107] D. H. Kim and J. H. Kim, “Effective background model-based RGB-D dense visual odometry in a dynamic environment,” *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1565–1573, 2016.

[108] M. Rünz and L. Agapito, “Co-Fusion: Real-time segmentation, tracking and fusion of multiple objects,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 4471–4478.

[109] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, “Learning to refine object segments,” in *Eur. Conf. on Computer Vision (ECCV)*, 2016, pp. 75–91.

[110] M. Rünz and L. Agapito, "MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *IEEE/ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, 2018, pp. 10–20.

[111] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Intl. Conf. on Computer Vision (ICCV)*, 2017, pp. 2961–2969.

[112] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2015, pp. 343–352.

[113] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, "VolumeDeform: Real-time volumetric non-rigid reconstruction," in *Eur. Conf. on Computer Vision (ECCV)*, 2016, pp. 362–379.

[114] A. Concha and J. Civera, "An evaluation of robust cost functions for RGB direct mapping," in *European Conference on Mobile Robots (ECMR)*, 2015, pp. 1–8.

[115] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "MID-Fusion: Octree-based object-level multi-instance dynamic SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 5231–5237.

[116] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss, "ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 7855–7862.

[117] M. Strecke and J. Stückler, "EM-Fusion: Dynamic object-level SLAM with probabilistic data association," in *Intl. Conf. on Computer Vision (ICCV)*, 2019, pp. 5865–5874.

[118] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft mavs," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 5303–5310.

[119] M. Bryson, M. Johnson-Roberson, and S. Sukkarieh, "Airborne smoothing and mapping using vision and inertial sensors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 2037–2042.

[120] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-manifold preintegration for real-time visual–inertial odometry,” *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[121] A. I. Mourikis and S. I. Roumeliotis, “A multi-state constraint Kalman filter for vision-aided inertial navigation,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2007, pp. 3565–3572.

[122] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “Robust visual inertial odometry using a direct ekf-based approach,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 298–304.

[123] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual–inertial odometry using nonlinear optimization,” *Intl. J. of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[124] T. Qin, P. Li, and S. Shen, “VINS-Mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Trans. Robotics*, vol. 34, no. 99, pp. 1–17, 2018.

[125] R. Scona, S. Nobili, Y. R. Petillot, and M. Fallon, “Direct visual SLAM fusing proprioception for a humanoid robot,” *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[126] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, “Robust dense mapping for large-scale dynamic environments,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 7510–7517.

[127] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, “StaticFusion: Background reconstruction for dense RGB-D SLAM in dynamic environments,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 3992–3999.

[128] S. Nobili, R. Scona, M. Caravagna, and M. Fallon, “Overlap-based ICP tuning for robust localization of a humanoid robot,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 4721–4728.