ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Dottorato di Ricerca in
COMPUTER SCIENCE AND ENGINEERING
Ciclo 29°

Settore Concorsuale di afferenza:	09/H1
Settore Scientifico-Disciplinare:	ING-INF/05

SEMANTIC SLAM
A NEW PARADIGM FOR OBJECT RECOGNITION AND SCENE RECONSTRUCTION

Presentata da: Tommaso Cavallari

Coordinatore Dottorato

Chiar.mo Prof. Ing.
Paolo Ciaccia

Relatore

Chiar.mo Prof. Ing.
Luigi Di Stefano

Esame finale anno 2017

Tommaso Cavallari: *Semantic SLAM,* A new paradigm for object recognition and scene reconstruction, © 2016

SUPERVISOR:
Luigi Di Stefano

LOCATION:
Bologna, Italy

ABSTRACT

Simultaneous localisation and mapping (SLAM) is a technique studied in computer vision and robotics that, given measurements obtained from one or more sensors, allows incremental building of a map of the environment (*i.e.* the *mapping* process) and simultaneous estimation of the position and orientation of the very same sensor used to acquire the input data (*i.e.* the *localisation* process). The localisation and mapping phases are typically performed iteratively, back-to-back, in order to allow each to rely on informations provided or updated by the other. SLAM systems can rely on the input provided by different kinds of sensors to estimate the pose of the mobile agent carrying them: works in literature proposed systems deploying laser range finders, inertial or odometry sensors, GPS units, and cameras. The subset of SLAM systems relying on image sensors (either monocular or not) is named visual SLAM. Visual SLAM systems typically allow the generation of accurate reconstructions of the explored environment but, until very recently, did not provide high level informations on the contents of the reconstructed scenes, useful to foster high level reasoning by subsequent algorithms.

In this thesis we focus on the topic of Semantic SLAM, *i.e.* we propose techniques to obtain semantically accurate reconstructions of the explored environment by combining efficient SLAM systems with state-of-the-art semantic image segmentation algorithms. We show how, by relying on such semantic reconstructions, the accuracy of the localisation phase of a SLAM pipeline can improve, by accounting for the presence of semantic informations during the camera pose estimation step. We thus realise a "semantic loop", where the availability of high level clues betters the mapping process, in turn helping the subsequent localisation phase. A full system, drawing inspiration from the presented research, allowing a real-time and automatic semantic mapping of large-scale environments is then presented.

An ancillary, but nevertheless important, component of simultaneous localisation and mapping systems is a technique to allow the estimation of sensor position separately from the main SLAM loop (*i.e.* to recover from failures in the localisation algorithm). We present a technique that, by exploiting the appearance of image patches can reliably localise the likely position of the sensor used to acquire such images. The latter technique associates the likeness of distinctive points in the candidate image with locations in a world coordinate frame where similar points have been observed in the past (*e.g.* by associating image patches depicting chairs with locations in the map where similar chairs are located). A relocalisation system such as the one we present can be easily included in a Semantic SLAM system to allow a more robust mapping process wherein camera tracking failures can be reliably recovered from.

AUTHOR'S PUBLICATIONS

During the PhD period, the author contributed to the following publications. Research conducted during the development of some of them is integral part of this thesis.

[1] Nicholas Brunetto, Samuele Salti, Nicola Fioraio, Tommaso Cavallari and Luigi Di Stefano. 'Fusion of Inertial and Visual Measurements for RGB-D SLAM on Mobile Devices'. In: *International Conference on Computer Vision Workshops*. Dec. 2015, pp. 148–156.

[2] Tommaso Cavallari and Luigi Di Stefano. 'Volume-Based Semantic Labeling with Signed Distance Functions'. In: *Pacific Rim Symposium on Image and Video Technology*. Vol. 8334. 2015, pp. 544–556.

[3] Tommaso Cavallari and Luigi Di Stefano. 'On-line Large Scale Semantic Fusion'. In: *European Conference on Computer Vision Workshops*. 2016.

[4] Tommaso Cavallari and Luigi Di Stefano. 'SemanticFusion: Joint Labeling, Tracking and Mapping'. In: *European Conference on Computer Vision Workshops*. 2016.

[5] Tommaso Cavallari, Stuart Golodetz*, Nick Lord*, Julien Valentin, Luigi Di Stefano and Philip H. S. Torr. 'On-the-Fly Adaptation of Regression Forests for Online Camera Relocalisation'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[6] Federico Tombari, Tommaso Cavallari and Luigi Di Stefano. 'Automatic extraction of pole-like objects using point cloud library'. In: *GIM International* October (2014).

[7] Federico Tombari, Nicola Fioraio, Tommaso Cavallari, Samuele Salti, Alioscia Petrelli and Luigi Di Stefano. 'Automatic detection of pole-like structures in 3D urban environments'. In: *International Conference on Intelligent Robots and Systems*. 2014, pp. 4922–4929.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Prof. Luigi Di Stefano, for all the trust, support, patience, and insightful words of encouragement he gave me in the last few years. Reaching this target would not have been possible without his guidance.

I would also like to thank my family: my mother Patrizia, my father Claudio, my sister Chiara and my grandmother Anna. They have always been there for me and I will never be able to thank them enough.

Another thank goes to current and former members of the Computer Vision and Networks Laboratories of the Faculty of Engineering in Bologna: Samuele Salti, Federico Tombari, Nicola Fioraio, Alioscia Petrelli, Fabio Bruè, Nicholas Brunetto, Alessio Tonioni, Matteo Poggi, Riccardo Spezialetti, Paolo Galeone, Gianluca Berardi, Perluigi Zama-Ramirez, Alessandro Zanna, and Alessandro Pernafini. Our in-depth discussions on philosophical, ethical, political, scientific topics and current events made the last few years memorable. Thanks to Mario and Peter too, for their patience in listening to all kinds of such discussions.

I would also like to thank Prof. Philip H. S. Torr, for allowing me to spend a period in his research group in Oxford and conduct exciting work, that made me grow as a PhD student. Finally, another thank goes to Stuart Golodetz, Nick Lord and all the other members of the Torr Vision Group in Oxford. Their friendship and scientific advice in the months I spent in the United Kingdom was invaluable.

CONTENTS

1

INTRODUCTION

Simultaneous localisation and mapping (SLAM) is a technique studied in computer vision and robotics that, given measurements obtained from one or more sensors, allows incremental building of a map of the environment (*i.e.* the *mapping* process) and simultaneous estimation of the position and orientation of the very same sensor used to acquire the input data (*i.e.* the *localisation* process). Starting from a completely unknown environment, the measurements allow the reconstruction of a first and coarse view of the sensor's surroundings. The position of the subsequent measurements, relative to the initial samples, is then estimated using the information stored in the partial map that has just been built, which is in turn improved by integrating the new data. The SLAM process is repeated iteratively, to increase the map coverage of the environment as well as its accuracy. The precision of the sensor localisation step typically depends on the map quality, since a more detailed reconstruction of the scene provides better clues to identify the position and bearing of the sensor itself, thus creating a positive feedback loop between the *localisation* and *mapping* phases of the algorithm. In Chapter 2 we present the SLAM topic in more detail, by additionally providing a survey on the main research works in the field.

Simultaneous localisation and mapping is a core component for several technologies that are have been studied in the last decades and have started becoming well known to the general public in the last few years. Cars equipped with self (or assisted) driving capabilities, automated household appliances (*e.g.* vacuum cleaners, lawnmowers, . . .) and toys, detailed 3D maps of the world onto smartphones (used for augmented reality or path planning), augmented/virtual reality setups, or even the 3D scanning (and printing) systems: all rely on SLAM algorithms to reconstruct the world around the sensor and perform their task or provide useful informations to the user.

Simultaneous localisation and mapping can be performed with several kinds of sensors, each allowing attainment of different map types. In the computer vision field, predictably, the main instrument used to generate reconstructions of the world is the image sensor. Different techniques have been brought forward during the years, mainly split being between the 2D and 3D category of the input sensor: 2D sensors are the standard monocular RGB (or gray-scale) cameras, whilst techniques relying on the presence of 3D sensors can process input provided by *e.g.* laser scanners, stereo camera setups, or active RGB-D sensors based on structured light or time-of-flight technologies. This thesis work will focus on 3D SLAM performed using commodity RGB-D sensors such as the Microsoft Kinect, the Asus Xtion, or the Structure Sensor (the latter specifically designed for mobile mapping, being self-powered and small enough to be mounted on the back of a tablet device); that in recent years have become extremely affordable and thus amenable to deployment in the hands of end-users.

Current state of the art SLAM algorithms are, thanks to the work by hundreds of researchers and decades of effort, in general able to produce a fairly accurate map of the environment and estimate with sufficient accuracy the sensor position. Generated maps, though, are typically only concerned with the appearance of the world (*i.e.* its geometry), not providing high level informations on *what* is located in a certain area of the explored region. Availability of semantic informations in the maps can be indeed useful for higher level reasoning by the agent using such reconstructions: if we think about a self driving car having to take decisions on how to react to a small obstacle suddenly appearing in its path (and therefore in its continuously updated map of the surroundings), we can easily understand the importance of knowing whether this obstacle is a branch fallen from a tree or a person. A less extreme example, depicting the importance of the availability of semantic informations in the map, can be that of the autonomous mopping robot: by detecting an area of the floor to be washed as covered in carpet (geometrically very similar to other types of flooring, *i.e. flat*), the mobile agent can decide to avoid working on top of it. While still being a largely unexplored area, works dealing with the extraction of semantic information from the observed world and its embedding into the reconstructions generated by SLAM algorithms started appearing recently.

1.1 SEMANTIC SIMULTANEOUS LOCALISATION AND MAPPING

The work presented in this thesis is concerned with the integration of semantic information within the SLAM process. Specifically, we aim to develop a Semantic-SLAM system where information on content of the scene being explored is used to simultaneously improve the accuracy of camera localisation and embed knowledge into the generated map. We strive to show how a virtuous loop can ensue, by storing high level informations on the world in the map and employing such clues, in addition to sensor measurements, during the following camera localisation phases of the algorithm. Similarly as to how the availability of a better map causes a more accurate estimation of the sensor pose, we show that the accuracy of such estimation can be improved by relying on semantic clues. This allows the generation of a more accurate (and semantic) map of the surroundings of the sensor that, in turn, will benefit the subsequent runs of the algorithm.

No constraints on the kind of semantic information employed by our system will be put in place: ideally, we want to be able to classify each element of the images used to perform SLAM (*i.e.* to give a label to every pixel therein). Two research topics tackle this problem: on one hand we have the *object recognition* task, where the aim is to locate and classify certain objects in a scene; on the other we have *semantic segmentation*, whose aim is the subdivision of each image into several regions, each representing a certain category (such as: wall, floor, furniture, etc...). Clearly, a semantic segmentation algorithm is closer to what we would like to feed a Semantic SLAM algorithm (as described in the beginning of this section) but, under certain conditions, it is possible to consider the information provided by the output of an object recognition (or detection) algorithm and give a name (label) only to pixels belonging to the detected objects. Many current state of the art semantic segmentation and object recognition algorithms employ the so called Deep Learning paradigm. Deep Learning algorithms are, in general, based on neural networks made out of several layers, each providing a representation of the input at higher level of generalization, thus helping in the classification task for each sub region of the image fed to the algorithm.

One of our goals is to study the effect of the inclusion of semantic information into the SLAM process. To achieve this goal, as a first step, we need to define a data structure to store such information. For this reason, we propose to augment the map representations generated by SLAM systems with, for example, a label indicating the type of object present in each element of the reconstructions (Chapter 3). Given the great number of object recognition/segmentation algorithms, as a first approximation we will assume to have a correct pixel-wise labelling of the images captured by the RGB-D sensor; only successively we will focus on the selection of a reliable and sufficiently fast algorithm to include in the whole pipeline.

As anticipated, we also aim to exploit semantic information during the camera localisation phase. Purposely, we further augment the map representation to store, instead of a single semantic label per element, a full *probability mass function* for several categories. The camera pose estimation algorithm is then modified to account for the availability of such information, thus improving the quality of generated maps (Chapter 4).

Since state of the art labelling neural networks are (currently) not able to produce results in real time (by real-time here we mean "camera rates", *i.e.* 30 Hz), in Chapter 5 we finally present a SLAM pipeline able to cope with a "slow" semantic segmentation algorithm: by deploying the system on a personal computer with two different graphics processors we are able to achieve interactive processing rates even when the labelling operation takes hundreds of milliseconds.

1.2 APPEARANCE-BASED CAMERA RELOCALISATION

Simultaneous localisation and mapping algorithms iteratively estimate the position and orientation in the world of a sensor and rely on such estimation to augment a map of the environment with new informations. Expectedly, sensor localisation techniques are not flawless, sometimes failing in their task for a plethora of reasons depending on their specific implementation. When this happens, common SLAM systems enter into a so-called *relocalisation* mode, where the current contents of the generated map – possibly in concert with other sources of information – are used to identify likely positions of the sensor in order to restart the localisation and mapping loop.

Relocalisation systems are also useful to bootstrap a SLAM session with a previously acquired map of the environment, rather than starting a reconstruction from scratch. Looking back at the robot vacuum example previously mentioned, one can perceive how it is more efficient to store a map of the floors to clean and, on activation, apply suitable techniques to identify the current pose of the agent and start an optimised cleaning pattern, rather than having to explore the whole house again.

Several camera relocalisation systems rely on the identification, in the images being used to estimate the sensor pose, of parts of the already mapped scene for which the location in the world is known (having been estimated in the past). When correspondences between items observed in the image (for which the sensor pose is unknown) and items endowed with an absolute position in the world are known, it becomes possible to estimate the location and bearings of the sensor used to observe them. Semantic informations are implicitly used to perform such tasks: objects in the scene having distinctive appearance provide useful clues to the camera relocalisation. If a set of chairs is observed by the sensor mapping a household environment, for example, it is much more likely for the agent to be currently located in a sitting room than in a bedroom, therefore simplifying the bootstrap/failure recovery phase of the SLAM algorithm.

In Chapter 6 we present such a system, based on the online adaptation of regression forests, allowing reliable estimation of sensor position and orientation in a previously explored environment, by reliance on image appearance features. Our system provides, for each part of the image processed by it, a set of world locations where similarly appearing items have been previously observed (*e.g.* for an image patch depicting a stair step, the locations of steps previously detected whilst mapping the environment are returned). By deploying such informations in a robust pose estimation method (*i.e.* a RANdom SAmple Consensus-based technique) we are then able to estimate the likely sensor location and bearing thus becoming able to proceed with SLAM iterations.

1.3 SUMMARY OF CONTRIBUTIONS

To summarize, the research work carried out during the development of this thesis focused on implementation and evaluation of a robust Semantic SLAM system, where a state of the art Simultaneous Localisation and Mapping system, based on the processing on RGB-D images, has been augmented to integrate and exploit semantic information, such as the position of objects and the type of scene acquired by the sensor.

In the first part we will be concerned with the core of the SLAM system. We expect our work to have a twofold outcome: on one side the employment of semantic information during the tracking and mapping iterations will help obtain higher quality maps and more accurate camera localisation; an additional contribution is also the generation of semantic maps of the environment that may, in turn, be advantageous to higher level reasoning algorithms. Specifically, in Chapter 2 we will begin with a more formal description of the SLAM task, coupled with a survey of visual SLAM systems and the recent works that started to consider the deployment of semantic informations within the generated maps. In Chapter 3, we will then describe a system allowing the semantic mapping of a scene whilst, in Chapter 4, we will show how to improve the quality of semantic reconstructions by also taking into account higher level informations during the "localisation" part of the pipeline. We are, thus, realising a full semantic loop where such informations are stored in the reconstructed map and in turn relied on to improve the continuation of the SLAM loop. Finally, in Chapter 5 we will tackle some of the shortcomings abstracted away while presenting the joint labelling, tracking and mapping system just mentioned, thereby describing a complete and deployable pipeline.

In the second part of this thesis (Chapter 6) we will then focus on the relocalisation component of a SLAM system, by presenting a camera pose regression forest, relying on appearance-based informations, allowing robust estimation of sensor location in a previously explored environment. Such technique, when embedded into a simultaneous localisation and mapping pipeline, can reliably allow recovery from tracking failure scenarios or bootstrap novel reconstruction sessions within an already explored environment.

Finally, in Chapter 7, we will summarise the contributions of this thesis and draw concluding remarks.

Part I

SEMANTICFUSION: JOINT LABELLING, TRACKING, AND MAPPING

2 SEMANTIC SLAM: INTRODUCTION AND STATE OF THE ART

In this chapter we will, first, briefly describe the topic of Simultaneous Localisation And Mapping, introduced earlier, and discuss its applications. We will then focus on the field of *visual SLAM*, *i.e.* the localisation and reconstruction of scene representations by the employment of RGB sensors (possibly capturing also depth informations) and computer vision techniques. Finally, we will introduce the idea of *semantic SLAM*, drawing a bridge between the topic of SLAM and that of semantic segmentation of images.

The remainder of this first part of the thesis will be concerned with the description of a complete SemanticFusion system, relying on both the aforementioned topics to allow automatic generation of semantic reconstructions of the explored environments, *i.e.* 3D representations of scenes endowed with labels detailing the *kind* of objects being part of the scene.

2.1 SIMULTANEOUS LOCALISATION AND MAPPING

As briefly mentioned in the Introduction, SLAM systems deploy a processing loop concerned with the tasks of *localising* a mobile agent that is exploring the world and *mapping* the explored locations. The localisation and mapping phases are typically performed iteratively, back-to-back, in order to allow each to rely on informations provided or updated by the other.

The idea that performing the estimation of the pose of an agent (*e.g.* a mobile robot) in the world could benefit from being performed together with the identification of positions of fixed landmarks of interest in a map, describing the environment the agent is exploring, can be traced back to works from the late 1980s.

Initially, the tasks of localisation and mapping were studied separately. By applying probabilistic methods to the estimation of landmark positions and spatial relations between them, Smith and Cheeseman [106] and Durrant-Whyte [36] showed that the correlation between multiple observations of different objects of interest, in a single map, indeed increases with the amount of observations used to estimate their locations. Separately, in the same years, research in navigation systems (*i.e.* the *localisation* of robots) was being undertaken with different techniques: Chatila and Laumond [20] and Crowley [28] deploy laser range finders or ultrasonic sensors to measure the position of the agent, while Ayache and Faugeras [2] rely on stereo vision algorithms, and account for sensor errors by combining different measurements via Kalman filtering [67]. The two tasks were cast under the same light at the beginning of the 1990s by Smith, Self and Cheeseman [107] and Leonard and Durrant-Whyte [73] when it was observed that the uncertainties of multiple relative landmark observations were correlated with each other depending on the confidence on the pose estimated for the mobile agent. Accurately estimating the position of a robot and the location of points of interest in the world were found to be part of the same problem and thus, a formal definition of the Simultaneous Localisation and Mapping problem was proposed, and research previously focused on the tasks of *mapping* and *localisation* began to address the SLAM task.

A detailed look at the history of SLAM is given in the two tutorials by Durrant-Whyte and Bailey [4, 37], as well as in the book by Thrun, Burgard and Fox [113], and the chapter by Thrun and Leonard [114]. Such sources cover the classical formulations of the SLAM problem, based on Extended Kalman Filters [61, 108], Rao-Blackwellised Particle Filters [88] and maximum likelihood estimation. Another recent survey, by Cadena et al. [15], details the works presented in the years after the turn of the millennium, the modern formulation of SLAM problem, techniques adopted to tackle such task, and open challenges that will shape future directions of the research in this field.

2.2 VISUAL SLAM

Simultaneous Localisation and Mapping, as described above, does not pose any restriction on the kind of sensors used to estimate the pose of the agent within the world: systems relying on laser range finders [27], inertial sensors [71], odometry sensors [23], GPS units [70], and cameras [32] (or combinations of them [1, 93]) have been proposed.

The subset of SLAM techniques relying on processing of image data only is aptly named *visual SLAM*. In this thesis we propose a Semantic SLAM pipeline based on the availability of input RGB-D streams, therefore in the remainder of this section we will focus on works relevant to our research.

The field of visual SLAM is typically divided between two main approaches, one relying on data captured by monocular image sensors and the other relying on the presence of an additional depth channel, acquired *e.g.* by stereo or RGB-D sensors, providing information on the geometry of the observed scene.

From the former category, employing RGB informations only to generate maps of an environment wherein a monocular camera is moved freely by the user, we would like to point out the seminal work by Davison [31], followed by [32]. Eade and Drummond [38] and Civera et al. [26] also proposed systems relying on filtering approaches to perform visual odometry. All of those systems rely on sparse features and therefore output *sparse* maps of the explored environment, where no knowledge on the geometry of the observed scene is embedded.

Allowing instead the generation of *dense* maps of the environment, are the works by Newcombe and Davison [89], [92], Keller et al. [68] and Whelan et al. [121].

With the advent of affordable RGB-D sensors such as the Microsoft Kinect, research focused on the reconstruction of environments through employment of 2.5D images (*i.e.* images with an associated depth channel) spurred. Several works relied on the detection of 2D point features (such as SIFT [79], SURF [8], or ORB [99]) in the input images, their matching with keypoints extracted from previously observed frames (or *keyframes*), and the subsequent projection of such interest points in 3D space by using the known spatial relation between the RGB and the depth sensor to estimate camera poses and generate maps of the scene. Examples of such systems are by Endres et al. [40], Fioraio and Di Stefano [43], Henry et al. [59] and Mur-Artal, Montiel and Tardós [86].

A second strand of research focused on the generation of *dense* maps of the environment by RGB-D images, has been fostered by the presentation of the well known KinectFusion system by Newcombe et al. [91]. KinectFusion has been one of the first SLAM pipelines allowing the tracking of a sensor and the extraction of a detailed reconstruction of the scene in *real-time*, by offloading most of its processing to a GPU-based accelerator. The system relies on a *Signed Distance Function*-based map of the environment (that will be described in detail in Chapter 3) allowing, among other results, the attainment of synthetic depth images from arbitrary viewpoints. Such generated depth maps are used to perform camera pose estimation by employing a *Projective Iterative Closest Points* technique [9, 100]. The work we present in this thesis is built on this system and its extensions that will be mentioned in the following paragraphs.

Newcombe's work [91], well-favoured in its simplicity and effective in its core task – mapping of small/medium size environments rich in geometric structure – suffers though from four main shortcomings highlighted as follows.

(a) Reliance on a memory-demanding data structure allocated on the GPU, such as a dense fixed-size voxel grid storing the aforementioned Signed Distance Function, prevents the mapping of large scale scenes.

(b) The drift error, inherently accumulated during the ICP-based tracking process, may cause gross reconstruction errors when observing surfaces poor in geometric informations (*e.g.* smooth/flat surfaces) or closing camera path loops.

(c) The requirement for static and rigid scenes hampers usability outside controlled settings, severely limiting the breadth of practical applications.

(d) Finally, the original KinectFusion outputs a purely geometric map of the environment, thus not providing semantic hints to support higher level reasoning about the surroundings of the sensor. Only a few works embed semantic informations in the reconstructed map and, currently, none rely on them during the camera localisation phase of the pipeline.

In the remainder of this section we will mention some works tackling each of the aforementioned issues, excluding the last one, concerned with semantic informations. The latter, being core of the work of this thesis, will be treated in the next section.

EXTENTS OF THE MAPPED ENVIRONMENT Different kinds of approaches can be taken to extend the mapped workspace. The data structure holding the map representation is typically associated to a global coordinate frame allowing the generation of metrically accurate reconstructions. Such reference frame for the volume can be moved alongside with sensor movements, thus centering the map onto the position of the camera, possibly downloading parts of the voxel grid from GPU to CPU memory, as proposed by Roth and Vona [98] and Whelan et al. [120]. Building on this idea are also the works by Fioraio et al. [44] and Kähler, Prisacariu and Murray [64]: in their proposals, the map of the scene is divided in smaller volumes, reconstructed independently and merged with suitable techniques. Alternatively, Kähler* et al. [66] and Nießner et al. [94] employ sparse voxel grids indexed by hash tables (or as hierarchical structures: Chen, Bautembach and Izadi [21] and Zeng et al. [124]) to diminish occupancy and increase the bounds of the mappable space. Hybrid methods combining the strengths of hierarchical structures and hash tables have also been proposed to allow the generation of reconstructions with variable degrees of accuracy, depending on the level of details of the observed scenes [65].

TRACKING ACCURACY Some works have proposed to rely on additional clues with respect to the purely geometry-based KinectFusion camera tracker, to attempt reducing the inherent drift error. In particular, Bylow, Olsson and Kahl [13] show that injecting per-pixel colour measurements within the cost function optimized by the tracker does improve accuracy, especially when dealing with flat or smooth surfaces that present distinctive colour features. Conversely, Zhou and Koltun [127] posit to deploy occluding contours rather than colour and demonstrate the effectiveness of their proposal in experiments focused on scanning individual objects featuring smooth and evenly coloured surfaces. Other approaches concern tackling drift by a global optimization step on a pose graph, which may better counteract misalignments showing up at loop closures. The pose optimization step may be run either off-line, after the capture process is terminated [126, 128], or on-line, by continuously optimizing the poses of partial volumes, possibly deforming the surfaces contained therein [30, 44, 58, 64].

MOVEMENT IN THE SCENE Lightly dynamic scenes are allowed by the original KinectFusion algorithm, though only due to the volume update step behaving as a low-pass filter forgetting old measurements after the fusion of a certain number of frames. However, scene motion is not explicitly considered, sudden movements of relatively large-size objects inevitably causing camera tracking failures. Recent works, instead, have demonstrated impressive results in non-rigid and dynamic settings by modelling motion as a fundamental property of the captured scene [34, 35, 90].

2.3 SEMANTIC SLAM

As mentioned in the previous section, embedding of semantic informations into SLAM algorithms was addressed by just a few works. A relevant early proposal in this field is the work by Castle, Klein and Murray [19], where locations of planar objects detected by SIFT features are incorporated into a SLAM algorithm based on Extended Kalman Filtering [61, 108]. Later, Civera et al. [25], lift the limitation of the previous approach to account for planar objects only, by building their system on top of the EKF SLAM pipeline of [26], detecting objects of interest via SURF keypoints and descriptors [8], and inserting corresponding 3D points into the filter state. Bao et al. [5–7] proposed the idea of "Semantic Structure from Motion" to jointly perform the object recognition and SLAM tasks; in their research, tough, they process entire image sequences offline and perform a global optimization on the resulting environmental map. All such approaches, also, do not employ RGB-D informations, relying instead on the processing of several images to estimate the 3D world structure.

On the converse, works exploiting the availability 3D information throughout the entire pipeline are those by Fioraio, Cerri and Di Stefano [41], [42], Salas-Moreno et al. [101] and Xiao, Owens and Torralba [123]. Fioraio and Di Stefano, in [42], propose a keyframe-based SLAM algorithm where detected objects are inserted as additional constraints in the bundle adjustment process used to estimate camera poses. Their work is then extended in [41] by deploying the Semantic Bundle Adjustment framework together with KinectFusion [91]. This enables the detection of known object instances, estimation of their 6 DOF poses, and embedding of such objects into a globally optimized pose graph to help counteracting drift. Although pursuing a diverse graph-based mapping strategy, the SLAM++ system by Salas-Moreno et al. relies on the detection of known object instances to perform camera tracking by an approach akin to KinectFusion: detected objects are used to estimate the sensor location, by rendering a synthetic view of their placement and aligning the real depth image to such view through the ICP algorithm [100]. Xiao, Owens and Torralba introduce a semantically annotated dataset; while not the main focus of their work, semantic informations on the object location are used during the bundle adjustment process to better constrain the generated reconstruction of the environment. In their work they show a full "semantic loop", where bounding boxes for objects manually labelled in a subset of frames are used to improve the world map; in turn this allows to propagate their labels to previously unlabelled frames thus reducing the effort needed by the user to annotate the entire sequences.

Additional recent works, concerned with the generation of semantically annotated reconstructions, are those by Hermans, Floros and Leibe [60], Valentin et al. [117], Golodetz et al. [52], Miksik et al. [84], and McCormac et al. [83].

Hermans, Floros and Leibe [60] present a pipeline allowing the transfer of semantic labels associated to 2D images by *randomised decision forests* to 3D point cloud reconstructions obtained by adapting the method described in [46]. In their system, the 2D semantic segmentations are further refined with a 3D Conditional Random Field before being stored as per-point state in the final reconstruction. Golodetz et al. [52] and Valentin et al. [117] show a system based on the InfiniTAM 3D reconstruction pipeline [66] that, employing multi-modal user interaction, can learn to classify voxels of the reconstructed map into user selected categories via *streaming decision forests* trained on-line. Miksik et al. [84] deploy a setup based on a head-mounted stereo camera together with the VoxelHashing 3D reconstruction pipeline [94]. By tracking the target of a portable laser pointer through the acquired frames, the user is able to mark areas of the scene as pertaining to a certain object category. Such labels are then fed to a densely connected Conditional Random Field that learns how to classify voxels online in the reconstructed scene.

The systems described until this point rely on semantic segmentations obtained either with the deployment of decision forests or conditional random fields. Thanks to the focus on deep learning in the last years, several semantic segmentation techniques were proposed that could process entire images in fractions of a second, providing pixel-wise category labels or probability mass functions over a set of such categories. Gupta et al. [54], process pairs of RGB and Depth images with multiple deep neural networks followed by an SVM classifier, generating a threefold output: bounding boxes for object detection, per-pixel confidences to segment such instances and a full-image semantic segmentation output. Long, Shelhamer and Darrell [77] show how Fully Convolutional Networks can provide accurate per-pixel, per-category scores on entire images in a deterministic amount of time. Eigen and Fergus [39] demonstrate how a single deep network architecture can successfully be employed for three different tasks: predicting depth and normals from RGB images as well as performing semantic segmentation to infer, again, per-pixel category probabilities. Zheng et al. [125], then, join the strengths of Conditional Random fields and Convolutional Neural Networks within a unique framework trained end-to-end to obtain semantic segmentation.

Similarly to our proposals, described in Chapter 3 and Chapter 5, the very recent work by McCormac et al. [83] combines the output of a Convolutional Neural Network for Semantic Segmentation with the ElasticFusion SLAM system [121] to obtain semantically annotated 3D maps, by fusing multiple predictions associated with different frames of the input RGB-D stream.

The next chapter will describe a first extension of the KinectFusion system [91] allowing the generation of semantic reconstructions of the explored scene by the integration of the output of a Fully Convolutional Network [77].

We will build on the ideas there gathered in the following chapter, Chapter 4, where we will show how the issue of the inherent drift that may be accumulated by a purely geometric tracker can be tackled by relying on semantic observations such as category labels rather than low-level cues like colour [13] or occluding contours [127] during the camera pose estimation step. We realize a closed loop between the semantic labelling and the dense mapping and tracking processes, so to allow one process to beneficially influence the other. In this respect, our approach is more similar in spirit to previous works aimed at creating synergistic interactions between object detection and SLAM [41, 42, 101].

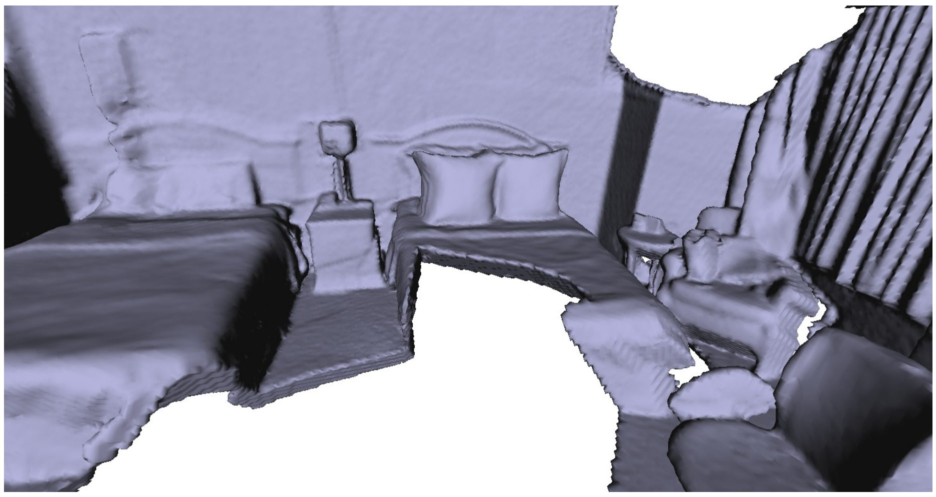
Additionally, while previous work [52, 84, 117] has addressed interactive scenarios and shown how to obtain semantic reconstructions by placing the user in the loop, we propose a fully automatic approach: all the pipelines that we present do not require any user interaction to perform the labelling and train the classifier, and, therefore, any untrained user can proficiently reconstruct the explored scenes just by moving around a hand-held RGB-D sensor. Finally, improving on the automatic volume labelling method reported in Chapter 3, and unlike all the above mentioned works addressing semantic reconstruction, our final proposal, described in Chapter 5, is not constrained to keeping track of only one label per element of the reconstruction but instead can gather evidence concerning all categories across the whole map. The proposed pipeline will yield, in each spatial location, the full probability mass function across several categories rather than estimating the most likely label only. Such a richer output enables not only generation of semantically labelled maps, but also assessment of the likeliness of each and every category across the whole scene surface.

3 VOLUME-BASED SEMANTIC LABELLING

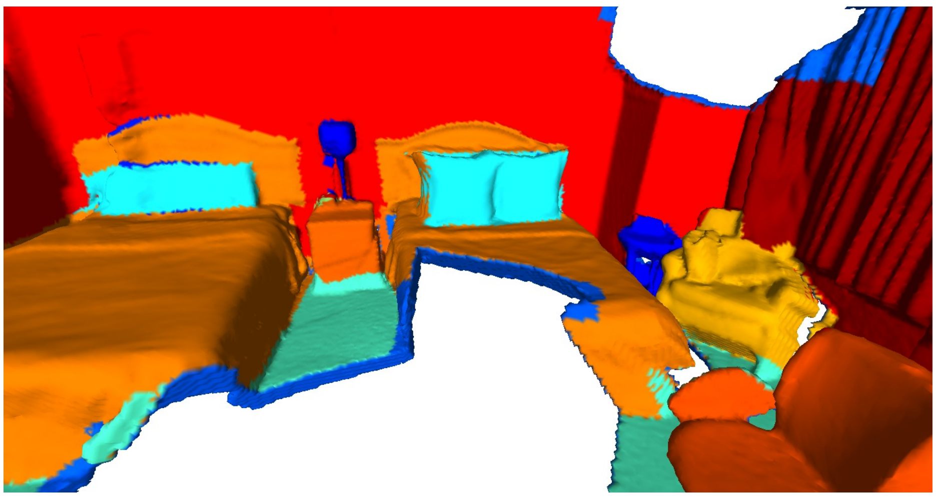
3.1 SUMMARY OF CONTRIBUTIONS

Research works on the two topics of Semantic Segmentation and SLAM (simultaneous localisation and mapping) have been following separate tracks. In this chapter, we link them quite tightly by delineating a category label fusion technique that allows for embedding semantic information into the dense map created by a volume-based SLAM algorithm such as KinectFusion. The approach here described allows the generation of a semantically labelled dense reconstruction of the environment around the sensor from a stream of RGB-D images.

It is thanks to the employment of semantic reconstructions akin to those described in the following pages that, in a later chapter, we will show how the exploitation of semantic informations can improve the accuracy of the camera pose estimation (*i.e.* the *localisation* step). We evaluate the proposed volumetric labelling approach by using a publicly available, semantically annotated, RGB-D dataset to generate fully labelled reconstruction in the following scenarios: a) employing ground truth labels, b) corrupting such annotations with synthetic noise, c) deploying a state of the art semantic segmentation algorithm based on Convolutional Neural Networks. Results show that the proposed technique outputs reasonably accurate, fully annotated, environment maps. Additionally, thanks to the temporal smoothing effect naturally provided by this approach, the inevitable labelling failures that take place during processing of single frames in the RGB-D input stream have a reduced chance of hindering the final reconstructions.



(a)



(b)

Figure 3.1: The top picture shows a shaded visualization of the standard KinectFusion output on the hotel scene of the Sun3D dataset [123], note that geometric informations only are used to generate such image. The bottom picture illustrates the type of output delivered by our technique: a fully labelled, dense reconstruction wherein each surface element is assigned a category tag, here colour coded for ease of visualization.

3.2 RATIONALE

In the last years, the Computer Vision community renewed its interest in the task of Simultaneous Localisation and Mapping by leveraging on RGB-D information. This research trend has been fostered by the development of ever cheaper sensors as well as by the more and more ubiquitous presence of smart mobile platforms, possibly having such sensors on board. Many works tackled issues related to reliable camera tracking, accurate mapping, scalable world representation, efficient sensor relocalisation, loop closure detection, map optimization. A major breakthrough in the realm of RGB-D SLAM was achieved by the KinectFusion algorithm by Newcombe et al. [91], which firstly demonstrated real-time and accurate dense surface mapping and camera tracking.

On separate tracks, researchers working on object detection and semantic segmentation proposed many interesting techniques to extract high-level knowledge from images by recognition of object instances or categories and subsequent region labelling Especially thanks to the recent developments in the field of deep convolutional neural networks, year after year, new benchmark-beating algorithms are proposed that enable to quickly process raw images and extract from them valuable semantic information.

However, just a few works have tried to draw a bridge between the two aforementioned fields, though we believe that both research areas could benefit significantly from tighter integration. Indeed, a SLAM process may be improved by deploying high-level knowledge on the type of objects encountered while the moving agent explores the environment, whereas object detection and semantic labelling techniques could be ameliorated by deploying multiple views from tracked sensor poses.

In this chapter we propose a technique capable to obtain incrementally a dense semantic labelling of the environment from a stream of RGB-D images while performing tracking and mapping *à la KinectFusion* [91]. Therefore, differently from the map concerned only with the 3D shape of the surfaces present in the environment, yielded by a typical SLAM algorithm such as KinectFusion (and depicted in Figure 3.1a), our technique additionally provides a fully labelled map that embodies the information on *what* kind of object (*e.g.* a wall, chair, bed, pillow, furniture...) each reconstructed surface element belongs to. A view from one of such dense semantic maps is reported in Figure 3.1b, with each colour representing a different category label.

The remainder of the current chapter is structured as follows: Section 3.3 describes, first, the camera tracking and mapping method employed in our work and, subsequently, illustrates our proposal concerning the integration of a semantic labelling algorithm's output within the SLAM framework; finally, Section 3.4 shows how the proposed volume-based semantic labelling technique behaves when feeding it with *a*) "correct", manually annotated, labels, *b*) labels corrupted by synthetic noise, *c*) "real" labels obtained by a state of the art semantic segmentation algorithm.

3.3 DESCRIPTION OF THE METHOD

To obtain a densely labelled map of the environment captured by the sensor, we adopt a dense, volume-based, approach. Similarly to KinectFusion [91], the map is represented by a regular 3D grid of fixed size, wherein each element (*voxel*) stores a value defining a Signed Distance Function [29], but, peculiarly, we also provide each voxel with a label specifying the type of object appearing in that spatial location, together with an indication of the confidence on the assigned label. In the remainder of this section we will, first, describe the Signed Distance Function data structure (Subsection 3.3.1), as employed in the original KinectFusion pipeline [91] to generate accurate reconstructions of the explored environments and, subsequently, our proposed approach to achieve integration of semantic labels into such representation (Subsection 3.3.2).

To update the information stored into the voxel grid by integrating new measurements, we need to track the RGB-D sensor as it moves within the environment. In KinectFusion [91], camera tracking is performed by ICP-based alignment between the surface associated with the current depth image and that extracted from the TSDF. Later, Bylow et al. [14] and Canelhas, Stoyanov and Lilienthal [17] proposed to track the camera by direct alignment of the current depth image to the mapped environment encoded into the TSDF as the zero-level isosurface. This newer approach has been proven to be faster and more accurate than the original KinectFusion tracker.

Camera tracking by direct alignment of the acquired depth image with respect to the TSDF volume relies on the following consideration: assuming that the estimated pose of the camera is correct and noise does not affect the current and previous depth measurements, then each depth pixel should correspond to the projection of a 3D world point that, given the content of the TSDF grid, features a null distance function (i.e. lies on the surface). Clearly, noiseless depth images and perfectly accurate camera pose estimations cannot be obtained in real settings. However, Bylow et al. [14] show that, for small camera movements (such as those occurring to the sensor during online real-time tracking), iteratively minimizing the sum of the squared TSDF values determined by the 3D points corresponding to depth pixels in the RGB-D image, allows for accurate estimation of camera poses. In our work, we decided to employ the aforementioned direct camera-tracking method on such considerations of speed and accuracy. As no source code for the algorithm has been made available by the authors, we implemented the algorithm according to its description in the original paper. More precisely, our code has been obtained by properly modifying a publicly available implementation of the standard KinectFusion algorithm[1] in order to introduce both the direct camera tracking method as well as the dense semantic labelling process.

For details on the camera pose estimation algorithm, we refer the reader to the previously mentioned article by Bylow et al. [14] and Chapter 4, where we show how the results obtained by applying such technique can be improved when taking into account the presence of semantic labels in both the reconstruction and input RGB-D frames, thus increasing the accuracy of the estimated camera poses.

3.3.1 *Data structure for 3D reconstructions*

Core element of any SLAM system is the map of the environment being explored. This map is relied on by the localisation component to allow the estimation of the sensor pose from which new measure-

1 https://github.com/Nerei/kinfu_remake

ments are acquired. As the localisation and mapping loop proceeds, the map is in turn updated, to account for the new data acquired by the sensor.

As previously anticipated, in a visual SLAM system such as KinectFusion [91], base of the pipeline described in this thesis work, the reconstruction of the environment is defined via a Truncated Signed Distance Function (TSDF), popularised by Curless and Levoy [29].

A Signed Distance Function (SDF) is a continuous function $\Phi(\mathbf{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ defining, for each point in a 3D space, the signed distance of the point to its closest surface. SDF values are positive when the point is located *outside* of a surface (in front of it) and negative otherwise (*i.e.* the point is *behind* the surface represented by the function). A point located exactly on a surface is assigned an SDF value of 0. To clarify, Figure 3.2 shows a colour-coded representation of the SDF values associated to a 2D shape (*i.e.* the corresponding distance transform). The extension to 3D surfaces is straightforward.

A Signed Distance Function can be used to *indirectly* define the map of the environment being explored. Since each position in space is associated to the distance from its closest mapped surface, it is trivially possible to identify empty and occupied areas: positive values of the function identify free space, while areas where the SDF assumes negative values belong to the interior of an object. Extraction of a representation of the mapped area is less trivial but nonetheless feasible: by employing suitable techniques, it is possible to determine the zero-level isosurface of the function and either render synthetic images from arbitrary viewpoints or generate a triangular mesh that can be further processed according to the task at hand. More details on this in the following subsections.

Values of the SDF are, in the KinectFusion pipeline [91], truncated up to a small value δ (typically in the order of a few centimetres) to represent uncertainty on the surface being reconstructed: points in the world located farther away from the object boundaries than the *truncation distance* δ, are assigned the SDF value of δ to indicate that, when observed from different camera viewpoints, they might result closer to other surfaces. Similarly, points pertaining to the interior of an object have SDF values up to a distance of $-\delta$ from the surface. Signed distances assigned to points located farther away, that would have increasing negative SDF values, are left uninitialised: while

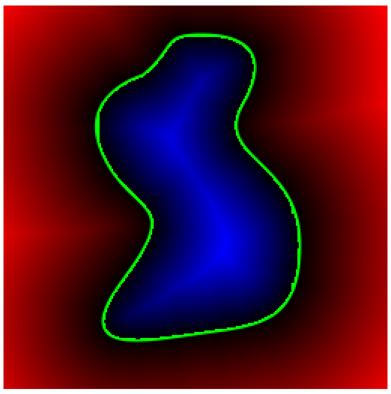


Figure 3.2: Representation of the 2D Signed Distance Function for a curved shape. The green path represents the surface encoded by the SDF. The gradients represent the corresponding values: black-to-blue indicates increasing negative distances from the surface (*i.e.* points *inside* the contour) while black-to-red represents increasing positive distances (*i.e.* points located *outside* the shape).

from a specific camera viewpoint they might be believed to be part of the interior of an object, when viewed from other vantage points they might indeed result located *outside* such object of interest (*i.e.* behind it). A SDF to which is applied the just-described truncation distance is named Truncated Signed Distance Function (TSDF, $\phi(\mathbf{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}$) and is used to define the map of the environment being reconstructed.

Truncated Signed Distance Functions are, conceptually, continuously defined over the whole $\mathbb{R}^3$ domain. In practice, in KinectFusion-based reconstruction pipelines, TSDFs are approximated through the definition of a regular voxel grid storing values of the function in discrete spatial locations. Due to practical memory occupation concerns, the domain of this discretised TSDF is also bounded, thus allowing the reconstruction of an environment of limited size. The mapping between values of the continuous function $\phi(\mathbf{x})$ to the discretised version $\tilde{\phi}(\mathbf{x})$ can be trivially obtained by sampling the values of ϕ corresponding to the spatial locations of the centre of each voxel. To, on the other hand, obtain values of ϕ for arbitrary locations in space, typically, a trilinear interpolation of the values of $\tilde{\phi}$ corresponding to the 8 voxels closest to the location of interest is required. A discretised representation of the TSDF is well suited to GPU-based parallel processing and, by choosing a sufficiently small voxel size, the loss in accuracy derived from the non bijective mapping between ϕ and $\tilde{\phi}$ can be minimised.

In the remainder of this thesis we will use the terms Truncated Signed Distance Function, TSDF, and the symbol ϕ to refer to the continuous definition of the function, assuming the mapping between formal definition of ϕ and implementation, based on the *discretised*, $\tilde{\phi}$ to be applied as necessary.

A second function, $\Omega(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}_0^+$ is similarly defined over the mapped space to store a weight for each voxel. Such value is used in the integration process to *blend* past measurements already stored in the TSDF grid with new depth values provided by the sensor. A weight of 0 indicates locations for which the Truncated Signed Distance Function is currently uninitialised, *e.g.* as previously mentioned, points located behind a surface, farther away than the truncation distance δ.

If desired, each voxel in the grid can be extended to store an RGB triplet and a corresponding weight. This would allow the generation of a coloured reconstruction of the environment. By applying the considerations made in the previous paragraphs on the continuous/discrete conversions, we can define a function, $\psi(\mathbf{x}) : \mathbb{R}^3 \to [0, 255]^3$, to represent the colour associated to each point in the world and a weight function, $\Omega_c(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}_0^+$, again used during the integration step.

The next subsections will detail, first, the process through which the surface measurements encoded within a frame (acquired with, *e.g.* a RGB-D sensor) can be robustly integrated in the 3D voxel grid; subsequently, the opposite operations, allowing the generation of synthetic depth images from arbitrary viewpoints and the reconstruction of a 3D mesh representing the mapped environment in its entirety will be described.

3.3.1.1 *Integration of depth frames*

In a 3D SLAM pipeline relying on an *indirect* map representation such as a TSDF, core of the mapping phase is the update of the function values according to the measurements acquired by the sensor. RGB-D sensors, such as the Kinect, produce a colour and depth frame pair for each sampling interval. Assuming the camera pose in the world to be known (or having been correctly estimated during the localisation phase), a depth frame can be integrated (or *fused*, hence the name KinectFusion [91]) in the 3D voxel grid to improve the current representation of the world.

We denote with $T_{w,c} = (R, \mathbf{t}) \in \mathbf{SE}(3)$ the estimated camera pose, *i.e.* the rigid-body transformation mapping points in the camera reference frame to world coordinates, and $\mathbf{u} = (u, v)^T$ the location of a pixel in the input frame. We assume the depth frame $d(\mathbf{u}) : [0, w) \times [0, h) \rightarrow \mathbb{R}_0^+$ (with w and h indicating, respectively, width and height of the image) to have been previously undistorted thus encoding, for each pixel, the metric distance of the camera to the observed surface, if available (0 otherwise). For simplicity we also assume the RGB frame $i(\mathbf{u}) : [0, w) \times [0, h) \rightarrow [0, 255]^3$ to be registered with the depth image, thus being able to sample the colour associated to a depth measurement $d(\bar{\mathbf{u}})$ by observing the corresponding value of the RGB frame $i(\bar{\mathbf{u}})$. The intrinsic parameters of the depth sensor (focal distances f_x, f_y, and principal point c_x, c_y) are also assumed to be known, so as to be able to perform perspective projection of 3D points into pixels and vice-versa. We will indicate the projection operation of a 3D point onto the image plane as $\pi(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ and the backprojection of a 2D pixel associated to a depth onto a 3D point as $\pi^{-1}(\mathbf{u}, z) : (\mathbb{R}^2, \mathbb{R}) \rightarrow \mathbb{R}^3$. Formally:

$$\pi(\mathbf{p}) = \begin{bmatrix} f_x \frac{p_x}{p_z} + c_x \\ f_y \frac{p_y}{p_z} + c_y \end{bmatrix} \tag{3.1a}$$

$$\pi^{-1}(\mathbf{u}, z) = \pi^{-1}(u, v, z) = \begin{bmatrix} z \frac{u - c_x}{f_x} \\ z \frac{v - c_y}{f_y} \\ z \end{bmatrix} \tag{3.1b}$$

Fusion of the RGB-D frame is then performed as follows. Typically, every voxel in the 3D volume is visited in parallel on the GPU (as the operations performed for each are independent from the others) and the 3D position of its centre $\mathbf{p_v}$ is projected onto the depth map to select the coordinates of a pixel $\mathbf{u_v}$:

$$\mathbf{u_v} = \lfloor \pi(T_{w,c}^{-1} \mathbf{p_v}) \rceil \tag{3.2}$$

The real coordinates resulting from the application of the $\pi(\mathbf{x})$ operation are rounded to the nearest integer ($\lfloor . \rceil$) in order to sample from the input depth frame. Then, *iff* the projected location falls within the bounds of the input image and the corresponding depth measurement is available ($d(\mathbf{u_v}) > 0$), the data contained in the voxel has the chance of being updated. A signed distance from the voxel centre to the surface described by the depth image is computed, by considering the z coordinate of the voxel centre in the camera reference frame:

$$s_v = d(\mathbf{u_v}) - (T_{w,c}^{-1}\mathbf{p_v})_z \tag{3.3}$$

The truncation distance, δ, is then considered, so that voxels in the 3D volume are updated *iff* they lie within the sensor view frustum and $s_v \geqslant -\delta$. The (normalised) truncated signed distance $\hat{s}_v$ for the voxel is then defined as:

$$\hat{s}_v = \begin{cases} 1 & \text{if } s_v \geqslant \delta \\ \frac{s_v}{\delta} & \text{if } -\delta \leqslant s_v < \delta \end{cases} \tag{3.4}$$

The voxel TSDF value, $\phi(\mathbf{p_v})$, is updated by a weighted average with s_v. The weight $\Omega(\mathbf{p_v})$ is also increased, up to a maximum value Ω_{max}. Infinite weight growth is avoided to allow for temporal smoothing of the estimated surface distance: this approach allows older measurements to be forgotten after a certain number of volume updating steps. To this purpose, a function $w(x) : \mathbb{R} \to \mathbb{R}$ is defined so as to apply a weight to the TSDF value $\hat{s}_v$ considered during the averaging process. Several formulations have been proposed in literature, in our experiments we employed, firstly, the exponential approach described by Bylow et al. [14] since in their evaluation it was deemed to produce more accurate reconstructions:

$$w(x) = \begin{cases} 1 & \text{if } x \geqslant \epsilon \\ e^{-\sigma(x-\epsilon)^2} & \text{if } \epsilon \geqslant x \geqslant -\delta \\ 0 & \text{if } x < -\delta \end{cases} \tag{3.5}$$

with ϵ being typically a small negative value, such as $-2.5\,\mathrm{cm}$. The rationale behind such choice of weighting function is to provide high weight to voxels in front of the observed surface, while signalling uncertainty as to the content of voxels located behind the surface by using decreasing weights. After several experiments we decided to deploy instead the constant weighting function $w(x) = 1$ giving equal certainty to both voxels in front of the surface and behind it, as far as the truncation distance. Reconstructions were not severely hindered by this choice and, conversely, the processing speed of the system increased sensibly (due to the removal of several floating point operations performed on the GPU for every voxel being updated). To represent the weight associated to a voxel being updated we define:

$$w_v = 1 \tag{3.6}$$

Finally, Equation 3.4 and Equation 3.6 allow us to define the mechanism used to update the TSDF volume:

$$\phi_{t+1}(\mathbf{p_v}) = \frac{\phi_t(\mathbf{p_v})\Omega_t(\mathbf{p_v}) + \hat{s}_v w_v}{\Omega_t(\mathbf{p_v}) + w_v} \tag{3.7a}$$

$$\Omega_{t+1}(\mathbf{p_v}) = \min(\Omega_t(\mathbf{p_v}) + w_v, \Omega_{max}) \tag{3.7b}$$

The subscripts $_t$ and $_{t+1}$ indicate respectively the current and updated values of the $\phi(\mathbf{x})$ and $\Omega(\mathbf{x})$ functions.

3.3.1.2 *Integration of RGB frames*

The fusion of colour measurements in the voxel grid is an optional step, performed simultaneously to the integration of depth measurements. If processing speed of the pipeline or GPU memory occupation are a concern for the specific task being performed, a KinectFusion system can still work in absence of colour informations.

As mentioned in the previous subsection, coloured pixels in the RGB-D pair are registered to the depth image pixels thus, whilst sampling the depth $d(\mathbf{u_v})$, we can also sample the corresponding RGB triplet:

$$(r_v, g_v, b_v) = i(\mathbf{u_v}) \tag{3.8}$$

Then, *iff* the voxel being updated lies within the truncation distance of the surface being observed (*i.e.* $|\hat{s}_v| \leqslant \delta$), the values stored within it and defining the $\psi(\mathbf{x})$ and $\Omega_c(\mathbf{x})$ functions are updated.

Similarly to the depth fusion step, a weighted average of the pixel colours and those stored in the voxel is performed [13]. A weighting coefficient $w_{c,v}$ depending on the angle θ, between the optical axis of the sensor and the line joining the camera centre to the 3D point determined by the depth measurement for pixel $\mathbf{u_v}$, is defined:

$$w_{c,v} = w_v \cos(\theta) \tag{3.9}$$

The update is performed as follows:

$$\psi_{r,t+1}(\mathbf{p_v}) = \frac{\psi_{r,t}(\mathbf{p_v})\Omega_{c,t}(\mathbf{p_v}) + r_v w_{c,v}}{\Omega_{c,t}(\mathbf{p_v}) + w_{c,v}} \tag{3.10a}$$

$$\psi_{g,t+1}(\mathbf{p_v}) = \frac{\psi_{g,t}(\mathbf{p_v})\Omega_{c,t}(\mathbf{p_v}) + g_v w_{c,v}}{\Omega_{c,t}(\mathbf{p_v}) + w_{c,v}} \tag{3.10b}$$

$$\psi_{b,t+1}(\mathbf{p_v}) = \frac{\psi_{b,t}(\mathbf{p_v})\Omega_{c,t}(\mathbf{p_v}) + b_v w_{c,v}}{\Omega_{c,t}(\mathbf{p_v}) + w_{c,v}} \tag{3.10c}$$

$$\Omega_{c,t+1}(\mathbf{p_v}) = \min(\Omega_{c,t}(\mathbf{p_v}) + w_{c,v}, \Omega_{c,max}) \tag{3.10d}$$

3.3.1.3 *A note on memory requirements*

SLAM systems relying on fixed-size voxel grids as backing storage for the map, such as KinectFusion [91] and derived pipelines as the one described in this and following chapters, have very strong memory requirements. As mentioned towards the end of Subsection 3.3.1, a continuous TSDF is approximated by a dense voxel grid, wherein each voxel holds the TSDF information (and related values, *i.e.* weights and colours) for a certain region of space. The contents of each voxel, as described, are the following:

1. Truncated signed distance of the voxel from the closest world surface.

2. Weight used to compute the running average during the voxel update step.

3. RGB values representing the colour of the surface

4. Weight associated to the colour values.

The TSDF value and its associated weight can be stored as half precision floating point numbers, occupying 2 B each, whilst RGB triplets are stored as *unsigned chars*, and their associated weight as a fourth byte (floating point numbers are scaled to integers by multiplication for a constant scaling factor, *i.e.* 10). The total memory occupation for a single voxel, with such representation, is then 8 B.

Typical voxel grids have size 512^3 or 256^3, even though the latter is used only in cases where the amount of available memory is limited. With a voxel grid of size 512^3 this means a occupation of 1 GB of GPU memory, while with a 256^3 voxel grid this translates to a 128 MB requirement, both well within the memory limit of current GPU processors.

Extents of the mappable area are directly related to the dimension of each voxel (their *resolution*), the actual amount of occupied or free space does not have any effect on it: voxels located far away from the reconstructed objects, whilst uninformative to the purpose of the reconstruction, will be assigned a TSDF value equal to the truncation distance δ. The resolution is typically chosen accordingly to the kind of scene being reconstructed: to reconstruct small objects with many details, a voxel with a side of 2.5 mm could be indicated, thus deploying a cubic volume having sides of $\approx$1.3 m. Slightly larger environments could benefit from 0.5^3cm^3 or 1^3cm^3 voxels, thus allowing the reconstruction of environments respectively as big as $\approx 2.5^3\text{m}^3$ or $\approx 5^3\text{m}^3$. For even larger reconstructions, *e.g.* for room-sized scenes, voxels having sides of up to 2 cm can be used, thus allowing the reconstruction of scenes up to $\approx 10^3\text{m}^3$. The loss in accuracy caused by choosing voxels with sides bigger than 2 cm is too severe to allow deployment of the system in larger environments.

In this and the following chapter we deploy TSDF volumes composed of 512^3 voxels with sizes ranging between 1.5^3 and 2^3cm^3, depending on the extents of the specific scene to reconstruct. If one wants to map a larger environment, different techniques have to be deployed: the dense voxel grid does not scale further. A non-dense approach, where voxels in GPU memory are stored only for locations close to the actual surfaces being reconstructed, thus maximising the memory efficiency, is described in Chapter 5 and is based on dynamic indexing structures that allow on-demand allocation of voxel blocks, thus allowing the mapping of large scale environments.

3.3.1.4 *Surface extraction*

The availability of a dense TSDF grid allows the generation of accurate surface predictions from arbitrary viewpoints by finding the zero-level isosurface stored within the voxel grid. To this purpose, a virtual camera is placed in the environment, with position and orientation of interest, and a per-pixel raycast operation is performed, as described by Parker et al. [95]. By marching a ray for each pixel in the virtual image plane (in parallel, on the GPU), from the camera centre, and sampling the TSDF values of the voxels being traversed it is possible to identify *zero-crossings* of the function. A surface being observed from the front is detected when, whilst marching a ray, a positive-to-negative transition in the SDF values is identified. By interpolating the position of the positive and negative-valued voxels it is possible to estimate the position of the 3D point representing the surface. On the converse, if a negative-to-positive zero-crossing is encountered (signifying a surface observed from the inside of an object) or if the ray being marched exits the reconstruction volume, the corresponding pixel is set to not visible.

A normal for each raycasted point can be computed by numerically approximating the derivatives of the TSDF values in the neighbourhood of the point itself. Colour for the rendered surface can also be extracted from the voxel grid, by trilinearly interpolating the colours associated to the 8 voxels located around raycasted points. More details on the surface raycasting process and techniques to speed up the ray marching phase can be found in the article by Newcombe et al. [91].

3.3.1.5 *Mesh extraction*

An unstructured mesh can also be generated from the TSDF volume: by deploying the marching-cubes algorithm [78] it is possible to extract a set of triangles representing the zero-level isosurface encoded by the SDF values. The mesh extraction algorithm can be trivially parallelised on the GPU, by processing each voxel with an ad-hoc thread. Surface normals for each triangle are computed by deploying the cross product between two of its edges, whilst vertex colours can be sampled from the grid by trilinearly interpolating the nearby voxel colours, as described in the previous subsection.

For further details on the marching-cubes algorithm we refer the interested reader to the original paper by Lorensen and Cline [78]. In the following subsections we will detail the modifications applied to the TSDF data structure and the algorithms used to integrate semantic informations in the reconstruction.

3.3.2 *Data structure for labelled reconstructions*

To obtain a densely labelled representation of the environment, we assume the RGB-D sensor output to be fed to a semantic segmentation algorithm. Without lack of generality, the output of such an algorithm can be represented as a "category" map, *i.e.* a bitmap having the same resolution as the input image, wherein each pixel is assigned a discrete label identifying its category (or the lack thereof).

Moreover, we assume to be provided with a "score" map, where each value represents the confidence of the labelling algorithm in assigning a category to the corresponding pixel of the input image. Different semantic segmentation algorithms may indeed produce their output in heterogeneous formats (*e.g.* per-pixel categories, labelled superpixels, 2D or 3D bounding boxes, 3D cluster of points, polygons...) but it is typically possible to reconcile those into the aforementioned intermediate representation. The reliance on such an algorithm-agnostic format may also allow us to exploit, simultaneously, the output from diverse labelling techniques, either aimed at detection of different categories or in order to combine their predictions by fusing the score maps. As not every semantic segmentation technique can be run in real time for every frame captured by the sensor, the proposed label storage and propagation technique also allows for the robust integration of unlabelled frames.

As described in Subsection 3.3.1, a typical TSDF volume holds for each voxel the (truncated) distance of its centre from the closest surface in the environment together with a weight (also truncated to a maximum value) and the colour of the observed surface.

To augment the voxel data structures and label elements of the volume, several approaches may be envisioned. The most informative is to store, as an histogram, a probability mass function representing the probability for the voxel to represent an object of a certain class. Advantage of this approach is the possibility to properly label a multi-category voxel (such as one spatially located between two or more objects), also, analogously to the trilinear interpolation of SDF values and colours, one may interpolate between neighbouring voxels to obtain a spatially continuous *p.m.f.* Unfortunately, practical memory occupancy issues forbid us to rely on such an approach: each voxel already holds a TSDF value, a weight and associated colour (together with its weight). Subsection 3.3.1.3 describes in detail the memory requirements of the *vanilla* version of the reconstruction pipeline we are describing. Typical consumer GPU cards rarely provide more than $2-3$GB of total usable memory. Hence, by encoding the probability of each class using a single byte (as floating point numbers in the interval $[0, 1]$ can be trivially mapped to integers in the range $[0, 255]$ with an acceptable loss of precision), we can not store probabilities for more than $8-12$ categories without filling most of the available GPU memory. Also, since the integration of a new frame into the volume during the "mapping" phase of the algorithm requires a visit to each voxel in the grid, the more categories one wishes to handle, the slower turns out the entire tracking pipeline, practically limiting the maximum number of probabilities that can be stored in each voxel.

The above considerations lead us to store, instead, a single category per voxel, together with a "score" expressing the confidence on the accuracy of the assigned label. In the chosen representation discrete labels are stored as unsigned short numbers while the score is represented once again as an half precision floating point number, bringing the total memory occupation for a single voxel to 12 B (including the pre-existing data). Clearly we lose information using such label encoding, as we can no longer represent properly those voxels featuring more than one likely label but, accordingly, the memory requirements for a 512^3 voxel grid is fixed to 1.5 GB of GPU RAM, regardless of the total number of handled categories. Moreover, such a minimal representation mandates special care in implementing the volume update operation to insert new labelled data into the grid, in order to avoid situations where a voxel gets continuously switched between different categories. The process undertaken to integrate new labelled frames in the TSDF volume is described in the next subsections, with the assumption that the fusion step described earlier is performed at the same time as it.

3.3.2.1 *Label Fusion Process*

As mentioned, to store semantic information into each voxel, we augment the data structure by adding a discrete category label together with a floating point score, expressing the confidence in the stored label. Hence, a running average approach such as that used during the depth fusion step cannot be used for the semantic labelling information, as different categories cannot be directly confronted. We could, hypothetically, store in each voxel the labels as we receive them (by projecting each 3D cell's coordinates onto the label bitmap and sampling the corresponding pixel), but this would be prone to errors, as a single mislabelled region would possibly overwrite several correct voxel labels acquired in the past. Additionally, not every pixel may be labelled; possibly entire frames, when using a slow semantic segmentation technique which cannot be run on every input image.

We therefore propose an evidence weighting approach: similarly to the depth fusion process described previously, the coordinates of each voxel in the volume are projected (in parallel, on the GPU) onto the depth image and label/confidence maps and, *iff* they result within the truncation distance δ from the observed surface, we update the informations stored in the voxel depending on the sampled category and score pair. Specifically, each time a voxel is projected onto a pixel having the same label, we increment its score. On the converse, if the category stored in the voxel differs from the one in the corresponding pixel (*e.g.* due to a labelling error or to being on the seam between two differently labelled regions), we decrement the associated score. Only when the score reaches a negative value we replace the stored category with the new one.

As for the evidence increment/decrement weight applied to the score, we deploy the confidence of the semantic segmentation algorithm, as sampled from the input score map that we assume to be provided together with the labelling output itself. This choice naturally induces an hysteresis-like effect, protecting the consistency of the labels stored into the volume when areas of the input image are assigned to different categories in subsequent frames. Typically, a mislabelled region has associated a low score, such value will then not be able to bring in enough evidence to change the category associated with a correctly labelled area of the reconstruction. Conversely, assuming the initial labelling of a region to be wrong (*i.e.* with a low confidence), a correct labelling from subsequent frames will easily be able to replace the initial, erroneous, tag. A possible pitfall becomes evident if, for any reason, the confidence associated to a wrong labelling result by the semantic segmentation algorithm is very high but, as more frames are integrated into the TSDF volume, stored scores will increase above the maximum one the labelling algorithm is able to provide; when such a situation is reached, a single incorrect segmentation will not have the chance to adversely affect the volume contents. An unlabelled area (or entire frame, without lack of generality) has no effect on the volume labelling process: each corresponding voxel will be left unchanged.

Similarly to the geometric integration approach, we clamp the maximum label score for a voxel to allow for an easier change of category if, suddenly, a region of space is consistently tagged as a different object for several frames (*e.g.* in non-static situations, when an object is removed from the scene). Algorithm 1 shows the pseudocode for the proposed volume-based label updating process.

3.4 EXPERIMENTAL EVALUATION

To evaluate the proposed volume labelling approach we perform tests using different types of semantically segmented data. Our tests deploy the video sequences included in the Sun3D dataset [123]. On their website, Xiao, Owens and Torralba provide multiple RGB-D video sequences captured using a Kinect sensor, depicting typical indoor environments such as hotel, conference rooms or lounge areas. Unique to this dataset, is the presence of manually acquired object annotations, in the form of per-object polygons, for multiple sequences. Each object is also given a unique name, which allows

Algorithm 1 Pseudo-code of the label updating process

```
for all voxels in the volume do
    $\mathbf{p_v} \leftarrow$ 3D world coordinates of the voxel centre
    $\mathbf{u_v} \leftarrow$ projection of the voxel onto the image plane (see Equation 3.2)
    $s_v \leftarrow$ SDF associated to the voxel (see Equation 3.3)
    $L_{in} \leftarrow$ category associated to the pixel $\mathbf{u_v}$
    $W_{in} \leftarrow$ labelling score associated to the pixel $\mathbf{u_v}$)
    $L_{tsdf} \leftarrow$ category associated to the current voxel
    $W_{tsdf} \leftarrow$ labelling score associated to the current voxel
    if $|s_v| \leqslant \delta \wedge L_{in} \notin (\mathrm{unlabeled}, \mathrm{background})$ then
        if $L_{tsdf} = \mathrm{unlabeled} \vee W_{tsdf} < 0$ then
            $L_{tsdf} \leftarrow L_{in}$
            $W_{tsdf} \leftarrow W_{in}$
        else if $L_{in} = L_{tsdf}$ then
            $W_{tsdf} \leftarrow \min(W_{tsdf} + W_{in}, W_{clamp})$
        else
            $W_{tsdf} \leftarrow W_{tsdf} - W_{in}$
        end if
    end if
end for
```

us to tell apart several instances of a same category (*e.g.* in a hotel room sequence we may have “pillow 1” and “pillow 2”).

To parse the dataset’s own object representation into our intermediate labelling format, described in Subsection 3.3.2, we adopt the following approach:

CATEGORY MAP Each named object is given an increasing (and unique) integer identifier, afterwards, its bounding polygon is painted as a filled shape into our category bitmap. Being the source data result of a manual annotation process, partial overlap of the object polygons is not considered a concern.

SCORE MAP Annotated shapes are the result of a manual annotation process, we therefore consider the *labelling algorithm*’s confidence maximal. Similarly to the category map, we draw each object’s bounding polygon onto the score map and fill it with the floating point value 1.0.

Figure 3.3 shows a frame from the hotel room sequence contained in the aforementioned dataset. We see that each object is correctly labelled and their confidences are maximal due to the manual labelling process.

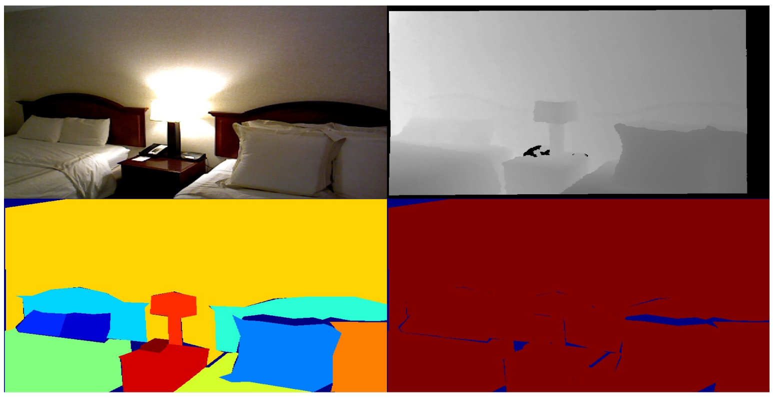


Figure 3.3: Labelled frame from the hotel room sequence of the Sun3D dataset [123]. From top left in clockwise order: RGB frame, depth frame, score map and category map. The score map has been drawn in false colours to increase visibility (blue is the minimum value while red is the maximum). In the category map each colour represents a different object instance.

We provide two kind of results, first by proving the robustness of the method under presence of synthetic noise in the labeller's output (Subsection 3.4.1). Subsequently, we show densely labelled volumes for several sequences, obtained using either ground truth labelling data or the semantic segmentation produced by a state of the art algorithm. For the latter, we evaluate the capability of the proposed fusion technique to reduce the number of erroneously labelled areas in the reconstructed volumes (Subsection 3.4.2).

3.4.1 *Robustness to synthetic label noise*

To investigate on the robustness of the proposed volumetric label integration process with respect to per-pixel semantic segmentation errors, for all the considered sequences, we corrupt the ground-truth category map associated with each frame with synthetically generated white noise. We then compare the resulting labelled volume to a reference volume obtained by executing the label fusion process on the noiseless, manually annotated, category maps. In particular, considering only those labels assigned to voxels representing a surface element (i.e. the zero-level isosurface of the TSDF), we compute the volumetric labelling error rate, *i.e.* the fraction of misclassified surface voxels. Our synthetic noise model is as follows: we sample pixels from the category map with a certain probability so to switch

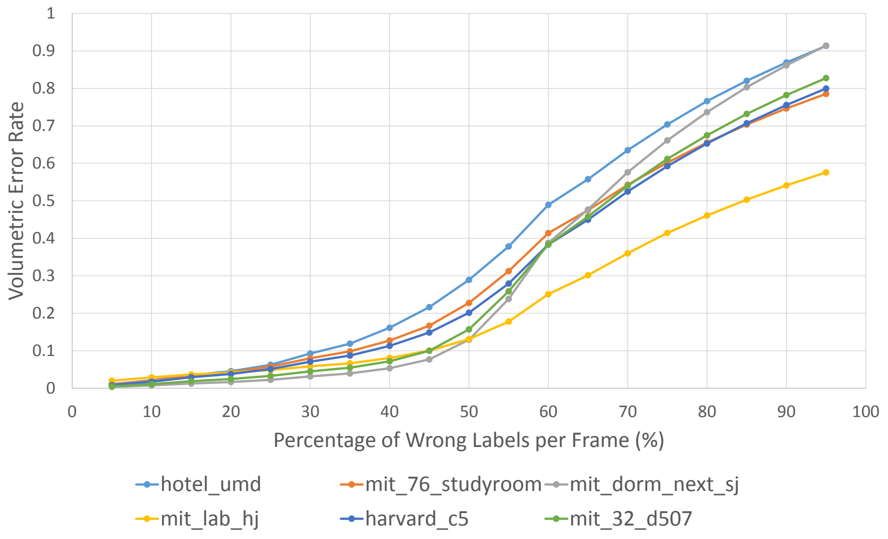
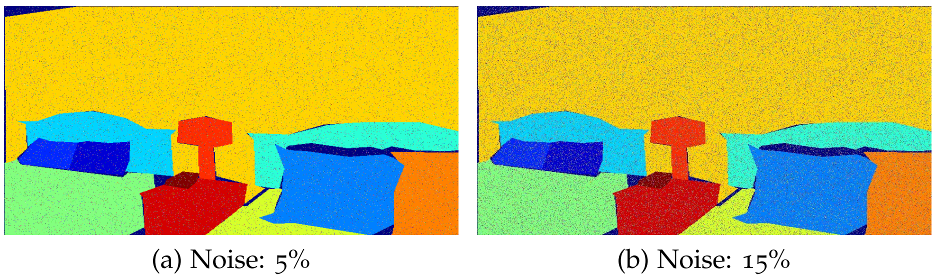


Figure 3.4: Volumetric labelling error rate for several sequences of the Sun3D dataset [123] when synthetic noise is added to the ground-truth category labels.

their correct label to wrong ones uniformly selected from the total pool of labels present in the sequence being examined. We also assign maximum confidence to such switched labels.

Figure 3.4 shows how, though the image labeller output is corrupted (as illustrated in Figure 3.5), thanks to the temporal label integration process, the final volume features a consistent labelling wherein each voxel is likely to have been correctly classified. Even when the probability to corrupt a label is as high as 50%, the proposed label integration can reduce the final volumetric error rate significantly, *i.e.* squeezing it down to less than 25% typically, to much less than 20% quite often. For more than 50% of wrong labels per input image, the error grows almost linearly with the noise level. The label fusion process still turning out beneficial in terms of noise attenuation: *e.g*, with as much as 70% wrong labels per image, the amount of misclassified surface voxels is typically less than 60%.

Figure 3.6 depicts the semantic reconstruction of a portion of the environment explored through the mit_dorm_next_sj sequence. It can be observed that the labelled surface represents accurately both the shape as well as the semantic of the objects present in the environment. The comparison between Figure 3.6a and Figure 3.6b allows for assessing the effectiveness of the temporal label integration process: though as many as 30% of the per-pixel labels in each


(a) Noise: 5%　(b) Noise: 15%

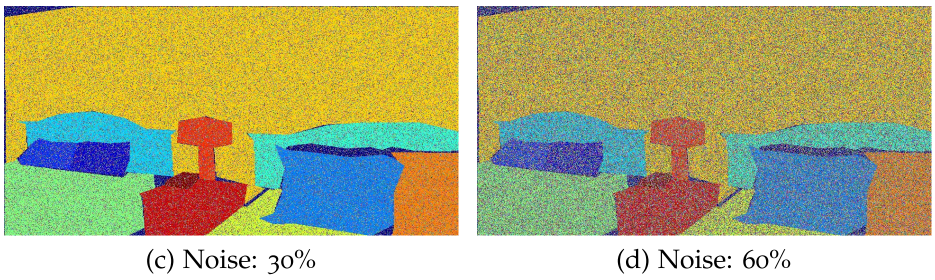

(c) Noise: 30%　(d) Noise: 60%

Figure 3.5: Examples of category bitmaps fed to the label fusion algorithm when correct labels are corrupted by increasing amounts of noise.

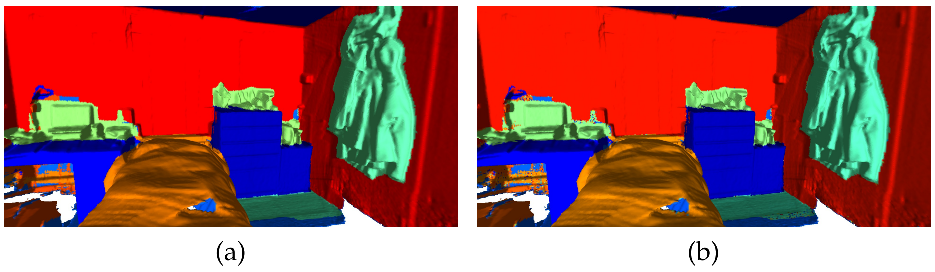

(a)　(b)

Figure 3.6: Semantically labelled reconstructions of the dorm sequence in the Sun3D dataset [123]: each surface element is coloured according to its category label. Left: reconstruction from noiseless per-pixel category maps. Right: reconstruction when 30% of the input labels in each map are switched to wrong. Labelling errors are visible by zooming onto the desk area only.

frame are wrong, just a few errors are noticeable with respect to the semantic reconstruction based on perfect noiseless input data. Indeed, such errors are mostly concentrated in the desk area, where the sensor did not linger for multiple frames and thus the evidence weighting process turned out less effective.

3.4.2 *Results in real settings*

To evaluate the effectiveness of our technique when using a real semantic labelling algorithm, we deploy the recent Semantic Segmentation approach proposed by Long, Shelhamer and Darrell [77]. Such algorithm uses a Convolutional Neural Network to produce a per-pixel labelling of an input image. The authors made available several pre-trained networks[2] based on the open source Caffe deep learning framework [62]. We focused our evaluation on the "FCN-16s NYUDv2" architecture due to similarities of its output category set with the type of objects present in the Sun3D dataset. The chosen network processes RGB-D images and produces per-pixel scores for 40 categories defined by Gupta, Arbelaez and Malik [53]. Using the aforementioned algorithm to label each frame of the input sequences, we fed our volume labelling pipeline with category maps wherein each pixel is assigned to the object class having the highest probability, storing then such values into the respective score maps.

Figures 3.7, 3.8, 3.9, 3.10, 3.11, 3.12 and 3.13 show views from the semantically labelled surfaces obtained by processing some of the sequences of the Sun3D dataset [123]. To allow for better comparative assessment of the performance achievable in real settings, in each Figure we report both the reconstruction obtained by feeding our algorithm with ground truth labels and with the output from the CNN mentioned above. Per-object identifiers from the Sun3D dataset have been manually mapped onto the corresponding categories defined in [53] to facilitate the comparison of results (in the hotel sequence of the Sun3D dataset for example, four different pillow objects are defined; we mapped all such identifiers to the single "pillow" category). Based on the comparison to the ground-truth reconstructions, it can be observed that the majority

2 https://github.com/BVLC/caffe/wiki/Model-Zoo#fcn

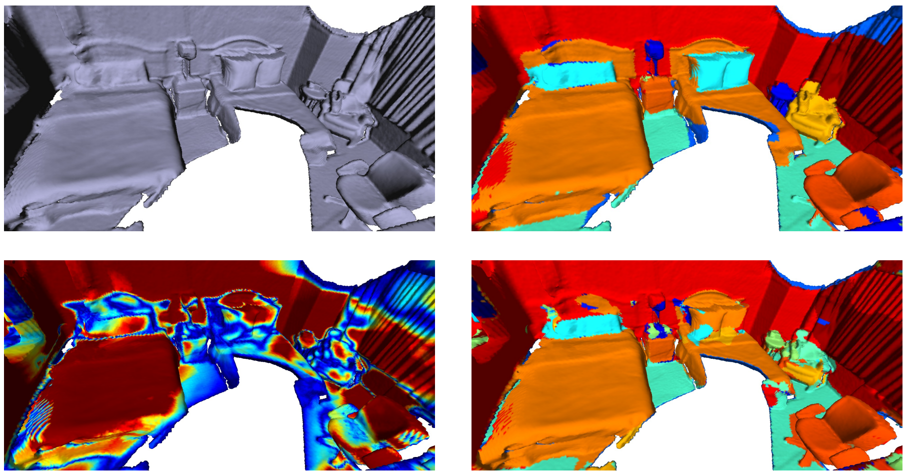


Figure 3.7: View from hotel_umd sequence of the Sun3D dataset [123]. From the top left, in clockwise order: standard KinectFusion output, semantically labelled view obtained by fusing manually annotated categories, semantically labelled view and associated confidence map obtained by fusing the labels computed by the CNN [77].

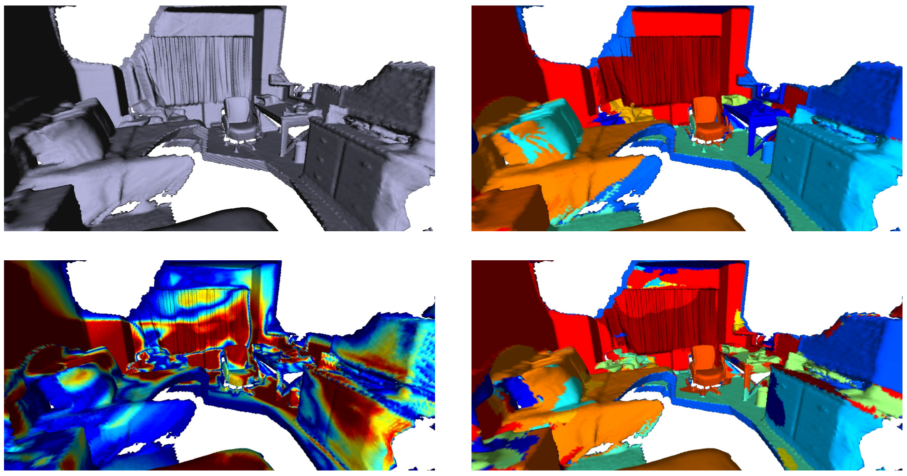


Figure 3.8: A second view from the hotel_umd sequence of the Sun3D dataset [123]. Images are ordered as in Figure 3.7.

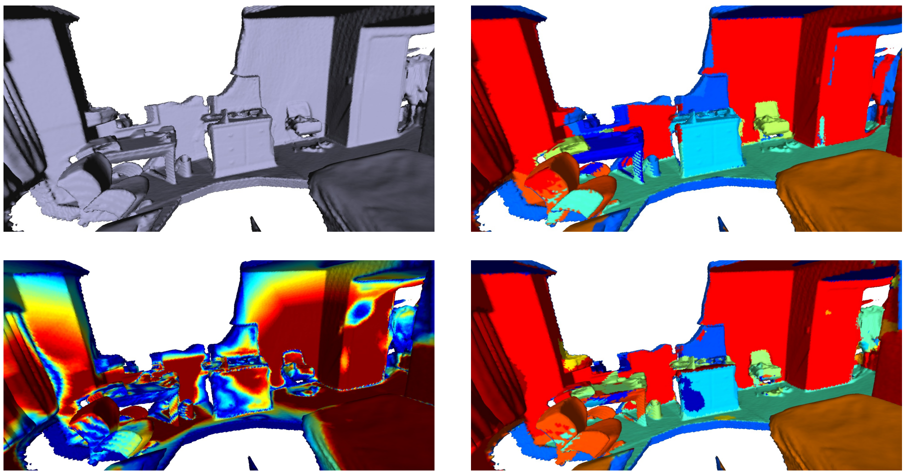


Figure 3.9: A third view from the hotel_umd sequence of the Sun3D dataset [123]. Images are ordered as in Figure 3.7.

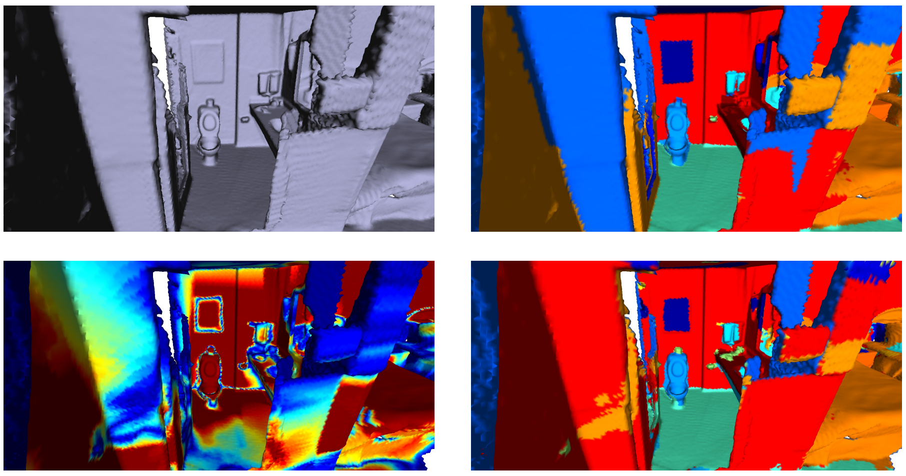


Figure 3.10: A fourth view from the hotel_umd sequence of the Sun3D dataset [123]. Again, images are ordered as in Figure 3.7.

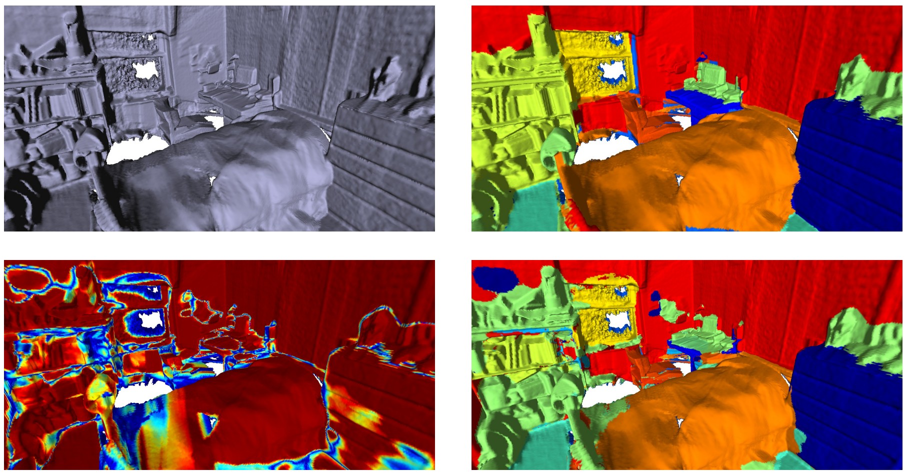


Figure 3.11: A view from the mit_dorm_next_sj sequence of the Sun3D dataset [123]. Once again, images are ordered as in Figure 3.7.

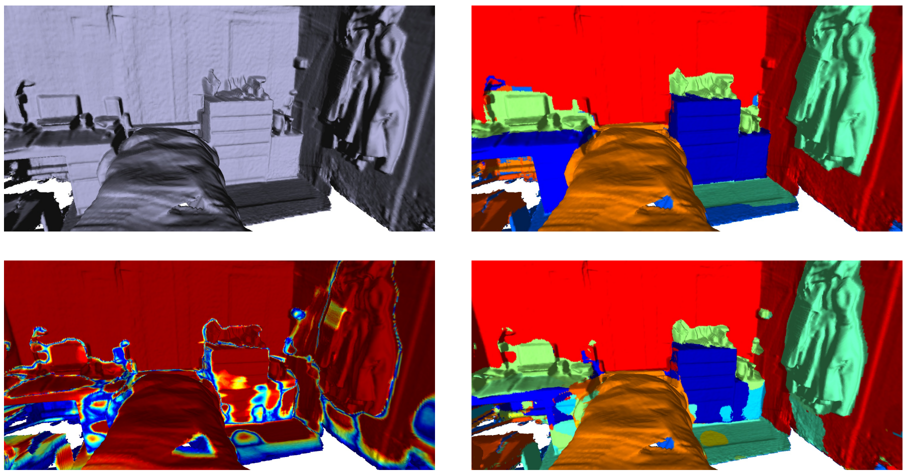


Figure 3.12: A second view from the mit_dorm_next_sj sequence of the Sun3D dataset [123]. Images are ordered as in Figure 3.7.

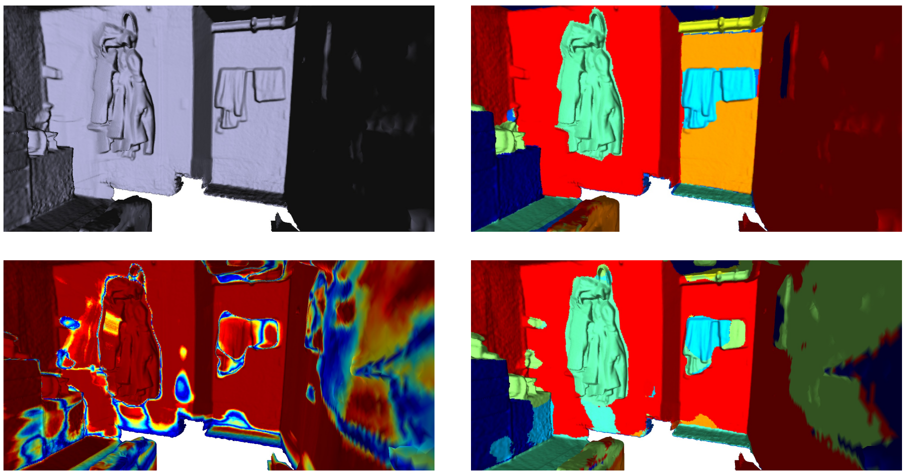


Figure 3.13: A third view from the mit_dorm_next_sj sequence of the Sun3D dataset [123]. Once again, images are ordered as in Figure 3.7.

of the labelled regions are consistently and correctly identified by the real algorithm and that, where labelling errors have been made, the associated confidence provided by the proposed label integration technique is likely low (such as in the TV stand in Figure 3.9 or on the bed corner, in Figure 3.11). We also provide a supplementary video[3] depicting fully labelled volumes for the two Sun3D sequences "hotel_umd" and "mit_dorm_next_sj". In the video we show the output of our algorithm when feeding it with manually annotated images and per-pixel categories provided by the CNN.

Finally, in Table 3.1 we assess the benefits brought in by our volumetric label integration technique with respect to per-frame labelling in real settings, *i.e.* when deploying a real semantic labelling algorithm such as the CNN proposed in [77]. The first part of the table reports per-frame semantic labelling error rates: this metric is computed for each frame using the ground-truth labels provided with the Sun3D dataset, dividing the number of incorrectly labelled pixels by the total number of labelled pixels. We show average per-frame error rate and associated standard deviation for the two sequences considered during the evaluation. The rightmost column, then, displays the volumetric error rate, *i.e.* the percentage of erroneously labelled surface voxels in the final reconstruction of the 3D volume (the same metric as Figure 3.4). The results in Table 3.1

3 `https://vision.disi.unibo.it/~tcavallari/phd_thesis/psivt2015.mp4`

	Per frame error rate (%)		Volumetric error rate (%)
Sequence	Average	Std. Dev.	
hotel_umd	34.9	22.6	24.1
mit_dorm_next_sj	26.1	15.7	19.3

Table 3.1: Volumetric vs. per-frame labelling The left side of the table reports the error rates yielded by CNN proposed in [77] on the individual frames of the two Sun3D sequences considered throughout this chapter. The rightmost column shows the percentage of voxels incorrectly labelled by our volumetric label integration method.

vouch how the proposed label integration technique can handle effectively varying and large per-frame labelling errors so to provide a significantly more accurate semantic segmentation of the reconstructed environment. It is also worth pointing out that the volumetric error rates reported in Table 3.1 turn out higher than those yielded by synthetic label noise (Figure 3.4) due to the diverse nature of labelling errors. Indeed, while in the experiment dealing with synthetic noise each pixel has a uniform and independent probability to be assigned to a wrong category, in real settings it is more likely that large connected image regions get labelled wrongly due to the spatial smoothness constraints enforced by real semantic labelling algorithms, such as *e.g.* the CNN deployed in our experiments.

3.5 FINAL REMARKS

In this chapter we described a first approach to bridge the gap between semantic segmentation and dense surface mapping and tracking, so as to attain a semantically labelled, dense reconstruction of the environment explored by a moving RGB-D sensor. We demonstrated its robustness by introducing significant noise in the labelled data fed as input, as well as its effectiveness by comparing the labelled surfaces achievable by fusing ground truth semantic informations to those obtained by deploying a state of the art semantic segmentation algorithm.

Our goal is to provide a tool usable alongside any kind of semantic perception algorithm in order to incrementally gather high-level knowledge on the environment and store it within the map itself. By exploiting the availability of semantic informations in each voxel, it becomes feasible to raycast in real time a category and confidence bitmap pair. This will allow the user to obtain a continuous stream of semantically labelled frames, possibly interacting with the system while mapping the space so as to either linger on low-confidence regions or even correct or improve the acquired semantic information. Moreover, the raycasting of such segmented frames remains possible even while employing perception algorithms that cannot be run in real time: while the segmentation algorithm cannot produce any new output, the system remains able to display the labels stored into the map.

In the next chapter we will show how the deployment of semantic data stored in the reconstructed volume can be used to improve the camera tracking algorithm by exploiting semantic cues together with geometric information to better align the current camera view to the surface embedded into the TSDF volume and thus lead to the generation of more accurate reconstructions.

4 VOLUME-BASED SEMANTIC TRACKING

4.1 SUMMARY OF CONTRIBUTIONS

In the previous chapter we showed how the integration of semantic information such as that provided by an image labelling algorithm can benefit a dense RGB-D SLAM pipeline, by allowing the generation of fully labelled reconstructions of the environment being explored.

As detailed in Chapter 2, recent research on the task of dense volume reconstruction has focused on improving different shortcomings of the original KinectFusion algorithm [91], *i.e.*:

(a) impossibility to map large scale environments;

(b) drift in the camera trajectory caused by the accumulation of small per-frame errors during the tracking process;

(c) low reliability in case of movement in the observed scene;

(d) lack of high level information in the generated maps.

Having provided a technique amenable to storing high level informations in the reconstruction (see Chapter 3), in this chapter we extend it to tackle a second shortcoming of volume-based reconstruction systems: the accumulation of camera tracking drift. Accordingly, we present an extended KinectFusion pipeline taking into account per-pixel semantic labels gathered from the input frames during the camera localisation step so as to increase the accuracy in the estimated camera trajectory. Thus, we realize a SemanticFusion loop whereby per-frame labels help better track the camera, and successful tracking enables to consolidate instantaneous semantic observations into a coherent volumetric map.

4.2 INTRODUCTION

Since the publication of the KinectFusion paper by Newcombe et al. [91], significant interest has spurred on the topic of dense surface mapping and tracking by means of an handheld RGB-D sensor, this resulting in a multitude of proposals focused on extending and/or improving the original algorithm from a variety of diverse perspectives. Thus, for example, attention has been devoted to extend KinectFusion to reconstruct either large scale [66, 94] or non-rigid [90] scenes, to improve the tracking module [127] or also to achieve detection of known object instances [41].

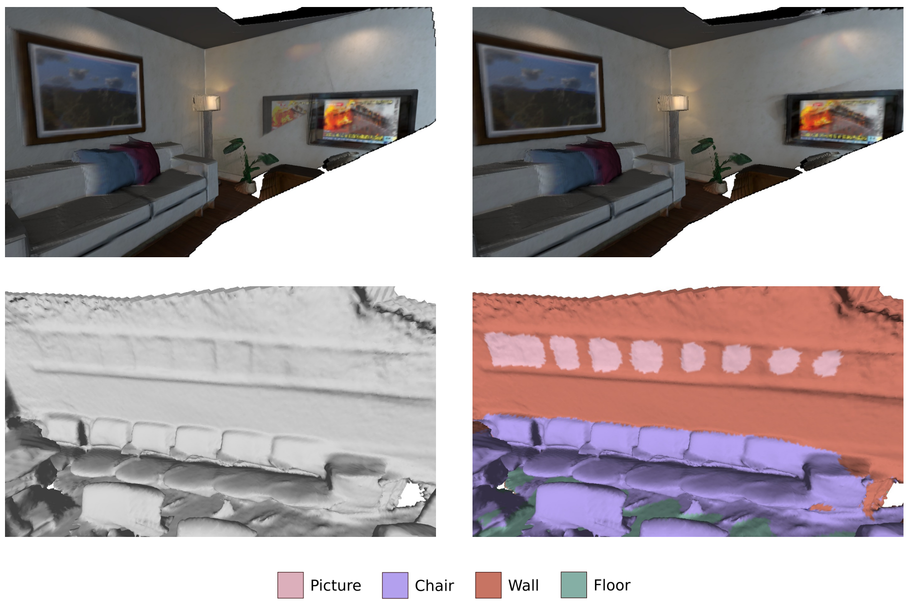


Figure 4.1: Advantages of the proposed technique. Top left: the scene reconstruction provided by KinectFusion shows gross errors due to the flat wall causing the geometry-based tracker to drift significantly. Top right: deployment of semantic cues into the tracking process may enable correct handling of planar surfaces. While KinectFusion is concerned with surface information only (bottom left), our method can provide a semantically labelled volumetric reconstruction of the workspace (bottom right).

Differently, the work described in this chapter is concerned with endowing KinectFusion with the ability to create a semantically labelled dense reconstruction of the environment as well as with deploying per-frame semantic information to improve camera tracking, the two processes carried out jointly and synergistically. Indeed, on one hand we track the camera by relying also on per-pixel semantic labelling of the current RGB-D frame, so that a likely camera pose should explain not only the geometry of the scene but also its semantics: *e.g.*, a pixel labelled as *picture* should preferably project onto a voxel tagged as *picture* than as *wall*. On the other hand, upon successful camera tracking we fuse the category map associated with the current frame into a volumetric representation, so that the final labelling results from suitable integration over time of the many fragile instantaneous observations delivered by the per-frame labeller. Thereby, semantic labelling ameliorates camera tracking, especially while acquiring scene elements – such as walls or windows – that may not provide distinctive geometric cues, and camera tracking robustifies semantic labelling by allowing for identification of observations dealing with the same surface patch at different times so to assess their likelihood and confidence. The two key advantages brought in by our proposal are exemplified in Figure 4.1.

Previous work has demonstrated how deployment of additional cues, such as colour [13] or contours [127], can significantly improve the accuracy of the original KinectFusion tracker, which relies on geometrical information only. Unlike previous work, here we advocate reliance on higher-level observations, such as per-pixel object category tags, to improve the tracking module and present a method that can also output automatically a fully labelled dense reconstruction of the environment, with confidence scores for multiple object categories stored into each voxel of the mapped space. This kind of semantically rich output may vastly facilitate difficult tasks related to 3D indoor scene understanding, object discovery, object recognition and grasping, path planning and human/robot navigation.

In the following section we will describe the implementation of the semantic tracker that we deploy in the KinectFusion pipeline. First, in Section 4.3, we will detail once again the modifications applied to the original voxel data structure: whilst in Chapter 3 we advocated the inclusion of a single category label and associated score to each voxel of the volume, for the purpose of semantic camera tracking we need to have available confidence values for multiple categories. Next, in Section 4.4, we will describe the label-based tracking approach, core of this chapter. Finally, in Section 4.5, we will present quantitative and qualitative results proving that the semantic tracking approach effectively reduces the error in estimating the real camera trajectory in presence of geometrically ambiguous surfaces and showing the fully labelled reconstructed volumes that allow us to perform high level reasoning on the observed environments.

4.3 MULTI-LABEL RECONSTRUCTION VOLUME

Cornerstone of the semantic tracking approach that we propose is the availability, during the reconstruction, of scores for multiple categories in each voxel of the reconstruction volume. Reasons for this claim will be given in the section describing the camera pose estimation technique but, to summarize them, we will briefly anticipate the idea on which the tracker is based. As mentioned in the previous section, we propose to track the camera by relying, in addition to geometric properties of the scene being observed, also on per-pixel semantic labelling of the current RGB-D frame: a likely camera pose should explain not only the geometry of the scene but also its semantics. Camera trackers (geometry-based as well as colour-based or, as in this case, employing *semantic* clues) estimate the sensor pose in the world by minimising an error term depending on the alignment between input frames and the current reconstruction stored in the map. The minimisation process requires the computation of gradients for the error function, in the semantic case this requires, for each pixel, to compute a gradient over the voxel grid depending on the label associated to the pixel, this in turn preventing us from storing a single label in each voxel (as described in Subsection 3.3.2) since the gradient computation would then not be feasible.

In each voxel we store multiple category labels as an histogram, wherein the value stored in each bin is as higher as the corresponding label is observed more frequently and with higher confidence scores. As such, a bin value becomes correlated to the probability that an object of a certain category is located at the voxel's 3D position. This representation allows us to both compare the evidence about the diverse categories gathered within each voxel, as well as to rank the labelling confidence between different voxels.

4.3.1 *Fusion of semantic labels in a multi-label volume*

In principle, updating the bins of the label histogram associated with a voxel may be performed similarly to the TSDF and colour updating processes described in Subsection 3.3.1.1 and Subsection 3.3.1.2.

As already mentioned, we assume the semantic segmentation algorithm to provide a label map $l(\mathbf{u}) : [0, w) \times [0, h) \to \mathbb{N}$ (with w and h indicating, respectively, width and height of the input RGB-D image) and a corresponding confidence map $s(\mathbf{u}) : [0, w) \times [0, h) \to \mathbb{R}_0^+$. When integrating a new frame then, at each voxel we would be able to update only the bin associated with the label observed at the corresponding pixel and we are faced with the issue of whether and how to update the other bins. Updating only the observed bin would inevitably result in the *plateauing* of the histogram at relatively high values in case several highly confidently and incoherent label measurements get fused into the same voxel over multiple frames. Instead, we devised a suitable updating technique that tackles this issue by both increasing the value of the bin associated with the observed label as well as decreasing those of all other bins. In particular, the increment is proportional to the confidence score provided by the labeller while the decrement factor takes also into account the evidence hitherto gathered into the histogram, to penalize less higher bins.

Formally, we denote the category histogram stored in each voxel via a function $\Gamma(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}^N$ (analogously to the SDF function $\Phi(\mathbf{x})$ described in Subsection 3.3.1), with N the number of categories of interest. We update each voxel $\mathbf{p_v}$ lying within the truncation region δ of the observed surface (as in the colour update process described in Subsection 3.3.1.2) in parallel, on the GPU, by defining the weight factor w_{l_i} applied to each category $i \in [0, N)$ depending on the label assigned to the pixel, $l(\mathbf{u_v})$ and its associated score $s(\mathbf{u_v})$, both computed by the semantic segmentation algorithm:

$$w_{l_i} = \begin{cases} s(\mathbf{u_v}) & \text{for} \quad i = l(\mathbf{u_v}) \\ \Gamma_t^i(\mathbf{p_v})(1 - s(\mathbf{u_v})) & \text{for} \quad i \neq l(\mathbf{u_v}) \end{cases} \tag{4.1}$$

With $\mathbf{u_v}$ corresponding to the pixel in the input label and confidence maps that the voxel $\mathbf{p_v}$ is projected onto, after application of the camera pose transformation (as in Equation 3.2). Then, we update each bin of the category histogram as follows:

$$\Gamma^{i}_{t+1}(\mathbf{p_v}) = \frac{\Gamma^{i}_{t}(\mathbf{p_v})\Omega_t(\mathbf{p_v}) + w_{l_i}}{\Omega_t(\mathbf{p_v}) + 1} \tag{4.2}$$

It worth observing that, analogously to TSDF and colour, the label update step is tantamount to a running average over the number of observations for a voxel. For the bin corresponding to the pixel's label, we use the labeller's score as the update factor, while for all the others we reduce the running average value by an amount depending on both such score and that previously stored in the bin being updated.

4.4 TRACKING ALGORITHM

To update the information stored into the volume by integrating new measurements, we need to track the RGB-D sensor as it moves within the environment. In KinectFusion [91] camera tracking is performed by ICP-based alignment [9] between the surface associated with the current depth image and that extracted from the TSDF. Later, Bylow et al. [14] and Canelhas, Stoyanov and Lilienthal [17] proposed to track the camera by direct alignment of the current depth image to the mapped environment encoded into the TSDF as the zero-level isosurface. This approach has been proven to be faster and more accurate than the original KinectFusion tracker.

Camera tracking by direct alignment of the acquired depth image with respect to the TSDF volume relies on the consideration anticipated in Section 3.3: assuming that the estimated pose of the camera is correct and noise does not affect the current and previous depth measurements, then each depth pixel should correspond to the projection of a 3D world point that, given the content of the TSDF grid, features a null distance function (*i.e.* lies on the surface of the reconstruction). Accordingly, they define an energy function with an error term associated to each pixel of the depth image being tracked.

By minimising such function, the rigid body transformation determining the camera pose can be estimated.

In this chapter, we propose to consider also the category labels assigned to pixels in the input image within the objective function minimized by the tracker. Indeed, a shortcoming of both the standard KinectFusion ICP tracking algorithm and the tracker based on direct alignment of depth images consists in the difficulty to localise the camera when the acquired frames depict scenes poor in geometry such as smooth or flat surfaces. We argue that exploiting the inherent structure determined by the boundaries between labelled objects can improve tracking accuracy when the scene features insufficient geometric clues.

Camera pose estimation is thus performed by finding the rigid body transformation, $T_{w,c} = (R, \mathbf{t}) \in \mathbf{SE}(3)$, that minimizes a cost function consisting of two separate terms. One term measures the geometric alignment of the depth frame to the surface implicitly encoded into the TSDF (as in [14]), while the other term captures the coherence between per-pixel labelling proposed by the semantic segmentation algorithm with respect to the label histograms already stored into voxel grid. By defining $\mathbf{p_u}$ as the 3D location in camera coordinate space of the point determined by a pixel $\mathbf{u}$ having a valid value in the depth image:

$$\mathbf{p_u} = \pi^{-1}(\mathbf{u}, d(\mathbf{u})) \tag{4.3}$$

The geometric error term for that pixel is then given by the truncated signed distance associated to its corresponding voxel:

$$E_d(T_{w,c}, \mathbf{u}) = \phi(T_{w,c}\mathbf{p_u}) \tag{4.4}$$

We analogously define the semantic error term, based on both the confidence of the labeller in the predicted pixel label and the amount of evidence already gathered on that label within the corresponding voxel:

$$E_l(T_{w,c}, \mathbf{u}) = s(\mathbf{u}) \left(1 - \Gamma^{l(\mathbf{u})}(T_{w,c}\mathbf{p_u})\right) \tag{4.5}$$

Thereby, the error term turns out high when the per-pixel labeler confidently predicts a category for which the degree of previous evidence stored in the category histogram is small.

The final objective function to be minimized by the tracker takes the following form:

$$E(T_{w,c}) = \sum_{\mathbf{u} \in \{[0,w) \times [0,h) : d(\mathbf{u}) > 0\}} \left(E_d(T_{w,c}, \mathbf{u})^2 + \alpha E_l(T_{w,c}, \mathbf{u})^2 \right) \quad (4.6)$$

Given a candidate pose, the first term of the sum quantifies the geometric misalignment between the current frame and the surface embedded into the TSDF. The second, instead, penalizes those poses where the labels assigned to pixels turn out incoherent with respect to the category histograms stored into the corresponding voxels. As such, the second term of the sum may be thought of as quantifying the semantic misalignment between the current frame and the volumetric map. The blending parameter, α, enables to weigh properly the contribution of two error terms.

By employing $\xi = (\nu_x, \nu_y, \nu_z, \omega_x, \omega_y, \omega_z) \in \mathbb{R}^6$, a minimal parametrization of the $T_{w,c}$ transformation, and knowing that the Lie algebra allows expressing $T_{w,c} = e^{\xi}$ and $\xi = \ln(T_{w,c})$ via the exponential and logarithm map operations [11, 110], we can define

$$E_{d\mathbf{u}}(\xi) = \phi(e^{\xi}\mathbf{p}_{\mathbf{u}}) \quad (4.7)$$

and

$$E_{l\mathbf{u}}(\xi) = s_{\mathbf{u}} \left(1 - \Gamma^{l(\mathbf{u})}(e^{\xi}\mathbf{p}_{\mathbf{u}}) \right) \quad (4.8)$$

so to rewrite Equation 4.6 as:

$$E(\xi) = \sum_{\mathbf{u}} \left(E_{d\mathbf{u}}(\xi)^2 + \alpha E_{l\mathbf{u}}(\xi)^2 \right) \quad (4.9)$$

Using Equation 4.9 we can, finally, express the camera tracking cost function as:

$$\xi = \underset{\xi}{\operatorname{argmin}} \sum_{\mathbf{u}} \left(E_{d\mathbf{u}}(\xi)^2 + \alpha E_{l\mathbf{u}}(\xi)^2 \right) \quad (4.10)$$

By linearising the function around an initial pose $\hat{\xi}$ (such as the estimated camera pose for the previously tracked frame) and assuming a small movement between the two acquisitions (as mentioned before, a reasonable assumption in the hand-held tracking scenario), it is possible to perform an iterative nonlinear minimization using a method such as Levemberg-Marquardt [75, 81] in order to estimate the camera pose, ξ, that optimally aligns the current frame to the volumetric reconstruction both geometrically and semantically.

The gradient of the cost function required by the iterative minimization to compute the increment applied to ξ is attained independently for the two terms.

As for the gradient of the geometric term, in each pixel $\mathbf{u}$, we trilinearly interpolate the values of the TSDF in the 8 voxels closest to point $e^{\xi}\mathbf{p}_{\mathbf{u}}$.

The gradient for the semantic error term is also computed by trilinear interpolation, though, in this case, of the values $1 - \Gamma^{l(\mathbf{u})}(\mathbf{x})$, with $\mathbf{x}$ representing voxel coordinates around $e^{\xi}\mathbf{p}_{\mathbf{u}}$. Thereby, for each pixel we consider the category histogram bin associated to the label assigned to the pixel itself. It is worthwhile observing that optimization of the semantic error mandates maintaining in each voxel an histogram concerning the likeliness of all categories, as we propose in this chapter, whilst storing the evidence gathered for the most likely label only, as described in Chapter 3, would make it impossible to compute the gradient of the semantic error due to voxels lacking the information on whether moving along a certain direction would either increase of decrease the likeliness of the sensed pixel label.

4.5 EXPERIMENTAL EVALUATION

In this section we present quantitative (Subsection 4.5.1) and qualitative (Subsection 4.5.2) results showing how the proposed camera localisation algorithm, based on both geometry and semantic clues, can successfully reduce the trajectory error in several challenging sequences. Purposely, we employ the well-known RGB-D SLAM Dataset by Sturm et al. [111] and the ICL_NUIM Dataset by Handa et al. [56]. We also show reconstructed environments obtained from sequences part of the Sun3D Dataset [123] together with a sequence depicting a household environment.

As already mentioned, to successfully track the camera pose and update the volume, we assume to be provided in each frame with per-pixel labels and confidence scores by a semantic segmentation algorithm. Here, as in Subsection 3.4.2, we employed the Fully Convolutional Networks recently proposed by Long, Shelhamer and Darrell [77]. From the complete probability distribution over all the categories of interest output by the network we then select the label having the maximum probability and use such values to populate the $l(\mathbf{u})$ and $s(\mathbf{u})$ matrices given as input to our algorithm. The authors provide several pretrained networks based on different sets of object categories and input features. For this chapter we considered two such networks, one based on the 59 categories of the Pascal Context dataset [85] ("FCN-8s PASCAL-Context", as the authors name it in their paper) and another trained on 40 categories of the NYUDv2 dataset [105] ("FCN-16s NYUDv2"). While the FCN-8s network process only the input RGB frame to obtain the categories, the network trained on the NYUD dataset requires also computation of the HHA depth embedding [54]. The deployment of either network requires a vast amount of GPU computing power and memory but the latter, relying on processing both an RGB and HHA image, needs twice the GPU RAM as the former ($\approx$ 8.8GB), which reduces significantly the GPU memory available to store the proposed voxel grid containing category histograms. Eventually we decided to privilege the "FCN-8s PASCAL-Context" network in our experimental evaluation. We point out, though, that the semantic tracking technique we propose does not rely on a specific image labelling algorithm, but merely expects to be provided with per-pixel labels and scores.

As between the 59 categories detected by the aforementioned labelling algorithm we are interested only in a subset comprising indoor objects (e.g. "tables", "chairs", "walls", "tv monitors" and not categories such as "sheep" or "mountain"), we filter the network output at each pixel so to keep the most likely label within the subset of categories of interest. All the performed tests consider 12 categories.

A NOTE ON MEMORY OCCUPATION As in Subsection 3.3.1.3 we detail the memory occupation required by the proposed semantic voxel grid. In each voxel we store the following pieces of information:

TSDF VALUE Half precision float, 2 bytes

TSDF WEIGHT Half precision float, 2 bytes

COLOR R,G,B and weight channels stored each as unsigned char, 4 bytes

CATEGORY HISTOGRAM Each bin as a floating point value in the interval $[0, 1]$, mapped as $[0, 255]$ into an unsigned char, 12 bytes

The total occupancy for a 512^3 voxel grid storing a category histogram with 12 bins in each voxel is therefore $\approx 2.5\text{GB}$ of GPU memory, within the possibilities of modern graphic cards.

We would like to point out that, with a dense volume-based mapping approach such as the one described here, the memory footprint of the system exhibits a cubic dependence on both the voxel and workspace sizes. Comparatively, increasing the amount of bytes stored in each voxel increases memory occupancy only linearly. Thus, storing 12 more bytes in each voxel, as we do for labels, does not change the working constraints of the system dramatically. To clarify this claim, let us assume the extent of the environment to be 8^3m^3: by employing a quite standard 512^3 voxel grid we would be able to map the environment with a resolution of $\approx 1.6\text{cm}$, this requiring $1\,\text{GB}$ of GPU memory without storing per voxel-labels and, as mentioned before, $\approx 2.5\text{GB}$ when also storing an histogram with 12 label scores in each voxel. Conversely, should we wish to map the same environment with a voxel resolution of $0.5\,\text{cm}$, a grid of 1600^3 voxels would be necessary, this requiring $32\,\text{GB}$ of GPU memory without per voxel-labels and $\approx 80\text{GB}$ in case of semantic voxel grids. Both cases are intractable with the current graphics hardware thus, in practice, with the proposed system we are able to handle workspaces of similar size as KinectFusion.

4.5.1 *Quantitative results*

As mentioned, we evaluate the performances of the proposed label-aware tracking method on RGB-D sequences part of the TUM[111] and ICL_NUIM[56] datasets. As customary in the evaluation of SLAM algorithms, we deploy the RMS Absolute Trajectory Error performance metric described by Sturm et al. [111]. We compare our approach to a standard geometry-based tracker based on direct alignment of the depth image as well as to an improved tracker that employs also colour information by Bylow, Olsson and Kahl [13]. As the article describing the colour-based tracker did not consider the ICL_NUIM dataset in the evaluation, we implemented their tracking algorithm and show here the RMS ATE values also for our implementation.

As for the runtime configuration of our algorithm, we employ a voxel grid spanning $8^3 m^3$ of space. The truncation distance has been set to $0.3\,m$ and the α blending coefficient applied to our semantic error term to 0.085, after a grid search over the parameter space.

Figure 4.2 shows the accuracy of different tracking approaches in several sequences of the TUM RGBD-SLAM dataset[111]. Overall, we observe that most TUM sequences can be tracked quite successfully by a purely geometric approach, so that both semantics and colour have got no chances to bring in notable improvements in accuracy. Nor they cause any harm, though. Instead, in those sequences, like “floor” and “room”, turning more challenging for a purely geometric tracker, employment of additional clues, such as semantic labels – or color – does help reducing the tracking error quite significantly. Indeed, the improvement achievable by our semantic tracker versus a purely geometric approach is much higher in “room” than in “floor”, as the former sequence is characterized by a richer semantic content (*i.e.* presence of several object categories) while in the latter the sensor continuously observes the floor of a room, which renders color cues more distinctive than semantics.

Figure 4.3 shows the tracking error on the sequences of the ICL_NUIM dataset. Handa et al. [56] provide two sets of four RGB-D sequences obtained by rendering views from a synthetically generated model of two environments: a living room and an office. One set contains noiseless images while the depth frames in the

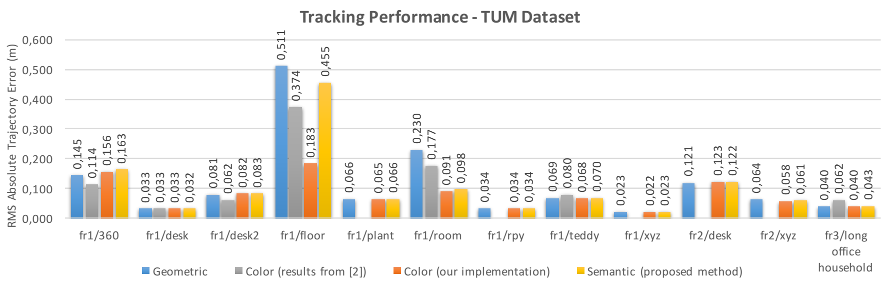


Figure 4.2: Tracking performances in sequences part of the TUM[111] dataset. *Geometric* represents the purely geometric KinectFusion tracker. For the *Color*-based approach we show both the results reported by Bylow, Olsson and Kahl [13] ($\alpha = 0.1$), as well as those achieved by our own implementation of Bylow's approach. *Semantic* denotes our proposal.

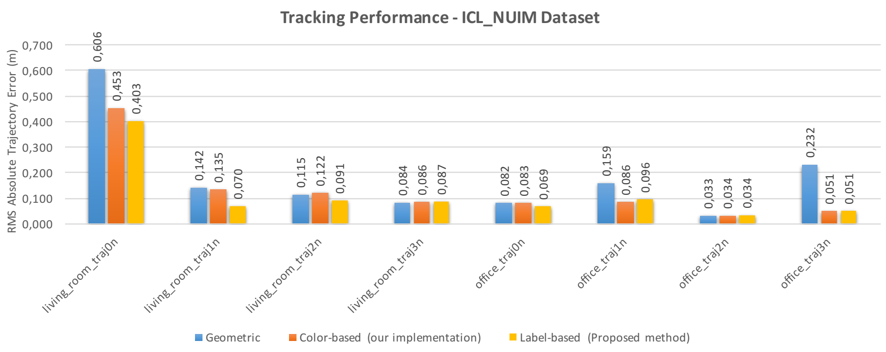


Figure 4.3: Tracking performances in sequences part of the ICL_NUIM[56] dataset. Series as in Figure 4.2.

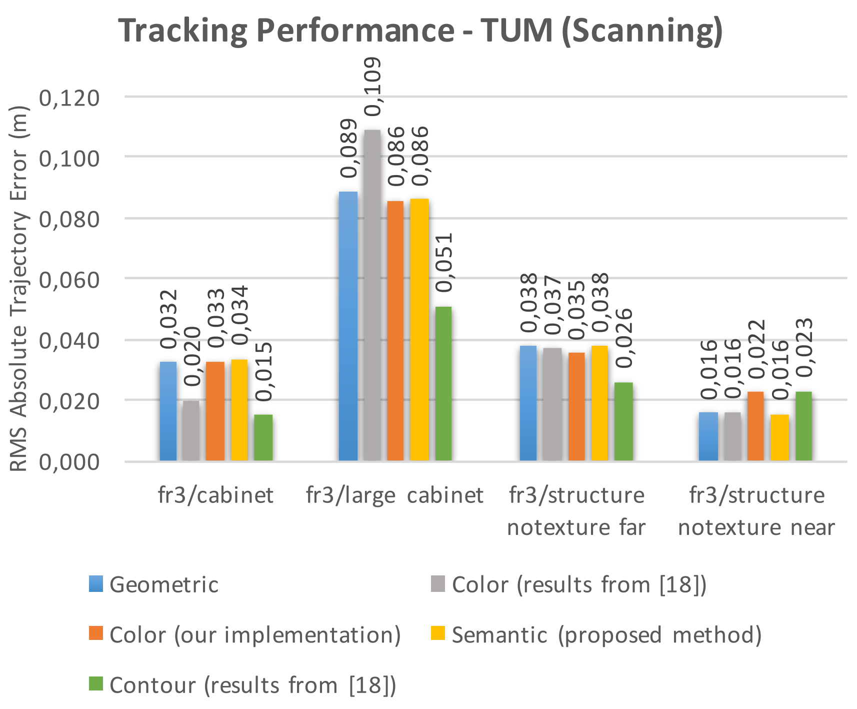


Figure 4.4: Tracking performances in the object scanning sequences considered in [127]. The method proposed in [127] is referred to as *Contour*. As for the original colour based tracker, we show here the results reported in [127], which were obtained by the authors running Bylow's code. As usual, we also provide the results attained by our own implementation of Bylow's tracker.

other have been corrupted by noise akin to that present in the images acquired by a real Kinect sensor. In our tests we consider the latter, noisy, set. Once again it can be observed how, in those sequences where the geometric tracker has more difficulties in estimating the correct camera trajectory, deployment of semantic labels can ameliorate tracking accuracy notably. Moreover, unlike the TUM dataset, it turns out here that, more often than not, semantic labels compare favourably with respect to colour cues, possibly due to the richer semantic content of the scenes.

Finally, Zhou and Koltun [127] evaluate their contour-based tracking method on a small subset of the TUM dataset, *i.e.* four sequences, each focused on scanning a single object featuring smooth and evenly coloured surfaces. In Figure 4.4 we report the trajectory errors for all the considered tracking methods, including Zhou's, on the four sequences used for the evaluation in [127]. Clearly, scanning a single object is neither the typical operating mode nor

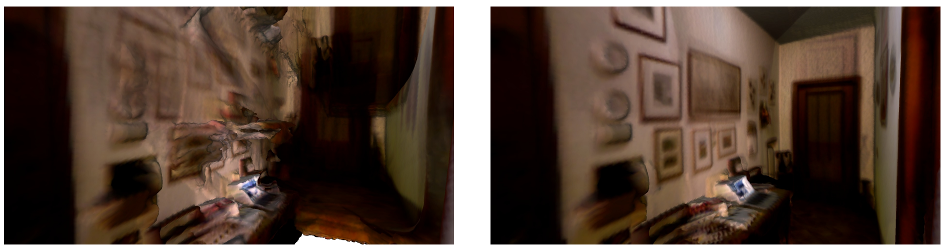


Figure 4.5: Household scene captured with a Kinect sensor. Left: a purely Geometric tracker cannot estimate camera poses accurately due to the flatness of the wall. Right: deployment of Semantic labels enable to map the environment correctly.

particularly suitable a scenario to our method, which, instead, is aimed at automatic reconstruction and volumetric labelling of relatively large workspaces featuring a number of diverse categories of interest (indeed, we set the volume size to $2.5^3 m^3$ in these four sequences). Yet, Figure 4.4 vouches how employing semantic labels in object scanning settings is not detrimental to the overall tracking accuracy. Contours seem the most effective cues in these settings, while the results yielded by semantic labels may be judged on par to those attainable by deploying colours.

4.5.2 *Qualitative results*

In this section we show, at first, qualitative results proving the advantages attainable by employing a label-aware camera tracker within a dense, volume-based reconstruction framework such as KinectFusion.

Figure 4.5 concerns an RGB-D sequence that we captured via a Kinect sensor in a corridor within a household environment. While the flatness of the wall prevents successful tracking based solely on the geometry of the surfaces, the low confidence of the "wall" label associated with the paintings provides enough distinctive cues to enable correct reconstruction of the environment.

Figure 4.6 deals with the hotel room sequence contained in the SUN3D dataset [123]. When mapping the hotel's bathroom, the camera briefly lingers on the mirror above the faucet. An RGBD sensor such as the Kinect is unable to detect the mirror as a flat surface, instead *mirroring* the shape of the reflected environment as a

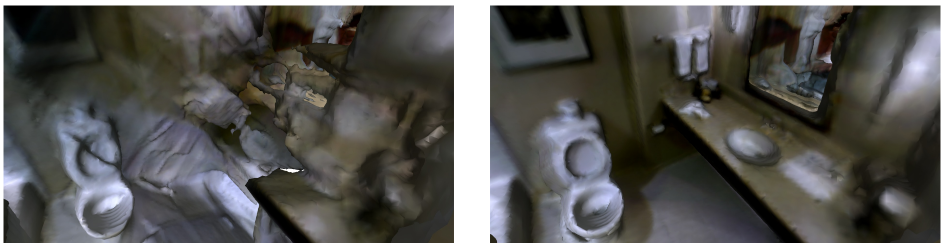


Figure 4.6: Fatal tracking failure prevented. Left: the mirror causes a purely geometric camera tracking approach to fail, the reconstructed volume turning out unrecognisable. Right: tracking succeeds and the bathroom is correctly mapped thanks to the employment of semantic clues during camera pose estimation.

window on a different room. A purely geometric tracking, thus, is unable to estimate the correct camera trajectory, resulting in a very bad reconstruction such as that shown by the left picture. However, the correct and coherent semantic labels associated to the surfaces surrounding the mirror deployed in our tracking cost function allow for better constraining camera poses and obtain an accurate reconstruction. The mirror still appears as a hole in the wall showing the rest of the hotel room because depth measurements in that spatial location are farther away than the mirror itself, but the rest of the bathroom is correctly reconstructed.

Finally, analogously to Subsection 3.4.2, we provide qualitative results to demonstrate the capability of our method to output semantically labelled volumes by relying on the category histogram stored in each voxel of the reconstruction. Figure 4.7 shows details of the "room" sequence (part of the TUM dataset [111], top row) and the aforementioned household sequence (also depicted in Figure 4.5, bottom row), with each voxel labelled according to the tag exhibiting highest score in the category histogram. The score is shown in an adjacent picture, as a heat map. It can be observed that most voxels are labelled correctly and heat maps are quite reliable, due to high confidence labels unlikely turning out wrong and mislabelled areas featuring low scores. Moreover, the maps provide evidence on the presence of both large scene structures as well as smaller objects. It is worth highlighting that with the "room" sequence the quality of the 3D reconstruction is equivalent to that achievable by deploying colour cues (see Figure 4.2) however, by our method, one can also gather high level information concerning which types of

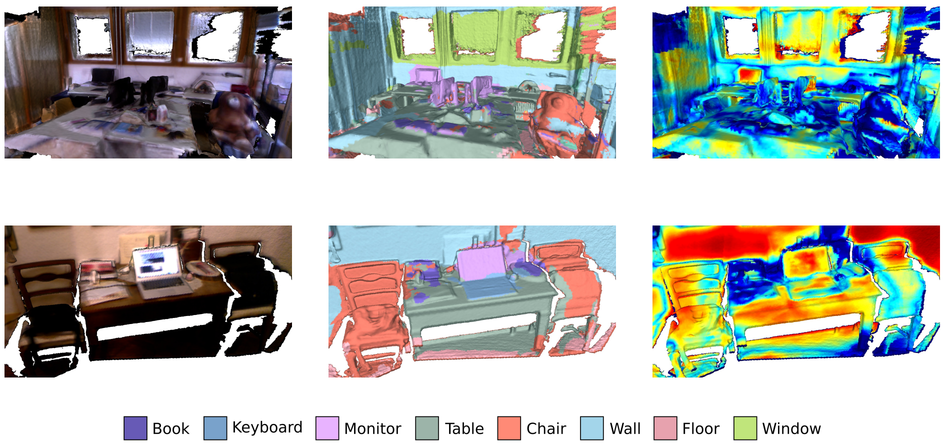


Figure 4.7: Details of the semantic reconstruction from the "room" sequence (from the TUM Dataset [111], Top) and household sequence (Figure 4.5, Bottom). Left: RGB mesh. Centre: Labels associated to the reconstructed volume. Right: confidence scores assigned to the labels (blue: low – red: high).

objects are present in the environment and where they are located in space. Additional qualitative results are available in the accompanying video[1].

4.6 CONCLUDING REMARKS

In this chapter, building on the ideas brought forward by the system described in Chapter 3, and peculiarly with respect to previous work aimed at robustifying the KinectFusion tracker, we show a dense, volume-based, SLAM pipeline that relies on a fully labelled volume wherein each voxel is endowed with the likeliness of the whole set of categories of interest, to generate a semantically labelled reconstruction of the scene being observed.

We have shown how high-level observations, such as per-pixel object category tags, can improve the dense mapping and tracking process popularized by KinectFusion, especially when dealing with surfaces featuring scarce geometric cues. Quantitative experiments on standard benchmark datasets suggest that, overall, reliance on semantic labels for tracking yields results comparable to deployment of colour. However, we believe that semantic information holds the potential to enable more reliable tracking as category tags are less

1 `https://vision.disi.unibo.it/~tcavallari/phd_thesis/r6d2016.mp4`

affected by nuisances such as light changes, shadows, reflections, blur. For example, a labeller working only on depths would allow to semantically track the camera in very low-light conditions or even in the dark. More generally, we expect the performance of semantic tracking to improve alongside advances in labelling algorithms, our framework allowing for accommodating such foreseeable advances seamlessly.

Moreover, the semantic map attainable by our method seems to provide valuable cues to facilitate indoor scene understanding, as it would typically detect and coarsely localise major large-size scene structures such as floor, walls, tables, windows, and chairs; as well as several smaller objects like monitors, books, and keyboards. Such information comes together with a reliable confidence map that may be deployed effectively within an high-level reasoning process.

The first major limitation of the approach we describe here concerns speed. Indeed, while the semantic tracker can run comfortably at about $50 - 60$ $\mathtt{fps}$, the Fully Convolutional Network we currently rely upon for semantic labelling takes about 250 $\mathtt{ms}$ per frame, which brings down the overall frame rate of our system at about $3 - 4$ $\mathtt{fps}$. Yet, as our proposal is agnostic to the actual labeller, it will be feasible to investigate on alternative labelling approaches that may improve the processing speed without overly penalizing accuracy.

The second limitation deals with our current fixed-size volume approach, which hinders the possibility to semantically map workspaces of arbitrary sizes such as big rooms or even multiple rooms. To address this, in the next chapter we will show how a large-scale, voxel-based, reconstruction pipeline such as VoxelHashing by Nießner et al. [94] can be adapted, by deploying components of the systems previously described in this thesis, to provide semantically labelled reconstructions of room-sized environments whilst maintaining an interactive processing rate.

5 ONLINE LARGE-SCALE SEMANTIC FUSION

5.1 SUMMARY OF CONTRIBUTIONS AND RATIONALE

The previous chapters described a Semantic SLAM system allowing the fully automatic generation of semantically annotated maps of the environment (Chapter 3) and, jointly, exploiting the availability of such annotated reconstructions to improve the task of camera pose estimation (Chapter 4), thus realising what could be called a "SemanticFusion" virtuous loop. While promising, the described pipeline suffers from some challenges that make its deployment in concrete systems difficult. First and foremost, its reliance on a dense voxel grid as map storage medium prevents the reconstruction of environments bigger than a few cubic metres: we empirically determined that deploying volumes bigger than 8^3m^3 forces the choice of voxel sizes too big to generate a sufficiently detailed reconstruction of the scene.

A second drawback is the amount of time needed to semantically label each frame: current pixel labelling pipelines are not yet capable of processing every pixel in 640×480 images, such as those provided by a consumer RGB-D sensor, in real time. A system such as FCN [77], while effective in its core task of producing per-pixel semantic labels, still requires hundreds of milliseconds to process a VGA image. A consideration can be drawn from this: a system requiring semantic informations as core part of its algorithm cannot be run in real time. Not being able to run a dense SLAM pipeline, such as the one we derived from KinectFusion, with a sufficient speed causes issues when applying the described tracking algorithm: in Section 4.4 we mentioned that directly aligning the depth image (as well as the semantic maps) to the data stored in the volume is a reliable way to estimate the camera pose *iff* the assumption of small camera movements between frames holds. Clearly, our ability to process at most a few frames per second increases the average camera motion between subsequent RGB-D measurements, impairing our ability of accurately estimating the trajectory followed by the sensor. As a consequence, the quality of the reconstructed map of the world decreases and, being it relied on by the tracking algorithm to estimate the camera motion for subsequent frames, we enter in a vicious circle negatively affecting the SLAM pipeline in its entirety.

Recent research towards 3D reconstruction has delivered reliable and fast pipelines that allow attainment of accurate volumetric maps for large environments. On the other hand, the majority of such works were concerned with 3D geometry only, the advent of commodity RGB-D sensors having made this task remarkably affordable and effective. In this chapter we present a system that, by building on the ideas described in the previous chapters and being rooted on a large scale RGB-D reconstruction system (VoxelHashing, by Nießner et al. [94]), can deliver interactively and automatically a map for a large scale environment featuring both geometric as well as semantic information. We show how the significant computational cost inherent to deployment of a state-of-the-art deep network for semantic labelling does not hinder interactivity thanks to suitable scheduling of the workload on an off-the-shelf PC platform equipped with two GPUs.

With the proposed system, the user can explore the environment interactively by a hand-held RGB-D sensor. As in most previous work, this allows to attain a dense, detailed 3D reconstruction of the scene; peculiarly to our system, though, the resulting map is also endowed *online* and *fully automatically* with semantic labels determining the likelihood of each surface patch to depict objects of specific categories.

Driving factor behind the development of this pipeline is the need for a system whereby an untrained user may reliably scan and acquire semantically annotated 3D reconstructions of large indoor environments. As highlighted in Section 2.3, previous work such as [52, 117] would allow generation of similarly annotated 3D maps, though requiring proper interaction with a trained user. Conversely, to minimize user effort, we seamlessly integrate dense mapping and semantic labelling tasks into a single pipeline that can output detailed reconstructions of large scale environments wherein each voxel stores a complete probability mass function over a set of semantic categories of interest, akin to Section 4.3. Hopefully, this accomplishment may foster research on topics such as indoor scene understanding, object discovery and/or recognition, human/robot interaction and navigation.

The chapter is organised as follows: the next section discusses the proposed system, quantitative and qualitative results are provided in Section 5.3, while in Section 5.4 we will draw concluding remarks.

5.2 DESCRIPTION OF THE METHOD

The proposed pipeline is composed of two subsystems, each tailored to a specific task, controlled by a main engine handling all input/output operations and dispatching work to both. The two subsystems are:

LABELLING SUBSYSTEM: tasked with semantically labelling the RGB images gathered from the sensor.

SLAM SUBSYSTEM: dealing with camera tracking, map building and on-demand rendering of the reconstructed 3D scene from arbitrary viewpoints.

In the next paragraphs we will provide a detailed description of the above subsystems and then show how the main engine ties them together to attain the overall system.

5.2.1 *Labelling Subsystem*

This subsystem represents the interface of our pipeline to an image-based semantic labelling algorithm: given an input RGB image and, optionally, a depth map, this block provides per pixel confidences for a set of categories of interest, thus providing us with a full probability mass function across categories for each pixel of the input image.

We would like to point out the difference with labelling systems assumed to be provided in Chapter 3 and Chapter 4: there we deployed an interface exposing, for each pixel, a single category and an associated confidence. The rationale for such choice was to retain agnosticity w.r.t. the selection of the semantic labelling algorithm to deploy alongside the pipeline: different algorithms, providing differently label data could have their output converted in the intermediate format we described. The pipeline we describe in this chapter is tightly linked with the Fully Convolutional Newtork system by Long, Shelhamer and Darrell [77]. As before, we strive to maintain the design of the subsystem's interface independent from the choice of per-pixel labelling pipeline but, pragmatically, we decide to make use of *all* the informations that the semantic segmentation pipeline provides.

More specifically, given input images of size $w \times h$ and a set of N categories of interest, C, the output of this subsystem is a "volume" of confidences, L, of size $w \times h \times N$ and wherein each element (u, v, i) represents the confidence that the semantic labelling algorithm assigns to category i at pixel $\mathbf{u} = (u, v)$. We define a function $L(\mathbf{u}) : [0, w) \times [0, h) \to [0, 1]^N$ to denote such volume. Should a single label for a pixel $\mathbf{u}$ become necessary, a simple argmax operation over the N confidences $L(\mathbf{u})$ would provide the required output. In our system, though, we exploit the availability of multiple confidences at each image location to reconstruct a multi-label 3D map of the environment, wherein each voxel is endowed with information about the likeliness of *each* category of interest.

The interface just described is sufficiently generic that any labelling algorithm may in principle be incorporated within our pipeline. For instance, algorithms returning rectangular or polygonal ROIs with associated labels can have their output post-processed to paint each ROI in the volume "slice" associated to the correct category. Overlapping ROIs of the same category may also be handled, *e.g.* by applying a max operator to the confidence stored in each pixel, whereas overlapping regions of different categories can be drawn on the corresponding slices and a final per-pixel normalization can then turn the confidence values for each pixel in a proper probability mass function. Additionally, multiple labelling algorithms may be deployed, the only requirement being to run a normalization step independently on each pixel volume "column". Yet, to minimize the post processing necessary to obtain the labelled volume, semantic labelling algorithms providing directly per-pixel confidences across categories are inherently more amenable to our pipeline, nowadays the most effective and efficient proposals in this space relying on Deep Learning [39, 54, 77, 125].

As previously mentioned, in the final system we deploy the FCN approach [77] as we found experimentally that, in our settings, this architecture can provide quite clearly the best trade-off between classification accuracy and speed. Given an input RGB image, the pre-trained networks yield per-pixel confidences for a large number of categories (20, 40 or 60, depending on the specific model chosen) dealing with both indoor and outdoor objects. As the use case of our system concerns mapping indoor environments by a commodity RGB-D sensor, similarly to the previous chapters, we reduce the number of categories by dropping the output concerning unnecessary classes and applying per-pixel softmax normalization on the remaining raw scores, in order to convert the values into a proper probability mass function.

5.2.2 *SLAM Subsystem*

Generation of a semantic map of the observed environment is a task left to the SLAM Subsystem. A typical Simultaneous Localisation and Mapping pipeline consists of two main components: the first localises the camera within the environment by tracking its movements over time (*localisation* task); the second relies on the estimated camera pose to integrate the data provided by the sensor into the current representation of the scene (*mapping* task). Typically a third, optional, component is tasked with visualisation of the reconstructed scene to provide feedback to the user.

In the system presented in this chapter we add a fourth component to perform what we call the *semantic fusion* task, *i.e.* integration within the reconstructed scene of the semantic information provided by the Labelling Subsystem. This might also be seen as part of the standard *mapping* task but, as we will illustrate in Subsection 5.2.3, we split the standard SLAM mapping operation (integrating data from the RGB-D sensor) and the semantic mapping operation (integrating the information provided by the labeller), in order to decouple them and allow for deferred integration of the per-pixel category probabilities into the scene representation. Indeed, this approach is mandatory to enable on-line operation of the overall semantic reconstruction pipeline.

The SLAM subsystem adopted in our system is built on top of the VoxelHashing reconstruction pipeline by Nießner et al. [94] that, unlike KinectFusion [91], permits mapping of large workspaces by storing the map as a hash-based data structure instead of a dense voxel grid. For a detailed description of VoxelHashing we refer the interested reader to the original paper. In the following, we highlight the core of such system together with the main modifications required to store the semantic information, peculiar to our approach.

5.2.2.1 *Map generation and storage*

One main limitation of KinectFusion consists in its reliance on a dense, fixed-size voxel grid allocated onto GPU memory. In particular, each voxel stores a Truncated Signed Distance Function value together with weight and colour. Signed Distance Function values represent the distance of each voxel to the closest surface. As mentioned in Subsection 3.3.1, using a Truncated SDF, *i.e.* imposing a maximum value that the SDF is allowed to reach, yields a more accurate scene reconstruction when fusing measurements taken from multiple viewpoints. Voxels farther away from the closest surface than the Truncation Distance, on the other hand, bear no useful information to the reconstruction task.

Given a chosen voxel resolution, the number of voxels required to represent the scene grows cubically with its size, regardless of the actual contents of the environment. Hence, scaling KinectFusion to large workspaces is constrained by the amount of available GPU memory. As a matter of example, a quite typical 512^3 voxel grid would allow for representing a cubic workspace of $\approx 5^3 \mathrm{m}^3$ - such as a medium size room - with $1\,\mathrm{cm}^3$ voxel resolution on state-of-the-art GPU hardware. Most of the GPU memory, however, would be wasted to store truncation values associated with empty space.

To address the scalability issue, we rely on the VoxelHashing system [94],which employs a hash-table to quickly index and store in the GPU memory only those voxel blocks bearing useful information (*i.e.* “non truncated” SDF values). This approach has a threefold advantage in comparison to the original, dense, voxel grid:

a) the amount of available GPU memory constrains the size of the actively mapped area rather than that of the whole workspace. As a result, much larger scenes can be mapped using state-of-the-art hardware.

b) Typically, it turns out possible to rely on a smaller voxel size, which provides a far more resolute map of the observed scene.

c) Efficient swapping techniques are deployed to migrate portions of the scene currently not needed by the mapping process onto a secondary storage medium (*e.g.* from GPU memory to RAM and, eventually, to disk) and then, as soon as needed, back to GPU memory, thereby enabling, in principle, reconstruction of environments of any arbitrary size.

The hash-based data structure employed by the system allows to efficiently index a heap of data blocks, each block representing the voxels defining the map of a limited region of space. As in the previous chapters, each voxel in the map is endowed with three tokens of information:

- TSDF Value;
- Weight;
- Colour.

The base memory occupancy for a single voxel amounts then to 8 bytes. In our pipeline, we augment the data structure by a histogram $\Gamma \in [0,1]^N$ storing a probability mass function over a set of N categories, as in Section 4.3. Each bin represents the probability that an item of a specific category is located in the surface area associated with the voxel by encoding a floating point value in the interval $[0,1]$. To reduce memory occupancy, such values are stored into bytes by scaling the floating point number to the interval $[0,255]$. The final size of the voxel data structure thus increases by N bytes. The set of categories is application dependent and in our tests we employ 8 categories, having each voxel occupying 16 bytes, thereby doubling the memory footprint with respect to the standard data structure. Doubling the per-voxel memory occupancy would be worrying if we were using a dense data structure as deployed by

KinectFusion. Conversely, thanks to the reduced memory pressure allowed by VoxelHashing, we can easily accommodate such informations onto the GPU memory and, if necessary, move it back and forth with the system RAM via swapping operations.

When a new voxel is allocated by VoxelHashing, its histogram is set to the uniform probability, thus having each bin initialized to the value $255/N$, so to express maximum uncertainty on the type of object located within its boundaries.

5.2.2.2 *Surface rendering*

Visualization of the reconstructed scene is typically performed via raycasting. First, a synthetic range image is extracted: given a camera pose of interest, a ray is marched for each pixel of the output image from the camera centre until a positive to negative zero-crossing of the TSDF function is encountered, this signalling the presence of a surface, as described in Subsection 3.3.1.4. Clearly, marching a ray from the camera centre in a "non-dense" system such as the one here described is expensive, since the hash table has to be queried for every step, therefore several optimizations are described in the VoxelHashing paper [94]. The InfiniTAM pipeline [66] – another implementation of a large-scale, hash-based reconstruction system – details more enhancements to the raycasting operation that can sensibly speed up the computation.

The raycasted range map can then be used to extract a coloured representation of the environment by performing trilinear interpolation of the RGB values associated to the 8 voxels closest to each zero crossing point. If speed is a priority, one can even avoid the interpolation step altogether and use the colour associated to the single voxel closest to the raycasted 3D point, at the expense of a slightly inaccurate colouring of the generated image. Point normals can be computed by either numerically estimating the gradient of TSDF values in the correspondence of the rendered point or by computing the cross product of vectors connecting neighbouring pixels in the range map, the latter method being faster but slightly less accurate.

Semantic labels for each rendered point can then be extracted. In order to determine the label of a single pixel, we apply an argmax operation over the N histogram bins associated to each raycasted

3D point and store the resulting label in an output category map and the associated confidence in an output probability map. While we could trilinearly interpolate between bins associated to the histograms of 8 neighbouring voxels, in order to obtain an interpolated histogram to subject, in turn, to the argmax operation, in practice we consider only the voxel whose centre is closest to the candidate 3D point, on account that, typically, object categories are "large scale" scene attributes and the interpolation of neighbouring voxel probabilities would not provide much additional information while notably slowing down the processing speed of the pipeline. Figure 5.1a provides an exemplar image obtained by raycasting into the current camera view the most likely label for each voxel and subsequently applying a shading pass.

Our category representation scheme allows also to render the likeliness of a specific category in each voxel observable from the current camera viewpoint. Indeed, we can provide visualizations detailing the spatial distribution of a certain category of interest, by selecting the histogram bin associated to that category in every voxel identified by the raycasted range image. For example, as shown in Figure 5.1b, we might wish to render the "chairness" of the reconstructed scene. It is worth observing that, although some surface patches belonging to chairs are mislabelled in Figure 5.1a, the right image provides evidence that these may indeed belong to chairs, this information likely to help performing an higher lever task such as segmenting out all the chairs present in the scene.

Once the range, normal, colour, category and score maps are extracted, shading can be applied to obtain pleasant visualizations, as in Figure 5.1. While the renderings just described may convey useful informations to the user, all but those dealing with the range and normal maps are optional in our pipeline and can thus be disabled to increase processing speed. The renderings of the range and normal maps, instead, are pivotal in the camera localisation step that will be described next.

5.2.2.3 *Camera localisation*

Camera localisation is an essential step of the SLAM pipeline: to integrate RGB-D frames coming from the sensor into the global map

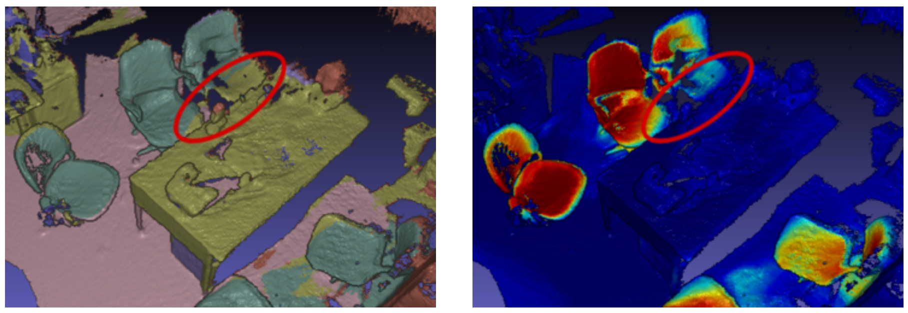

(a) Semantic labels (b) "Chairness" of the scene

Figure 5.1: Left: semantic labels assigned to the reconstruction of an office environment. Right: heat map showing the spatial distribution of "chair" objects. It can be seen how, even in presence of mislabelled areas (highlighted by the red ellipse), the "chair" confidence is not null and thus may help to segment out chairs.

of the environment, one has to know the pose from which the camera captured such informations. Typically, thanks to the availability of RGB-D sensors providing useful informations at 30 Hz and the high processing rate of systems such as KinectFusion/VoxelHashing, the task of sensor localisation can be simplified into a task of camera tracking, where the pose of the sensor at a given moment of time is computed in relation to the previous camera pose, therefore requiring only estimation of an incremental rigid body transformation (that, given the real-time processing rate nature of the system, is likely very close to the identity transform).

Several approaches to camera tracking task have been proposed in the literature related to KinectFusion, either relying on purely geometric clues, such as the projective ICP approach of the original paper [91] and the direct alignment methods described in [14, 17] and considered in Chapter 3 and Chapter 5, or aimed at deploying colour information to maximize the photo-consistency between pairs of consecutive frames [109, 119] or between the current frame and the colour information stored into voxels [13].

In the system described in this chapter we employ the projective ICP approach by Newcombe et al. [91] to estimate sensor pose. The method relies on the raycasted depth and normals map as seen from the previous camera pose and the current depth map. An iterative process performs a projective association between points in the current and in the raycasted range maps [10], computes an energy term at each pixel based on the point to plane metric [22], and finally minimizes the sum of all pixel energies by linearising such function in a neighbourhood of the previous pose and computing an incremental transformation using the Lie algebra representation [11, 110].

5.2.2.4 *Semantic Fusion*

Finally, the Semantic Fusion component is tasked with the integration into the voxel-based map of the environment of per-pixel semantic labels extracted from an input RGB-D frame by the Labelling subsystem. As mentioned earlier, this task is kept disjoint from the canonical "fusion" operation performed by the reconstruction pipeline, to allow the labeller to work asynchronously with respect to the SLAM process, thus not hindering the real-time nature of the latter due to the former being significantly slower.

Once a frame has been labelled, its associated pose, $T_{w,c}$, estimated by the camera localisation component, is retrieved and can be used to perform the actual fusion step. Akin to the integration of the RGB-D image, the process is applied only to those voxels that fall into the camera frustum and are "close enough" to the surface described by the depth frame associated with the previously extracted labels (that had been cached at the beginning of the labelling phase); purposely, we employ the same truncation distance δ used by the depth fusion step.

More precisely, as a first step, the location of each mapped voxel block[1] in the world coordinate frame is transformed in the appropriate camera reference frame by the inverse transformation $T_{w,c}^{-1}$ described by the sensor pose. The transformed block centre is then projected onto the image plane and, if the resulting coordinates lie inside the image bounds, the block is marked as potentially visible and thus to be updated. Thanks to the GPU, this first step can be efficiently carried out in parallel by associating a thread to each block. A *scan-and-compact* operation is then performed to gather the indices of all the blocks to be updated in a single buffer, which in turn is used to launch an update thread for each voxel residing in such blocks.

1 Wherein the original KinectFusion algorithm each voxel is processed independently, in the hash-based implementation of VoxelHashing the minimal unit of processing is the voxel *block*, as allocated in the heap.

Each voxel centre $\mathbf{p_v}$ is then projected onto the depth frame by applying the $T_{w,c}^{-1}$ transformation and the depth camera intrinsics to obtain the corresponding pixel coordinates $\mathbf{u}$, as in Equation 3.2. Its associated depth $d(\mathbf{u})$ is then sampled from the cached range map: *iff* the 3D point determined by such depth is sufficiently close to the voxel itself ($|s_v| \leqslant \delta$, see Equation 3.3), then the label probabilities vector $\Gamma(\mathbf{p_v})$ stored in the voxel is subject to the update operation that will be described next.

To integrate the pixel category probabilities provided by the labelling algorithm into the probability histogram stored in each voxel, we perform an operation akin to the running average adopted during the depth integration step, in turn followed by a re-normalization step to ensure attainment of a valid probability mass function. We denote as $L(\mathbf{u}) \in \mathbb{R}^N$ the p.m.f. computed for the pixel $\mathbf{u}$ and as $\Gamma(\mathbf{p_v}) \in [0,1]^N$ the p.m.f. corresponding to the voxel being updated. In the weighted average operation performed to fuse the probability values we deploy unitary weights for the new measurements, thus employing the same strategy used during the depth integration phase. We then sample the weight $\Omega(\mathbf{p_v})$ stored in the voxel, representing the (possibly clamped to a maximum value) number of fusion operations already performed on it. Elements of a (not yet normalised) updated histogram $\bar{\Gamma}(\mathbf{p_v})$ are computed as follows:

$$\bar{\Gamma}^i(\mathbf{p_v}) = \frac{\Gamma_t^i(\mathbf{p_v})\Omega(\mathbf{p_v}) + L^i(\mathbf{u})}{\Omega(\mathbf{p_v}) + 1} \qquad \text{with } i \in [1, N] \tag{5.1}$$

We then normalize the values so as to attain a valid probability mass function, $\Gamma_{t+1}(\mathbf{p_v})$, that is stored back into the voxel:

$$\Gamma_{t+1}^i(\mathbf{p_v}) = \frac{\bar{\Gamma}^i(\mathbf{p_v})}{\sum_i \bar{\Gamma}^i(\mathbf{p_v})} \qquad \text{with } i \in [1, N] \tag{5.2}$$

We do not update the weight associated to the voxel, leaving that task to the depth fusion component. While weights $\Omega(\mathbf{p_v})$ depend on the number of times a specific voxel $\mathbf{p_v}$ has been observed, and that number typically differs from the amount of times a voxel has been semantically labelled, we found no significant difference between using an ad-hoc semantic weight (that would need to be stored alongside the p.m.f. histogram) and just piggy-backing on the already present weight associated to the TSDF. Hence, we exploit the $\Omega(\mathbf{p_v})$ values to give an appropriate strength to the past probabilities and, by performing a weighted average during the update operation, prevent single measurements from significantly changing the p.m.f. Also, as typically the frame rate of the SLAM subsystem is constant and the time required by the labelling algorithm is also deterministic, the relationship between the depth weight (a clamped counter of the number of frames that have been integrated) and an ad-hoc "semantic weight" would be linear.

5.2.3 *Main engine*

The Main Engine of the system described in this chapter interacts with the RGB-D sensor, dispatches the work to the SLAM and Labelling subsystems, and provides the user with feedback on the ongoing operation by displaying rendered images depicting the reconstructed environment.

One of the key novelties of our proposal is its ability to perform SLAM and semantic mapping fully automatically and on-line. This means that, while the user moves around the RGB-D sensor, she/he would see on the screen a semantic reconstruction of the workspace created incrementally at interactive frame-rate. In other words, while in KinectFusion/VoxelHashing the user would perceive incremental reconstruction of the geometry of the scene interactively, and by deploying the systems described in the previous chapters she/he would observe semantic reconstructions of the environment but with low processing speed; the system here described is aimed at providing, just as interactively, both geometry and semantics in the form of surfaces tagged with category labels. Processing speed is therefore of paramount importance to the pipeline as a whole. While the SLAM Subsystem can comfortably keep-up with the 30 Hz RGB-D stream delivered by the sensor, state-of-the-art deep networks for semantic labelling require hundreds of milliseconds or even seconds to process a single frame.

This state of affairs mandates the two subsystems to be decoupled so to execute their code in parallel and prioritize SLAM to provide interactive feedback to the user. The Labelling subsystem is thus run on the remaining CPU and GPU time and, by exploiting the deferred Semantic Fusion algorithm described in Subsection 5.2.2.4, its output is integrated into the voxel map as soon as it becomes available. To obtain an even higher throughput, we deploy the two subsystems onto different GPUs, the Main Engine performing the appropriate copies or movements of the data to and from the different cards.

While all frames captured by the RGB-D sensor are used to perform the SLAM task (necessary to obtain an accurate map and a reliable camera localisation), only a minority of those are provided to the Labelling Subsystem. Choice of such candidate frames is left to the Main Engine and in this case performed in a greedy fashion, by ignoring all frames grabbed while the Labelling Subsystem is busy processing a frame and marking for labelling the first new frame received after the labeller has finished its work on a previous one. We elected not to use a queue-based system to privilege the labelling of several areas of the environment instead of rapidly filling the queue with similar frames acquired by nearby viewpoints thus having the labeller unavailable to process newly explored regions of the environment.

Figure 5.2 shows a sequence diagram detailing the execution flow of the system. Once an RGB-D frame is grabbed by the sensor, its data is copied to both GPUs and the labelling thread is activated. At the same time, the camera pose from which the scene was observed is estimated by the SLAM Subsystem and stored in a pose database together with an associated time-stamp, used to retrieve such pose at a later stage; subsequently, depth and colour information are fused into the hash-based TSDF structure. A raycasting operation is then performed to obtain the range and normals maps required by the projective ICP algorithm to localise the camera at the next iteration; if semantic visualization is desired then colour, label, and confidence maps are rendered as well.

The main engine then verifies if the labelling algorithm has terminated its computation; if not, another iteration of the SLAM pipeline is executed. Conversely, the labelling output (i.e. the $w \times h \times N$ volume storing per-pixel category probabilities described in Subsection 5.2.1) is transferred from the labelling GPU to

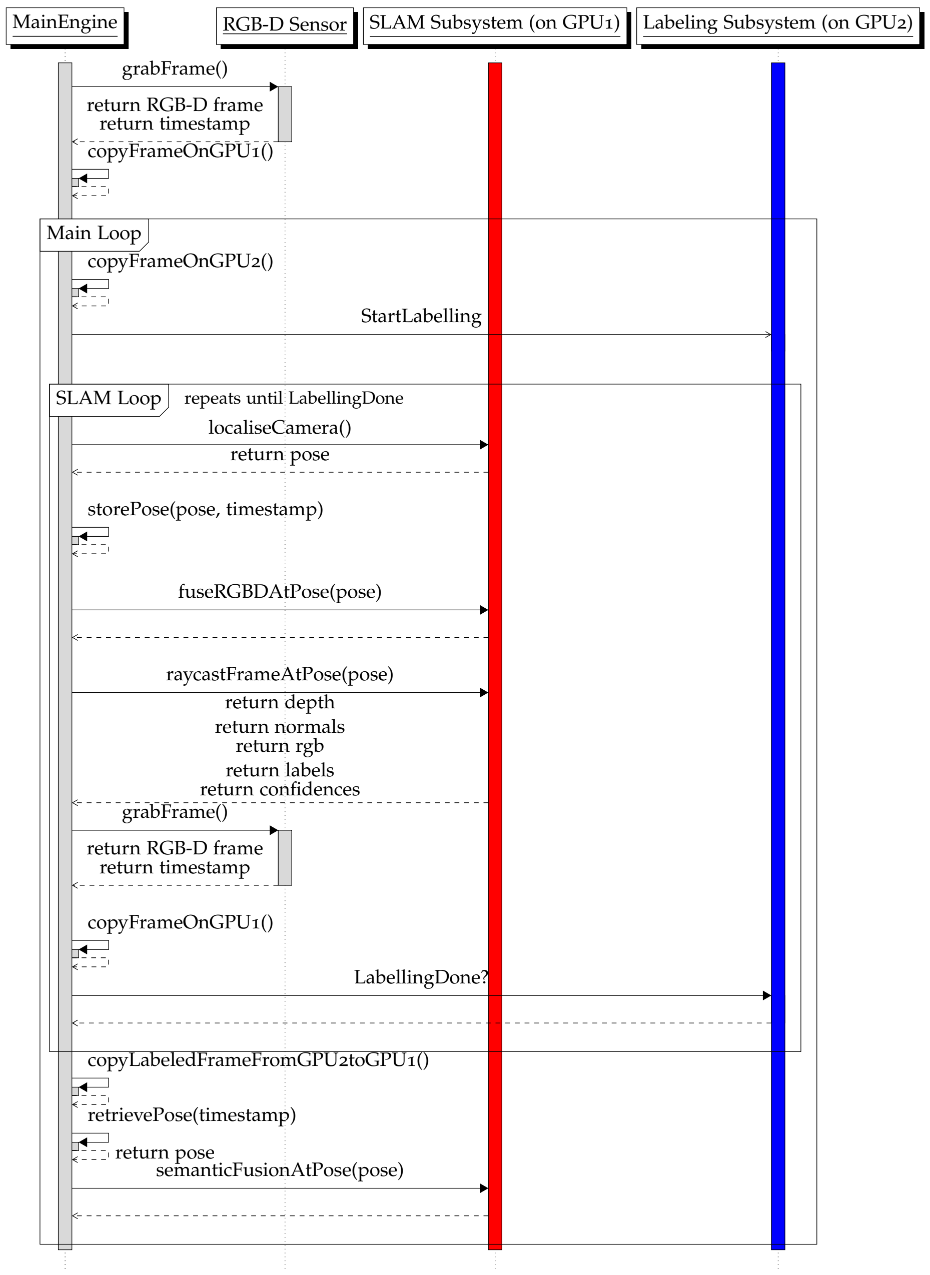


Figure 5.2: Sequence diagram depicting the execution flow of the proposed system. The SLAM and Labelling subsystems are deployed on two different GPUs, here colour coded in red and blue. The main engine moves data between the host memory and the two GPU memories as needed.

the SLAM GPU and the viewpoint from which the labelled frame had been observed is retrieved from the pose database according to its time-stamp. Semantic labels are then fused through the algorithm described in Subsection 5.2.2.4.

The process is repeated until the user wishes to terminate; at that point the entire map of the environment can be saved as a mesh by application of the marching cubes algorithm [78]. The mesh can be coloured using either the RGB values stored in the hash-based map or a colour mapped representation of category or confidence values.

5.3 RESULTS

The system presented in this chapter pursues interactive and fully automatic semantic mapping of large workspaces. In this section we will show quantitative and qualitative results provided by the system. Firstly, we present an evaluation of the computational requirements of the entire system, detailing the overall impact of the two main subsystems. Afterwards, we show qualitative results depicting the kind of semantic reconstructions that can be achieved by running the system.

5.3.1 *Performance evaluation*

The system here described relies on computation modules deployed on both the CPU and two graphics processors. Our testing setup consists in a PC equipped with a Intel Core i7 4960X CPU and two GeForce Titan Black graphics cards (each with 6GB of dedicated memory). The SLAM Subsystem is deployed on one card, while the Labelling Subsytem based on a Fully Convolutional Network [77] is deployed on the other (the amount of GPU memory required by the neural network is ≈ 5.5 GB).

Table 5.1 shows the average time spent in the main components of the pipeline. It is evident how the most computationally intensive component of the proposed system is that concerned with semantic labelling of input frames and how its decoupling and deployment on a separate graphics processor is necessary to maintain an inter-

	Times (ms.)		
	Multiple GPU		Single GPU
Algorithm Section	GPU 1	GPU 2	
Frame Grabbing + Preprocessing	10.27	–	20.21
Camera Localisation	4.30	–	24.90
Depth + RGB Fusion	8.79	–	16.96
ICP Raycast (Depth + Normals)	5.16	–	13.35
RGB + Labels + Confidence Raycast	6.49	–	23.79
Shading + GUI update	11.32	–	71.64
Other processing	7.70	–	10.26
SemanticFusion*	9.59	–	10.44
Frame Labelling	–	284.09	438.91
Total time per frame*	57.20	284.09	186.55

Table 5.1: Processing time broken down by component. *Total time per frame does not include the exact time spent in executing the SemanticFusion step, as this is performed only after the Labeling Subsystem terminates processing an input RGB-D image, and therefore its execution time is amortized over a larger number of frames. The total time per frame is thus the average time spent to process a frame, yielding a frame-rate of 17.48 fps for the multi GPU system and of 5.4 fps for the single GPU setup.

active frame-rate (≈17 Hz) and allow users to reliably deploy the system to semantically reconstruct a location.

For comparison, in the last column we show the processing times that can be obtained by running the proposed pipeline on a single GPU accelerator. For this test, we employed a workstation with an Intel Core i7 4930K CPU and a Tesla K40 GPU, providing 12 GB of graphics memory. The availability of a single GPU, even with twice the memory as those deployed in the previous test, severely hinders the overall system speed due to the computation being bound by the number of CUDA cores rather than by amount of available memory, this bringing evidence towards the idea of deploying two separate GPUs to realize our system.

5.3.2 *Qualitative results*

Our system enables the user to interactively attain a semantic reconstruction of the environment by employing a hand-held commodity RGB-D sensor such as the Kinect. In this section we show exemplar results obtained in different environments.

Several office sequences depicting a variety of indoor objects such as "monitors", "chairs", "tables", etc... were acquired and processed by our system. Figure 5.3 shows a view from one of such sequences where chairs, monitors, and the keyboard are labelled quite correctly. Wall and floor regions are also mostly correct while the "table" category shows a slightly lower segmentation accuracy, its labels bleeding into the "cabinet" located below. The Fully Convolutional Network model used to obtain the depicted results is "pascalcontext-fcn8s".

On a second video sequence, depicted in Figure 5.4, we were interested on a different set of semantic categories, including persons. To achieve a better labelling, we replaced the pre-trained "pascalcontext-fcn-8s" network used to label the earlier example, with the "voc-fcn-8s" network, also provided by the authors of the Fully Convolutional Networks paper [77]. The two models show different sensitivity towards separate labels: indeed, while in Figure 5.3 the desk category has an overall satisfactory detection rate,

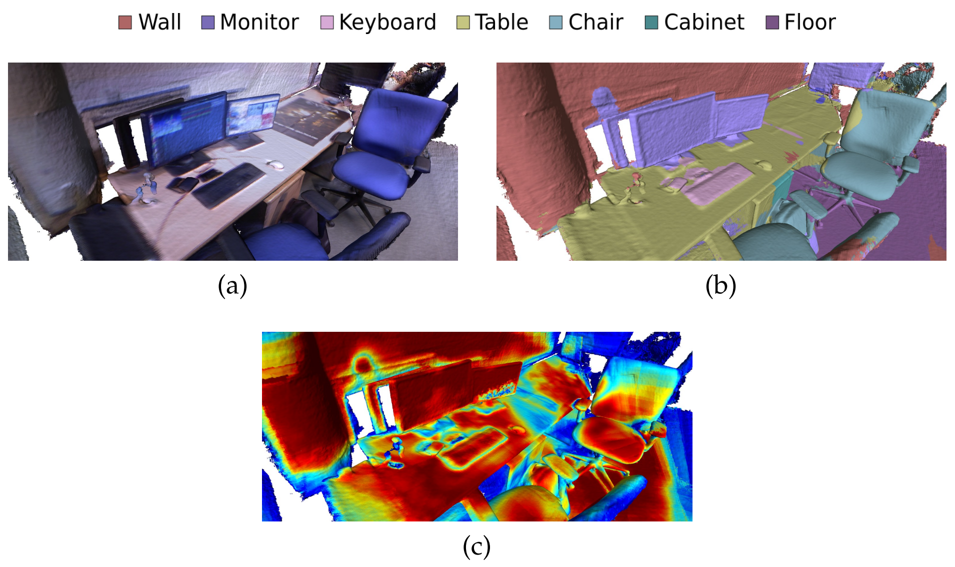


Figure 5.3: A semantically reconstructed office environment. (a) shows the coloured mesh generated by our system. (b) shows the most likely category label for each voxel, while the associated confidence values are displayed as a heat map in (c).

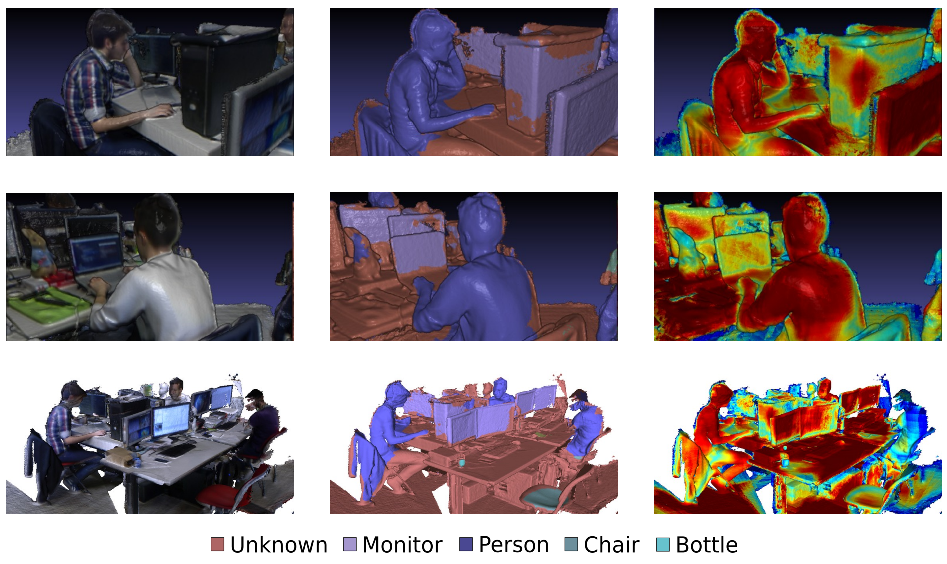


Figure 5.4: Office sequence depicting people working in front of computer monitors. Top two rows: raycasted views obtained while processing the sequence. Bottom row: detail of the mesh extracted after the reconstruction. The left column represents a coloured reconstruction of the scene. The central column shows the semantic labels associated to the surfaces according to the legend in the last row. The right column depicts the confidences associated to the selected labels, as heat maps wherein the blue-to-red gradient indicates increasing probabilities.

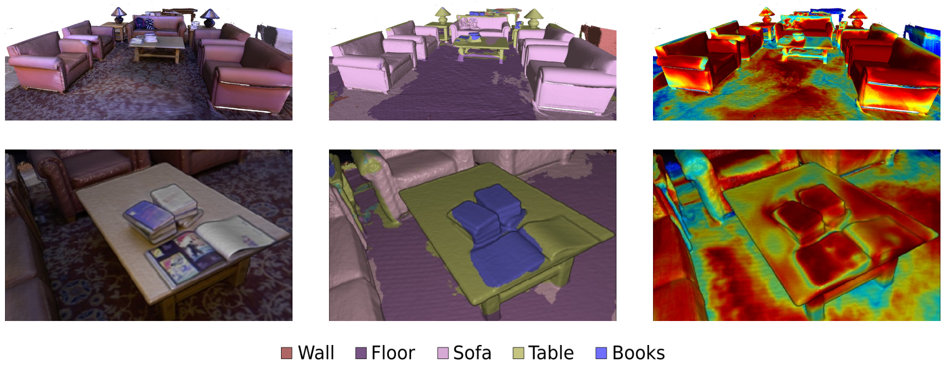


Figure 5.5: *ReadingRoom* sequence from the Stanford 3D Scene Dataset [128]. Top row: raycasted views acquired during the reconstruction. Bottom row: details from the final reconstruction. Columns as in Figure 5.4.

in Figure 5.4 the desks are not labelled at all, their voxels being instead assigned the "unknown" category.

Using the same neural network as in Figure 5.3, we have run our system also on sequences belonging to the Stanford 3D Scene Dataset by Zhou and Koltun [126] and Zhou, Miller and Koltun [128]. In Figure 5.5 and Figure 5.6 we show the resulting reconstructions. It can be observed how the system can label correctly large objects, such as sofas and tables. Moreover, voxels pertaining to smaller objects, such as the stacked books in Figure 5.5, are mostly correctly labelled alike. The two lamps in Figure 5.5 (top row) are inevitably mislabelled because "lamp" does not belong to the set of categories handled by the neural network. As concerns the potted plant in Figure 5.6, it is worth pointing out that labels tend to propagate into the wall due to the thin and partially reflective nature of the leaves, which causes depth estimation by the RGB-D sensor to fail and prevent the generation of an accurate 3D reconstruction of the object.

Throughout our experiments, we observed that the boundaries between different objects are reasonably accurate for smaller items (*e.g.* books, keyboards, chairs, monitors,...) while the contours dealing with larger objects, such as tables, sofas and structural elements (*i.e.* walls, floors and ceilings) tend to be less accurately localised and "bleed" onto neighbouring voxels. It is noteworthy, though, that the confidence associated to such incorrect boundary zones is typically significantly lower than that estimated within the internal portions

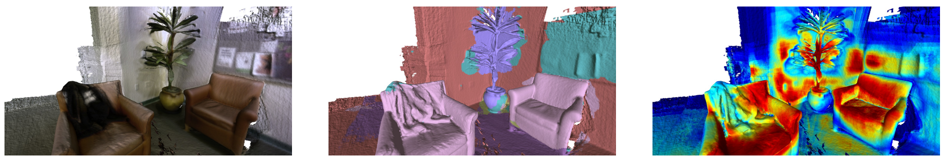


Figure 5.6: Details of the reconstruction of the *Lounge* sequence from the Stanford 3D Scene Dataset [126]. Columns as in Figure 5.4.

of objects. This effect is especially evident if the object is observed from a significant distance and can be traced back to the 2D nature of the semantic labelling process: labelled pixels are back-projected onto the voxel-based map by using the current depth image and thus a single (potentially mis-)labelled pixel affects a larger area of the reconstruction the farther away it is from the camera.

Overall, the experimental findings suggest that our system can label correctly both the main large-size scene structures such as floor, walls, tables, chairs as well as several smaller objects like monitors, books, keyboards. Moreover, the confidence maps turn out quite reliable, due to high probability labels unlikely turning out wrong and mislabelled areas featuring low scores. Therefore, our semantic reconstructions and associated confidence maps may provide valuable cues to facilitate high-level reasoning pursuing indoor scene understanding.

5.4 FINAL REMARKS

In this chapter we have presented an interactive system allowing an user to perform 3D reconstruction of a large scale environment while seamlessly performing semantic labelling the observed surfaces. To this purpose, by building upon the considerations and results described in the previous chapters, we split the proposed SLAM pipeline in two subsystems, each relying on state of the art approaches.

The Labelling Subsystem pursues per-pixel semantic segmentation of RGB-D images by a recently proposed deep neural network [77]. Indeed, our architecture is agnostic to the choice of actual labeller and, thus, holds the potential to accommodate the advances in the field, likely to be provided by the ever-increasing research efforts on deep learning architectures for semantic segmentation and object detection.

The SLAM Subsystem relies on the VoxelHashing approach [94] to handle reconstruction of large scenes. The pipeline described by Nießner et al., though, has been modified to achieve storage, deferred integration and visualization of semantic informations. In addition, to provide the user with a fluid and interactive experience, we deploy the proposed system on an off-the-shelf Personal Computer endowed with two GPUs and suitably schedule the workload on such platforms. A supplementary video[2] demonstrates the pipeline in operation. From the video, one may perceive the effect of the deferred semantic fusion and how this approach does not hinder incremental interactive reconstruction, while adding semantic information into the map over time.

An issue worthy of further investigation concerns the accuracy of the semantically labelled 3D maps: sometimes labels bleed onto voxels belonging to neighbouring objects due to the independence of category histograms stored in the voxel blocks as seen, for example, in Figure 5.5. Accordingly, the application of pairwise CRFs as described for example by Valentin et al. [117], either at label integration or mesh generation time so as to ensure spatial consistency of neighbouring labels would likely improve the appearance of the obtained reconstructions.

Among the shortcomings of our system is, additionally, reliance of the camera localisation step on a purely geometric tracking approach (*i.e.* the standard projective ICP technique used by VoxelHashing and KinectFusion) which, while good enough to accurately estimate camera poses across frames, is not immune from a certain amount of drift that may become evident when the hand-held sensor is brought back to a previously observed area. Frame-to-model alignment methods such as this one and the others described also, as mentioned in the previous chapter, rely on the assumption of small movements of the camera between subsequent frames. Were the user to subject the sensor to a sudden movement, tracking algorithms based on iterative minimization of a cost function would inevitably estimate incorrect poses, thus causing the fusion

2 https://vision.disi.unibo.it/~tcavallari/phd_thesis/gmdl2016.mp4

of the RGB-D measurement in the wrong location, damaging the map being reconstructed.

Systems such as the one described by Kähler, Prisacariu and Murray [64] allow effectively to detect such tracking failures, by deploying a Support Vector Machine to classify outputs of the ICP camera alignment operation *e.g.* by considering the number of inliers in the ICP algorithm or the residual energy. When tracking failures are detected, SLAM systems such as KinectFusion/VoxelHashing are switched to a so-called "re-localisation mode". In this mode of operation the fusion part of the pipeline is disabled to prevent the integration of incorrect data in the reconstruction. The system then attempts to estimate the absolute pose of the camera in the world by using informations already stored in the map and, if successful, restarts the SLAM pipeline from such pose. A naïve way to reinitialise the system is by continuously attempting to perform ICP alignment between the current depth frame and the depth map obtained by raycasting the contents of the TSDF from the last known camera pose, asking the user to carefully position the sensor to observe the same location in the world and, hopefully, have the ICP alignment succeed. More advanced techniques exist and, in the next part of the thesis, we will describe such a system, allowing the effective estimation of arbitrary camera poses in an environment being reconstructed.

Part II

CAMERA RELOCALISATION

6 APPEARANCE-BASED CAMERA RELOCALISATION

6.1 SUMMARY OF CONTRIBUTIONS

The SLAM pipelines described in the previous chapters are all concerned with semantic camera localisation (Chapter 4) and reconstruction (Chapter 3 and Chapter 5) of scenes explored by an user holding an RGB-D sensor. Camera pose estimation is indeed an important problem in computer vision, with applications not only in simultaneous localisation and mapping (SLAM) [66, 87, 91], but also in the virtual and augmented reality [3, 18, 52, 96, 97, 117] and navigation [72] fields. In SLAM systems such as those already mentioned, the camera pose is commonly initialised upon starting reconstruction and then tracked from one frame to the next (possibly relying on the presence of semantic clues, as in Chapter 4), but tracking can easily be lost due to *e.g.* rapid movement or textureless regions in the scene; when this happens, it is important to be able to relocalise the camera with respect to the scene, rather than forcing the user to start the reconstruction again from scratch.

Accurate relocalisation is also crucial for loop closure when trying to build globally consistent maps [44, 64, 121]. Common techniques either match the current image captured by the sensor against keyframes with known poses coming from a tracker, or establish 2D-to-3D correspondences between keypoints in the observed image and points in the reconstructed scene in order to estimate the camera pose. Recently, regression forests have become a popular alternative to establish such correspondences. They achieve accurate results, but must be trained offline on the target scene, preventing relocalisation in new environments.

In this chapter, we show how to circumvent this limitation by adapting a pre-trained forest to a new scene on the fly. The adaptation process allows a forest to learn how to regress 3D world coordinates for surface patches observed in a stream of input RGB-

D frames, thus associating appearance features to locations in the environment being reconstructed. Even though the process through which the camera pose is estimated does not explicitly rely on "semantic" informations, as the tracker in Chapter 4; it is easy to see how the semantics of the reconstructed scene (*i.e.* the kinds of objects observed, and their spatial locations) are key to the actual relocalisation process. For example, an image patch depicting a chair will be likely associated to a set of candidate 3D coordinates in the scene coherent with the locations of all the chairs observed during the forest adaptation process. Our adapted forests achieve relocalisation performance that is on par with that of offline trained forests, and our approach runs in under 150 ms, making it desirable for real-time systems that require online relocalisation.

The system here described can be deployed as part of a Semantic-Fusion pipeline in order to allow seamless recovery from several tracking failure situations, thus being amenable to be used by users not trained in the "fine art" of 3D reconstruction systems based on continuous camera tracking (*i.e.* users that may move the sensor quickly or, by not knowing the limitations of ICP-based tracking methods, observe surfaces with few geometric features).

6.2 RATIONALE AND STATE OF THE ART

Traditional approaches to camera relocalisation have been based around one of two main paradigms:

IMAGE MATCHING METHODS match the current image from the camera against keyframes stored in an image database (potentially with some interpolation between keyframes where necessary). For example, Gálvez-López and Tardós [48] describe an approach that computes a bag of binary words based on BRIEF descriptors [16] for the current image and compares it with bags of binary words for keyframes in the database using an L1 score. Gee and Mayol-Cuevas [50] estimate camera pose from a set of synthetic (*i.e.* rendered) views of the scene. Their approach is interesting because unlike many image matching methods, they are to some extent able to relocalise from novel poses; however, the complexity increases linearly with the number of synthetic views needed, which poses significant limits to practical use. Glocker et al. [51] encode frames using Randomised Ferns which, when evaluated on images, yield binary codes that can be matched quickly by their Hamming distance: as noted in [76], this makes their approach much faster than [50] in practice.

KEYPOINT-BASED METHODS find 2D-to-3D correspondences between keypoints in the current image and 3D scene points, so as to deploy *e.g.* a Perspective-n-Point (PnP) algorithm [57] (on RGB data) or the Kabsch algorithm [63] (on RGB-D data) to generate a number of camera pose hypotheses that can be pruned to a single hypothesis using RANSAC [45]. For example, Williams, Klein and Reid [122] recognise/match keypoints using an ensemble of randomised lists, and exclude unreliable or ambiguous matches when generating hypotheses. Their approach is fast, but needs significant memory to store the lists. Li and Calway [76] use graph matching to help distinguish between visually-similar keypoints. Their method uses BRISK descriptors [74] for the keypoints, and runs at around 12 fps. Sattler, Leibe and Kobbelt [102] describe a large-scale localisation approach that finds correspondences in both the 2D-to-3D and 3D-to-2D directions before applying a 6-point DLT algorithm [57, 112] to compute pose hypotheses. They use a visual vocabulary to order potential matches by how costly they will be to establish.

Some hybrid methods use both paradigms. For example, Mur-Artal, Montiel and Tardós [86] describe a relocalisation approach that initially finds pose candidates using bag of words recognition [49], which they incorporate into their larger ORB-SLAM system (unlike [48], they use ORB [99] rather than BRIEF features, which they found to improve performance). They then refine these candidate poses using PnP and RANSAC. Valentin et al. [115] present an approach that finds initial pose candidates using the combination of a retrieval forest and a multiscale navigation graph, before refining them using continuous pose optimisation.

Several less traditional approaches have also been tried. Kendall, Grimes and Cipolla [69] train a convolutional neural network to directly regress the 6D camera pose from the current image. Deng et al. [33] match a 3D point cloud representing the scene to a local

3D point cloud constructed from a set of query images that can be incrementally extended by the user to achieve a successful match. Lu et al. [80] perform 3D-to-3D localisation that reconstructs a 3D model from a short video using structure-from-motion and matches that against the scene within a multi-task point retrieval framework.

Recently, Shotton et al. [104] proposed the use of a regression forest to directly predict 3D correspondences in the scene for all pixels in the current image. This has two key advantages over traditional keypoint-based approaches: (i) no explicit detection, description or matching of keypoints is required, making the approach both simpler and faster, and (ii) a significantly larger number of points can be deployed to verify or reject camera pose hypotheses. However, it suffers from the key limitation of needing to train a regression forest on the scene *offline* (in advance), which prevents on-the-fly camera relocalisation.

Subsequent work has significantly improved upon the relocalisation performance of [104]. For example, Guzman-Rivera et al. [55] rely on multiple regression forests to generate a number of camera pose hypotheses, then cluster them and use the mean pose of the cluster whose poses minimise the reconstruction error as the result. Valentin et al. [116] replace the modes used in the leaves of the forests in [104] with mixtures of anisotropic 3D Gaussians in order to better model uncertainties in the 3D point predictions, and show that by combining this with continuous pose optimisation they can relocalise 40% more frames than [104]. Brachmann et al. [12] deploy a stacked classification-regression forest to achieve results of a quality similar to [116] for RGB-D relocalisation. Massiceti et al. [82] map between regression forests and neural networks to try to leverage the performance benefits of neural networks for dense regression, while retaining the efficiency of random forests for evaluation. They use robust geometric median averaging to achieve improvements of around 7% over [12] for RGB localisation. However, despite all of these advances, none of these papers remove the need to train on the scene of interest in advance.

In the remainder of this chapter, we show that the need for *offline* training on the scene of interest can be overcome through *online* adaptation to a new scene of a regression forest that has been pre-trained on a generic scene. We achieve genuine on-the-fly relocalisation similar to that which can be obtained using keyframe-based approaches [51], but with both significantly higher relocalisation performance in general, and the specific advantage that we can relocalise from novel poses. Indeed, our adapted forests achieve relocalisation performance that is competitive with offline-trained forests, whilst requiring no pre-training on the scene of interest, and relocalising in close to real time. This makes this approach a practical and high-quality alternative to keyframe-based methods for online relocalisation in novel scenes, suitable to be deployed in large scale reconstruction pipelines to allow a more robust user interaction during the mapping process. The next sections are structured as follows: first, in Section 6.3 we will describe the method we employ to perform pose regression from a continuous stream of RGB-D frames; then, in Section 6.4 we will evaluate the performances of the proposed system, both quantitatively and qualitatively.

6.3 ON-THE-FLY ADAPTATION OF POSE REGRESSION FORESTS

6.3.1 *Overview*

A high-level overview of our approach is shown in Figure 6.1. Initially, we train a regression forest *offline* to predict 2D-to-3D correspondences for a *generic* scene using the approach described in [116]. To adapt this forest to a new scene, we remove the contents of the leaf nodes in the forest (*i.e.* GMM modes and associated covariance matrices) whilst retaining the branching structure of the trees (including learned split parameters). We then adapt the forest *online* to the new scene by feeding training examples down the forest to refill the empty leaves, dynamically learning a set of leaf distributions specific to that scene. Thus adapted, the forest can then be used to predict correspondences for the new scene that can be used for camera pose estimation. Reusing the tree structures spares us from expensive offline learning on deployment in a novel scene, allowing for relocalisation on the fly.

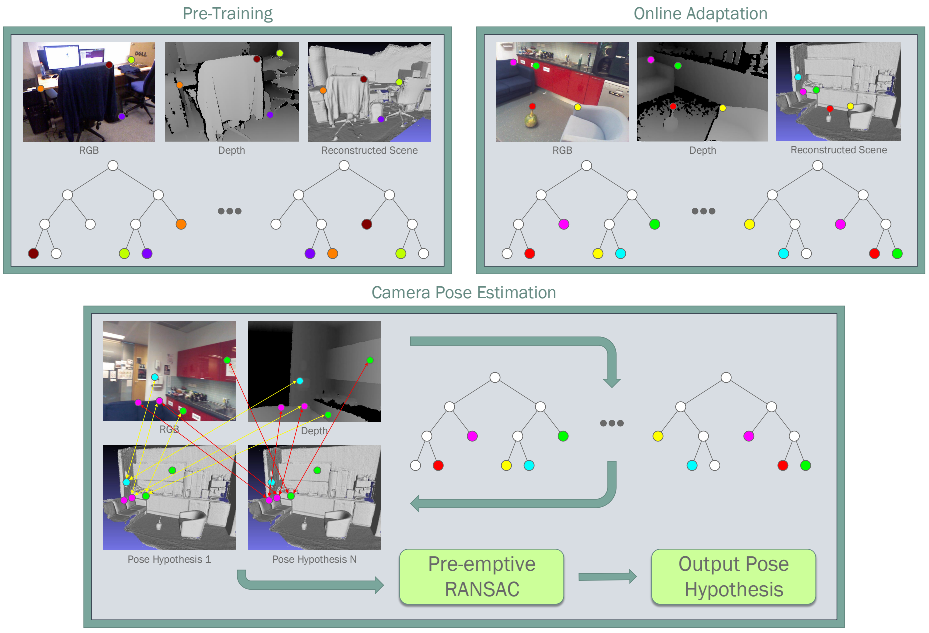


Figure 6.1: Overview of our approach. First, we train a regression forest *off-line* to predict 2D-to-3D correspondences for a generic scene. To adapt this forest to a new scene, we remove the scene-specific information in the forest's leaves while retaining the branching structure (with learned split parameters) of the trees; we then refill the leaves *online* using training examples from the new scene. The adapted forest can be deployed to predict correspondences for the new scene that are fed to Kabsch [63] and RANSAC [45] for pose estimation.

6.3.2 *Details*

6.3.2.1 *Offline Forest Training*

Training is done as in [116], greedily optimising a standard reduction-in-spatial-variance objective over the randomised parameters of simple threshold functions. Like [116], we make use of 'Depth' and 'Depth-Adaptive RGB' ('DA-RGB') features, centred at a pixel $\mathbf{p}$, as follows:

$$f_{\Omega}^{\text{Depth}} = d(\mathbf{p}) - d\left(\mathbf{p} + \frac{\delta}{d(\mathbf{p})}\right) \tag{6.1}$$

$$f_{\Omega}^{\text{DA-RGB}} = i(\mathbf{p}, c) - i\left(\mathbf{p} + \frac{\delta}{d(\mathbf{p})}, c\right) \tag{6.2}$$

In this, $d(\mathbf{p})$ is the depth at $\mathbf{p}$, $i(\mathbf{p}, c)$ is the value of the c^{th} colour channel at $\mathbf{p}$, and Ω is a vector of randomly sampled feature parameters. For 'Depth', the only parameter is the 2D image-space offset $\boldsymbol{\delta}$, whereas 'DA-RGB' adds the colour channel selection parameter $c \in \{R, G, B\}$. We randomly generate 128 values of Ω for 'Depth' and 128 for 'DA-RGB' and concatenate the evaluations of these functions at each pixel of interest to yield 256D feature vectors.

At training time, a set S of training examples, each consisting of such a feature vector $\mathbf{f} \in \mathbb{R}^{256}$, its corresponding 3D location in the scene and its colour, is assembled via sampling from a ground truth RGB-D video with known camera poses for each frame (obtained by tracking from depth camera input). A random subset of these training examples is selected to train each tree in the forest, and we then train all of the trees in parallel.

Starting from the root of each tree, we recursively partition the training examples in the current node into two using a binary threshold function. To decide how to split each node n, we randomly generate a set Θ_n of 512 candidate split parameter pairs, where each $\theta = (\phi, \tau) \in \Theta_n$ denotes the binary threshold function

$$\theta(\mathbf{f}) = \mathbf{f}[\phi] \geqslant \tau. \tag{6.3}$$

In this, $\phi \in [0, 256)$ is a randomly-chosen feature index, and $\tau \in \mathbb{R}$ is a threshold, chosen to be the value of feature ϕ in a randomly-chosen training example. Examples that pass the test are routed to the right subtree of n; the remainder are routed to the left. To pick a suitable split function for n, we use exhaustive search to find a $\theta^* \in \Theta_n$ whose corresponding split function maximises the information gain that can be achieved by splitting the training examples that reach n. Formally, the information gain corresponding to split parameters $\theta \in \Theta_n$ is

$$V(S_n) - \sum_{i \in \{L,R\}} \frac{|S_n^i(\theta)|}{|S_n|} V(S_n^i(\theta)), \tag{6.4}$$

in which $V(X)$ denotes the spatial variance of set X, and $S_n^L(\theta)$ and $S_n^R(\theta)$ denote the left and right subsets into which the set $S_n \subseteq S$ of training examples reaching n is partitioned by the split function denoted by θ. Spatial variance is defined in terms of the log of the determinant of the covariance of a fitted 3D Gaussian [116].

For a given tree, the above process is simply recursed to a maximum depth of 15. We train 5 trees per forest. The (approximate, empirical) distributions in the leaves are discarded at the end of this process (we replace them during online forest adaptation, as discussed in the next section).

6.3.2.2 *Online Forest Adaptation*

To adapt a forest to a new environment, we replace the distributions discarded from its leaves at the end of pre-training with dynamically-updated ones drawn entirely from the new scene. Here, we detail how the new leaf distributions used by the relocaliser are computed and updated online.

We draw inspiration from the use of reservoir sampling [118] in SemanticPaint [117], which makes it possible to store an unbiased subset of an empirical distribution in a bounded amount of memory. On initialisation, we allocate (on the GPU) a fixed-size sample reservoir for each leaf of the existing forest. Our reservoirs contain up to 1024 entries, each of which storing a 3D location (in world coordinates) and an associated colour. At runtime, we pass training examples (of the form described in Subsection 6.3.2.1) down the forest and identify the leaves to which each example is mapped. We then add the 3D location and colour of each example to the reservoirs associated with its leaves.

To obtain the 3D locations of the training examples, we need to know the transformation that maps points from camera space to world space. When testing on sequences from a dataset, this is trivially available as the ground truth camera pose, but in a live scenario, it will generally be obtained as the output of a fallible tracker. To avoid corrupting the reservoirs in our forest, we avoid passing new examples down the forest when the tracking is unreliable. We measure tracker reliability using the support vector machine (SVM) approach described in [64]. For frames for which a reliable camera pose *is* available, we proceed as follows:

1. First, we compute feature vectors for a subset of the pixels in the image, as detailed in Subsection 6.3.2.1. We empirically

choose our subset by subsampling densely on a regular grid with 4-pixel spacing, *i.e.* we choose pixels $\{(4i, 4j) \in [0, w) \times [0, h) : i, j \in \mathbb{N}^+\}$, where w and h are respectively the width and height of the image.

2. Next, we pass each feature vector down the forest, adding the 3D position and colour of the corresponding scene point to the reservoir of the leaf reached in each tree. Our CUDA-based random forest implementation uses the node indexing described in [103].

3. Finally, for each leaf reservoir, we cluster the contained points using a CUDA implementation of Really Quick Shift (RQS) [47] to find a set of modal 3D locations. We sort the clusters in each leaf in decreasing size order, keeping at most 10 modal clusters per leaf. For each cluster we keep, we compute 3D and colour centroids, and a covariance matrix. The cluster distributions are used when estimating the likelihood of a camera pose, and also during continuous pose optimisation (see Subsection 6.3.2.3). Since running RQS over all the leaves in the forest would take too long if run in a single frame, we amortise the cost over multiple frames by updating 256 leaves in parallel for each frame, in round-robin fashion. A typical forest contains around 42,000 leaves, so each leaf is updated roughly once every 6 s.

Figure 6.2 provides an illustrative example of the effect that online adaptation has on a pre-trained forest: Figure 6.2a shows the modal clusters present in a small number of randomly-selected leaves of a forest pre-trained on the *Chess* scene from the 7-Scenes dataset [104]; Figure 6.2b shows the modal clusters that are added to the same leaves during the process of adapting the forest to the *Kitchen* scene. Note that whilst the positions of the predicted modes have (unsurprisingly) completely changed, the split functions in the forest's branch nodes (which we preserve) still do a good job of routing similar parts of the scene into the same leaves, enabling effective sampling of 2D-to-3D correspondences for camera pose estimation.

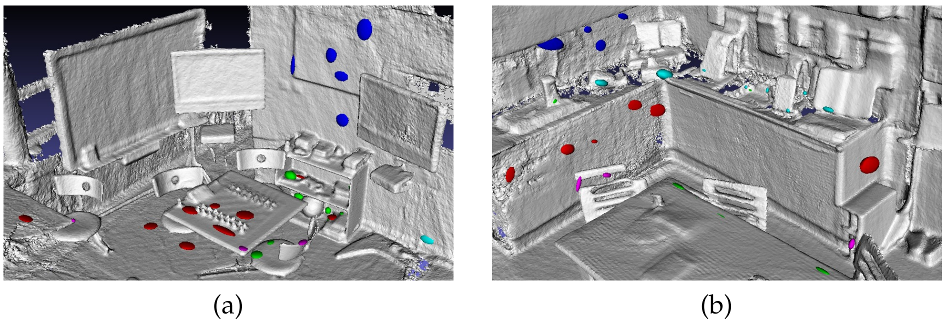


Figure 6.2: An illustrative example of the effect that online adaptation has on a pre-trained forest: (a) shows the modal clusters present in a small number of randomly-selected leaves of a forest pre-trained on the *Chess* scene from the 7-Scenes dataset [104] (the colour of each mode indicates its containing leaf); (b) shows the modal clusters that are added to the same leaves during the process of adapting the forest to the *Kitchen* scene.

6.3.2.3 *Camera Pose Estimation*

As in [116], camera pose estimation is based on the preemptive, locally-optimised RANSAC of [24]. We begin by randomly generating an initial set of up to 1024 pose hypotheses. A pose hypothesis $H \in \mathbf{SE}(3)$ is a transform that maps points in camera space to world space. To generate each pose hypothesis, we apply the Kabsch algorithm [63] to 3 point pairs of the form $(\mathbf{x}_i^{\mathcal{C}}, \mathbf{x}_i^{\mathcal{W}})$, where $\mathbf{x}_i^{\mathcal{C}} = d(\mathbf{u}_i)K^{-1}(\mathbf{u}_i^{\top}, 1)$ is obtained by back-projecting a randomly-chosen point $\mathbf{u}_i$ in the live depth image d into camera space, and $\mathbf{x}_i^{\mathcal{W}}$ is a corresponding scene point in world space, randomly sampled from $M(\mathbf{u}_i)$, the modes of the leaves to which the forest maps $\mathbf{u}_i$. In this, K is the intrinsic calibration matrix for the depth camera. Before accepting a hypothesis, we subject it to a series of checks:

1. First, we randomly choose one of the three point pairs $(\mathbf{x}_i^{\mathcal{C}}, \mathbf{x}_i^{\mathcal{W}})$ and compare the RGB colour of the corresponding pixel $\mathbf{u}_i$ in the colour input image to the colour centroid of the mode (see Subsection 6.3.2.2) from which we sampled $\mathbf{x}_i^{\mathcal{W}}$. We reject the hypothesis *iff* the L0 distance between the two exceeds a threshold.

2. Next, we check that the three hypothesised scene points are sufficiently far from each other. We reject the hypothesis *iff*

the minimum distance between any pair of points is less than 30 cm.

3. Finally, we check that the distances between all scene point pairs and their corresponding back-projected depth point pairs are sufficiently similar, *i.e.* that the hypothesised transform is "rigid enough". We reject the hypothesis *iff* this is not the case.

If a hypothesis gets rejected by one of the checks, we try to generate an alternative hypothesis to replace it. In practice, we use 1024 dedicated threads, each of which attempts to generate a single hypothesis. Each thread continues generating hypotheses until either (a) it finds a hypothesis that passes all of the checks, or (b) a maximum number of iterations is reached. We proceed with however many hypotheses we obtain by the end of this process.

Having generated our large initial set of hypotheses, we next aggressively cut it down to a much smaller size by scoring each hypothesis and keeping the 64 lowest-energy transforms (if there are fewer than 64 hypotheses, we keep all of them). To score the hypotheses, we first select an initial set $I = \{i\}$ of 500 pixel indices in d, and back-project the denoted pixels $\mathbf{u}_i$ to corresponding points $\mathbf{x}_i^{\mathcal{C}}$ in camera space as described above. We then score each hypothesis H by summing the Mahalanobis distances between the transformations of each $\mathbf{x}_i^{\mathcal{C}}$ under H and their nearest modes:

$$E(H) = \sum_{i \in I} \left(\min_{(\mu,\Sigma) \in M(\mathbf{u}_i)} \left\| \Sigma^{-\frac{1}{2}} (H\mathbf{x}_i^{\mathcal{C}} - \mu) \right\| \right) \tag{6.5}$$

After this initial cull, we use pre-emptive RANSAC to prune the remaining $\leqslant 64$ hypotheses to a single, final hypothesis. We iteratively (i) expand the sample set I (by adding 500 new pixels each time), (ii) refine the pose candidates via Levenberg-Marquardt optimisation [75, 81] of the energy function E, (iii) re-evaluate and re-score the hypotheses, and (iv) discard the worse half. In practice, the actual optimisation is performed not in $\mathbf{SE}(3)$, where it would be hard to do, but in the corresponding Lie algebra, $\mathfrak{se}(3)$. The details of this process can be found in [116], and a longer explanation of Lie algebras can be found in [11, 110].

This process yields a single pose hypothesis, which we can then return if desired. In practice, however, further pose refinement is sometimes possible. For example, were the relocaliser to be integrated into a 3D reconstruction framework such as KinectFusion [91], VoxelHashing [94] or InfiniTAM [64][1], we can attempt to refine the pose further using ICP [9]. Since tasks such as 3D reconstruction are one of the key applications of our approach, we report results both with and without ICP in Table 6.1.

6.4 EXPERIMENTS

We perform both quantitative and qualitative experiments to evaluate our approach. In Subsection 6.4.1, we compare our *adaptive* approach to state-of-the-art *offline* relocalisers that have been trained directly on the scene of interest. We show that our adapted forests can achieve competitive relocalisation performance despite being trained on very different scenes, allowing them to be used for *online* relocalisation. In Subsection 6.4.2, we show our ability to perform this adaptation on-the-fly from live sequences, allowing us to support tracking loss recovery in interactive scenarios. In Subsection 6.4.3, we evaluate how well our approach generalises to novel poses in comparison to a keyframe-based random fern relocaliser based on [51]. This relocaliser is also practical for on-the-fly relocalisation (hence its use in large scale reconstruction pipelines such as InfiniTAM [64]), but its use of keyframes means that it is unable to generalise well to novel poses. By contrast, we are able to relocalise well even from poses that are quite far away from the training trajectory. In Subsection 6.4.4, we compare the speed of our approach with random ferns during both normal operation (*i.e.* when the scene is being successfully tracked) and relocalisation. Our approach is slower than random ferns, but remains close to real-time, whilst achieving significantly higher relocalisation performance. In Subsection 6.4.5 we show some examples of situations where the

1 The research described in this chapter was performed during a period spent as a visitor in the "Torr Vision Group", University of Oxford, United Kingdom. For this reason the reconstruction pipeline used as base of the relocalisation system being described is "InfiniTAM" (currently maintained by members of the Department of Engineering Science in Oxford) instead of the SemanticFusion pipeline, based on VoxelHashing, described in the previous chapter. This has no implications on the relocaliser, since its implementation does not depend on the specific SLAM pipeline employed. Its deployment in a system such as "SemanticFusion", described earlier, would be straightforward.

proposed relocalisation method fails and draw conclusions so as to the likely causes of the errors. Finally, in Subsection 6.4.6 we show additional qualitative examples of challenging but successful camera relocalisations.

6.4.1 *Adaptation Performance*

In evaluating the extent to which we are able to adapt a regression forest that has been pre-trained on a different scene to the scene of interest, we seek to answer two questions. First, how does an adapted forest compare to one that has been pre-trained offline on the target scene? Second, to what extent does an adapted forest's performance depend on the scene on which it has been pre-trained? To answer both of these questions, we compare the performances of adapted forests pre-trained on a variety of scenes (each scene from the 7-Scenes dataset [104], plus a novel scene depicting an office desk) to the performances of forests trained offline on the scene of interest using state-of-the-art approaches [12, 55, 104, 116].

The exact testing procedure we use for our approach is as follows. First, we pre-train a forest on a generic scene and remove the contents of its leaves, as described in Section 6.3: this process runs *offline* over a number of hours or even days (but we only need to do it once). Next, we adapt the forest by feeding it new examples from a training sequence captured on the scene of interest: this runs *online* at frame rates (in a real system, this allows us to start relocalising almost immediately whilst training carries on in the background, as we show in Subsection 6.4.2). Finally, we test the adapted forest by using it to relocalise from every frame of a separate testing sequence captured on the scene of interest.

As shown in Table 6.1, the results are very accurate. Whilst there are certainly some variations in the performance achieved by adapted forests pre-trained on different scenes (in particular, forests trained on the *Heads* and *Pumpkin* scenes from the dataset are slightly worse), the differences are not profound: in particular, relocalisation performance seems to be more tightly coupled to the difficulty of the scene of interest than to the scene on which the forest was pre-trained. Notably, all of our adapted forests achieve

Training Scene		Relocalisation Performance on Test Scene							
		Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	Average (all scenes)
Chess	Reloc	99.8%	95.7%	95.5%	91.7%	82.8%	77.9%	25.8%	81.3%
	+ ICP	99.9%	97.8%	99.5%	94.1%	91.3%	83.3%	28.4%	84.9%
Fire	Reloc	98.4%	96.9%	98.2%	89.7%	80.5%	71.9%	28.6%	80.6%
	+ ICP	99.1%	99.2%	99.9%	92.1%	89.1%	81.7%	31.0%	84.6%
Heads	Reloc	98.0%	91.7%	100%	73.1%	77.5%	67.1%	21.8%	75.6%
	+ ICP	99.3%	92.3%	100%	81.1%	87.7%	82.0%	31.9%	82.0%
Office	Reloc	99.2%	96.5%	99.7%	97.6%	84.0%	81.7%	33.6%	84.6%
	+ ICP	99.4%	99.0%	100%	98.2%	91.2%	87.0%	35.0%	87.1%
Pumpkin	Reloc	97.5%	94.9%	96.9%	82.7%	83.5%	70.4%	30.7%	75.5%
	+ ICP	98.9%	97.6%	99.4%	86.9%	91.2%	82.3%	32.4%	84.1%
Kitchen	Reloc	99.9%	95.4%	98.0%	93.3%	83.2%	86.0%	28.2%	83.4%
	+ ICP	99.9%	98.2%	100%	94.5%	90.4%	88.1%	31.3%	86.1%
Stairs	Reloc	97.3%	95.4%	97.9%	90.8%	80.6%	74.5%	45.7%	83.2%
	+ ICP	98.0%	97.4%	99.8%	92.1%	89.5%	81.0%	46.6%	86.3%
Ours (Desk)	Reloc	97.3%	95.7%	97.3%	83.7%	85.3%	71.8%	24.3%	79.3%
	+ ICP	99.2%	97.7%	100%	88.2%	90.6%	82.6%	31.0%	84.2%
Average	Reloc	98.4%	95.3%	97.9%	87.8%	82.2%	75.2%	29.8%	80.9%
	+ ICP	99.2%	97.4%	99.8%	90.9%	90.1%	83.5%	33.5%	84.9%

Table 6.1: The performance of our *adaptive* approach after pre-training on various scenes of the 7-Scenes dataset [104]. We show the scene used to pre-train the forest in each version of our approach in the left column. The pre-trained forests are adapted *online* for the test scene, as described in the main text. Percentages denote proportions of test frames with $\leqslant 5\text{cm}$ translational error and $\leqslant 5°$ angular error.

results that are within striking distance of the state-of-the-art *offline* methods (Table 6.2), and are considerably better than those that can be achieved by online competitors such as the keyframe-based random fern relocaliser implemented in InfiniTAM [51, 64] (see Subsection 6.4.3). Nevertheless, there is clearly a trade-off to be made here between performance and practicality: pre-training on the scene of interest is impractical for on-the-fly relocalisation, but achieves somewhat better results, probably due to the opportunity afforded to adapt the structure of the forest to the target scene.

This drop in performance in exchange for practicality can be mitigated to some extent by refining our relocaliser's pose estimates using the ICP-based tracker [9] in InfiniTAM [66]. Valentin et al. [116] observe that the 5 cm/5° error metric commonly used to evaluate relocalisers is "fairly strict and should allow any robust model-based tracker to resume". In practice, ICP-based tracking is in many cases able to resume from initial poses with even greater error: indeed, as Table 6.1 shows, with ICP refinement enabled, we are able to relocalise from a significantly higher proportion of test frames. Whilst ICP could clearly also be used to refine the results of offline methods, what is important in this case is that ICP is fast and does not add significantly to the overall runtime of our approach, which remains close to real time. As such, refining *our* pose estimates using ICP yields a high-quality relocaliser that is still practical for online use.

6.4.2 *Tracking Loss Recovery*

In Subsection 6.4.1, we investigated our ability to adapt a forest to a new scene by filling its leaves with data from a training sequence for that scene, before testing the adapted forest on a separate testing sequence shot on the same scene. Here, we quantify our approach's ability to perform this adaptation *on the fly*, by filling the leaves frame-by-frame from the testing sequence: this allows recovery from tracking loss in an interactive scenario without the need for prior training on anything other than the live sequence, making our approach extremely convenient for tasks such as interactive 3D reconstruction.

Scene	[104]	[55]	[116]	[12]	**Us**	**Us+ICP**
Chess	92.6%	96%	99.4%	99.6%	99.2%	99.4%
Fire	82.9%	90%	94.6%	94.0%	96.5%	99.0%
Heads	49.4%	56%	95.9%	89.3%	99.7%	100%
Office	74.9%	92%	97.0%	93.4%	97.6%	98.2%
Pumpkin	73.7%	80%	85.1%	77.6%	84.0%	91.2%
Kitchen	71.8%	86%	89.3%	91.1%	81.7%	87.0%
Stairs	27.8%	55%	63.4%	71.7%	33.6%	35.0%
Average	67.6%	79.3%	89.5%	88.1%	84.6%	87.1%

Table 6.2: Comparing our *adaptive* approach to state-of-the-art *offline* methods on the 7-Scenes dataset [104] (the percentages denote proportions of test frames with $\leqslant$ 5cm translation error and $\leqslant 5°$ angular error). For our method, we report the results obtained by adapting a forest pre-trained on the *Office* sequence (from Table 6.1). We are competitive with, and sometimes better than, the offline methods, without needing to pre-train on the test scene.

Our testing procedure is as follows: at each new frame (except the first), we assume that tracking has failed, and try to relocalise using the forest we have available at that point; we record whether or not this succeeds. Regardless, we then restore the ground truth camera pose (or the tracked camera pose, in a live sequence) and, provided tracking hasn't actually failed, use examples from the current frame to continue training the forest. As Figure 6.3 shows, we are able to start relocalising almost immediately in a live sequence (in a matter of frames, typically 4–6 are enough). Subsequent performance then varies based on the difficulty of the sequence, but rarely drops below 80%, except for the challenging *Stairs* sequence. This makes our approach highly practical for interactive relocalisation, something we also show in an accompanying video[2].

6.4.3 *Generalisation to Novel Poses*

To evaluate how well our approach generalises to novel poses, we examine how the proportion of frames we can relocalise decreases as

2 https://vision.disi.unibo.it/~tcavallari/phd_thesis/relocalisation.mp4

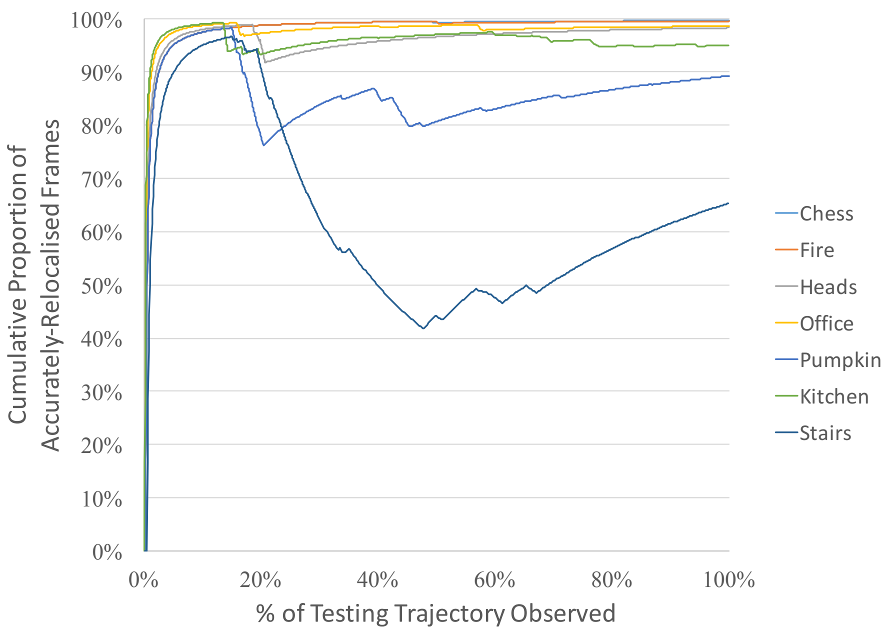


Figure 6.3: The performance of our approach for tracking loss recovery (Subsection 6.4.2). Filling the leaves of a forest pre-trained on *Office* frame-by-frame *directly* from the *testing* sequence, we are able to start relocalising almost immediately in new scenes. This makes our approach highly practical in interactive scenarios such as 3D reconstruction.

the distance of the (ground truth) test poses from the training trajectory increases. We compare our approach with the keyframe-based relocaliser in InfiniTAM [64], which is based on the random fern approach of Glocker et al. [51]. Relocalisation from novel poses is a well-known failure case of keyframe-based methods, so we would expect the random fern approach to perform poorly away from the training trajectory; by contrast, it is interesting to see the extent to which our approach can relocalise from a wide range of novel poses.

We perform the comparison separately for each 7-Scenes sequence, and then aggregate the results. For each sequence, we first group the test poses into bins by pose novelty. Each bin is specified in terms of a maximum translation and rotation difference of a test pose with respect to the training trajectory (for example, poses that are within 5 cm and 5° of any training pose are assigned to the first bin, remaining poses that are within 10 cm and 10° are assigned to the second bin, etc. . .). We then determine the proportion of the test poses in each bin for which it is possible to relocalise to within 5 cm

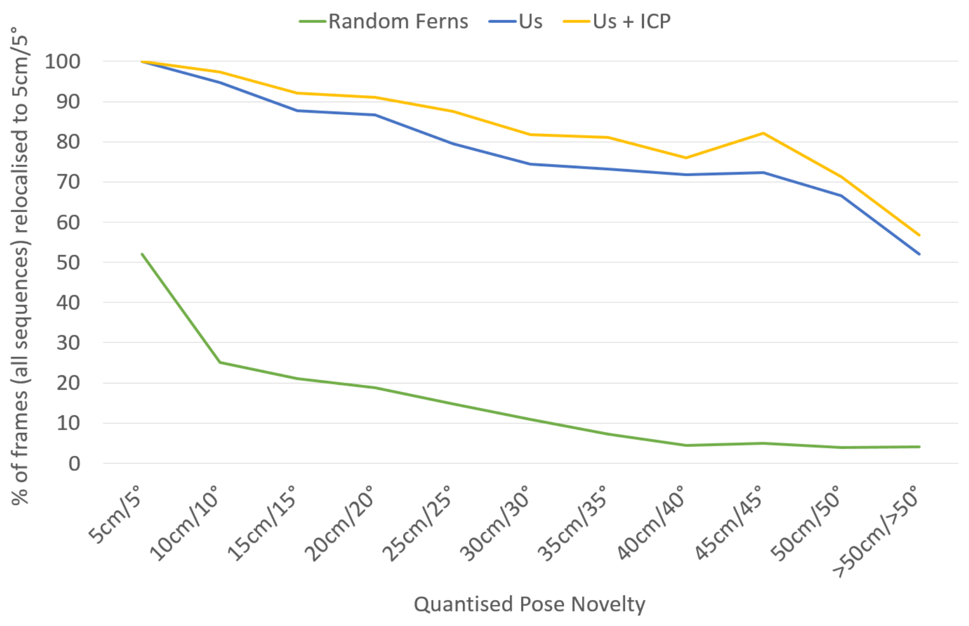


Figure 6.4: Evaluating how well our approach generalises to novel poses in comparison to a keyframe-based random fern relocaliser based on [51]. The performance decay experienced as test poses get farther from the training trajectory is much less severe with our approach than with random ferns.

translational error and 5° angular error using (a) the random fern approach, (b) our approach without ICP and (c) our approach with ICP. As shown in Figure 6.4, the decay in performance experienced as the test poses get further from the training trajectory is much less severe with our approach than with random ferns.

A qualitative example of our ability to relocalise from novel poses is shown in Figure 6.5. In the main figure, we show a range of test poses from which we can relocalise in the *Fire* scene, linking them to nearby poses on the training trajectory so as to illustrate their novelty in comparison to poses on which we have trained. The most difficult of these test poses are also shown in the images below, alongside their nearby training poses, visually illustrating the significant differences between the two.

As Figure 6.4 and Figure 6.5 illustrate, we are already quite effective at relocalising from poses that are significantly different from those on which we have trained; nevertheless, further improvements seem possible. For example, one interesting extension of this work might be to explore the possibility of using rotation-invariant split

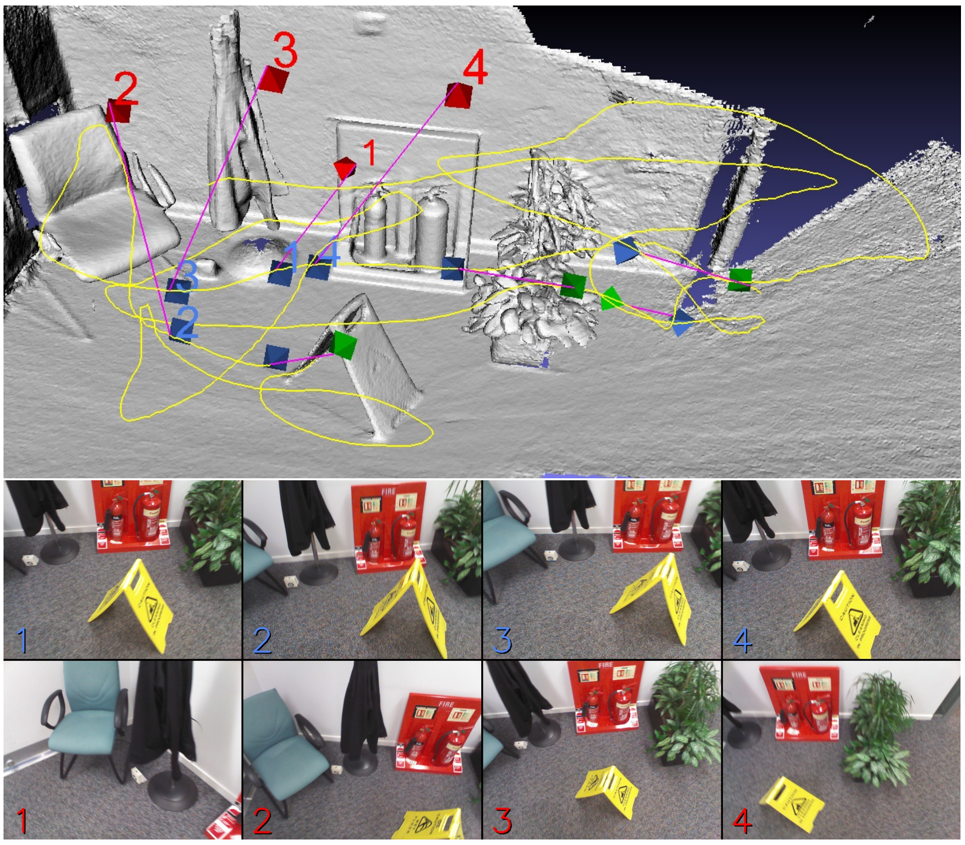


Figure 6.5: A qualitative example of novel poses from which we are able to relocalise to within 5 cm/5° on the *Fire* sequence from 7-Scenes dataset [104]. Pose novelty measures the distance of a test pose from a nearby pose (blue) on the training trajectory (yellow). We can relocalise from both easy poses (up to 35 cm/35° from the training trajectory, green) and hard poses (>35 cm/35°, red). The images below the main figure show views of the scene from the training poses and testing poses indicated.

functions in the regression forest to improve its generalisation capabilities.

6.4.4 *Timings*

To evaluate the usefulness of our approach for on-the-fly relocalisation in new scenes, we compare it to the keyframe-based random fern relocaliser implemented in InfiniTAM [51, 64]. To be practical in a real-time system, a relocaliser needs to perform in real time during normal operation (*i.e.* for online training whilst successfully tracking the scene), and ideally take no more than around 200 ms

	Random Ferns [51, 64]	**Us**
Per-Frame Training	0.9 ms	9.8 ms
Relocalisation	10 ms	141 ms

Table 6.3: Comparing the typical timings of our approach vs. random ferns during both normal operation and relocalisation. Our approach is slower than random ferns, but achieves significantly higher relocalisation performance, especially from novel poses. All of our experiments are run on a machine with an Intel Core i7-4960X CPU and an NVIDIA GeForce Titan Black GPU.

for relocalisation itself (when the system has lost track). As a result, relocalisers such as [12, 55, 82, 104, 116], whilst achieving impressive results, are not practical in this context due to their need for offline training on the scene of interest.

As shown in Table 6.3, the random fern relocaliser is fast both for online training and relocalisation, taking only 0.9 ms per frame to update the keyframe database, and 10 ms to relocalise when tracking is lost. However, speed aside, the range of poses from which it is able to relocalise is quite limited. By contrast, our approach, whilst taking 9.8 ms for online training and 141 ms for actual relocalisation, can relocalise from a much broader range of poses, still running at acceptable speeds. Additionally, it should be noted that our current research-focused implementation is not heavily optimised, making it plausible that it could be sped up even further with additional engineering effort.

6.4.5 *Analysis of Failure Cases*

As shown in the previous sections, our approach is able to achieve highly-accurate online relocalisation in under 150 ms, from novel camera poses and without needing extensive offline training on the target scene. However, there are inevitably still situations in which it will fail. In this section, we analyse two interesting failure cases, so as to help the reader understand the underlying reasons in each case.

6.4.5.1 *Office*

The first failure case we analyse is from the *Office* scene in the 7-Scenes dataset [104]. This scene captures a typical office that contains a number of desks (see Figure 6.6). Unfortunately, these desks appear visually quite similar: they are made of the same wood, and have similar monitors and the same associated chairs. This makes it very difficult for a relocaliser such as ours to distinguish between them: as a result, our approach ends up producing a pose that faces the wrong desk (see Figure 6.6d).

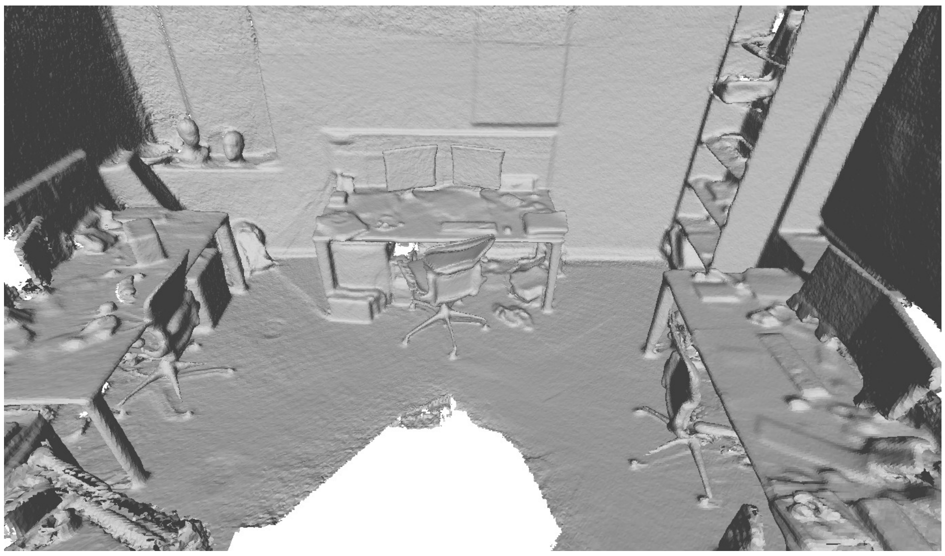
On one level, the pose we produce is not entirely unreasonable: indeed, it looks superficially plausible, and is oriented at roughly the right angle with respect to the incorrect desk. Nevertheless, in absolute terms, the pose is obviously very far from the ground truth.

To pin down what has gone wrong, we visualise the last 16 surviving camera pose hypotheses for this instance in Figure 6.7, in descending order (left-to-right, top-to-bottom). We observe that whilst the top candidate selected by RANSAC relocalises the camera to face the wrong desk, any of the next five candidates would have relocalised the camera successfully. The problem in this case is that the energies computed for the hypotheses are fairly similar for both the correct and incorrect poses.

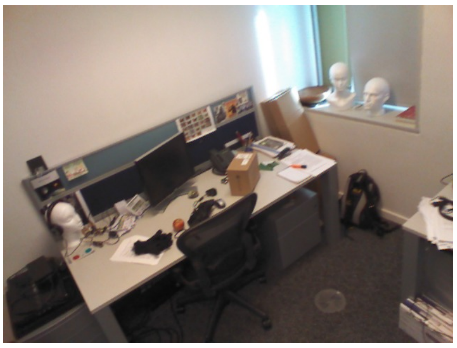
Although we do not investigate it here, one potential way of fixing this might be to score the last few surviving hypotheses based on the photometric consistencies between colour raycasts from their respective poses and the colour input image.

6.4.5.2 *Stairs*

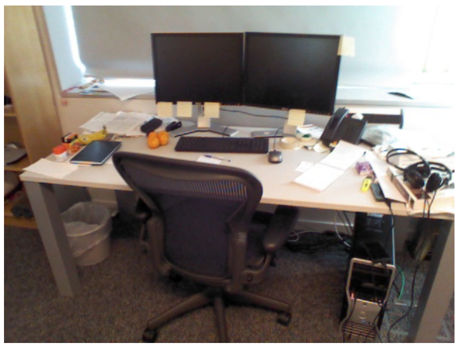
The second failure case we analyse is from the *Stairs* scene in the 7-Scenes dataset [104]. This is a notoriously difficult scene containing a staircase that consists of numerous visually-identical steps (see Figure 6.8). When viewing the scene from certain angles (see Figure 6.9), the relocaliser is able to rely on points in the scene that can be identified unambiguously to correctly estimate the pose, but from viewpoints such as that in Figure 6.8d, it is forced to use more ambiguous points,*e.g.* those on the stairs themselves or the walls. When this happens, relocalisation is prone to fail, since the relocaliser finds it difficult to tell the difference between the different steps.



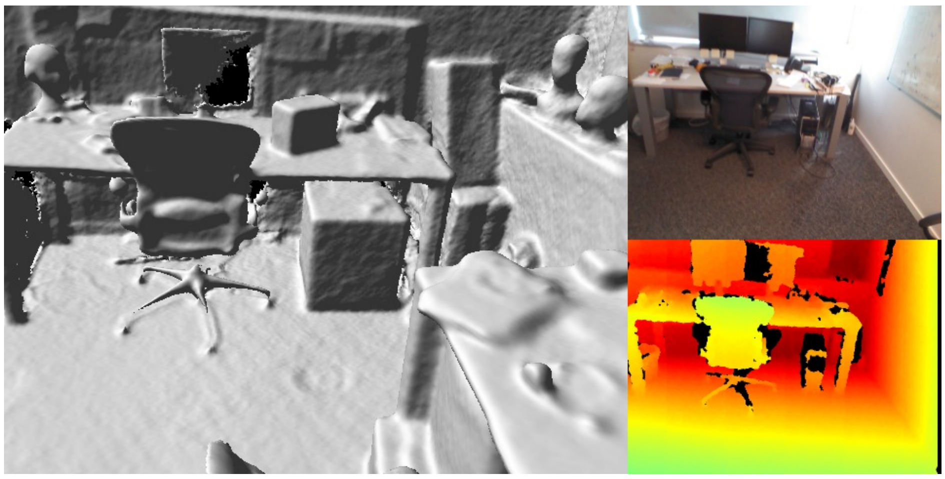
(a)



(b)



(c)



(d)

Figure 6.6: The *Office* scene from the 7-Scenes dataset [104]. (a) contains multiple desks, *e.g.* (b) and (c), that can appear visually quite similar, making it difficult for the relocaliser to distinguish between them. In (d), for example, the relocaliser incorrectly chooses a pose facing the desk in (b), whilst the RGB-D input actually shows the desk in (c).

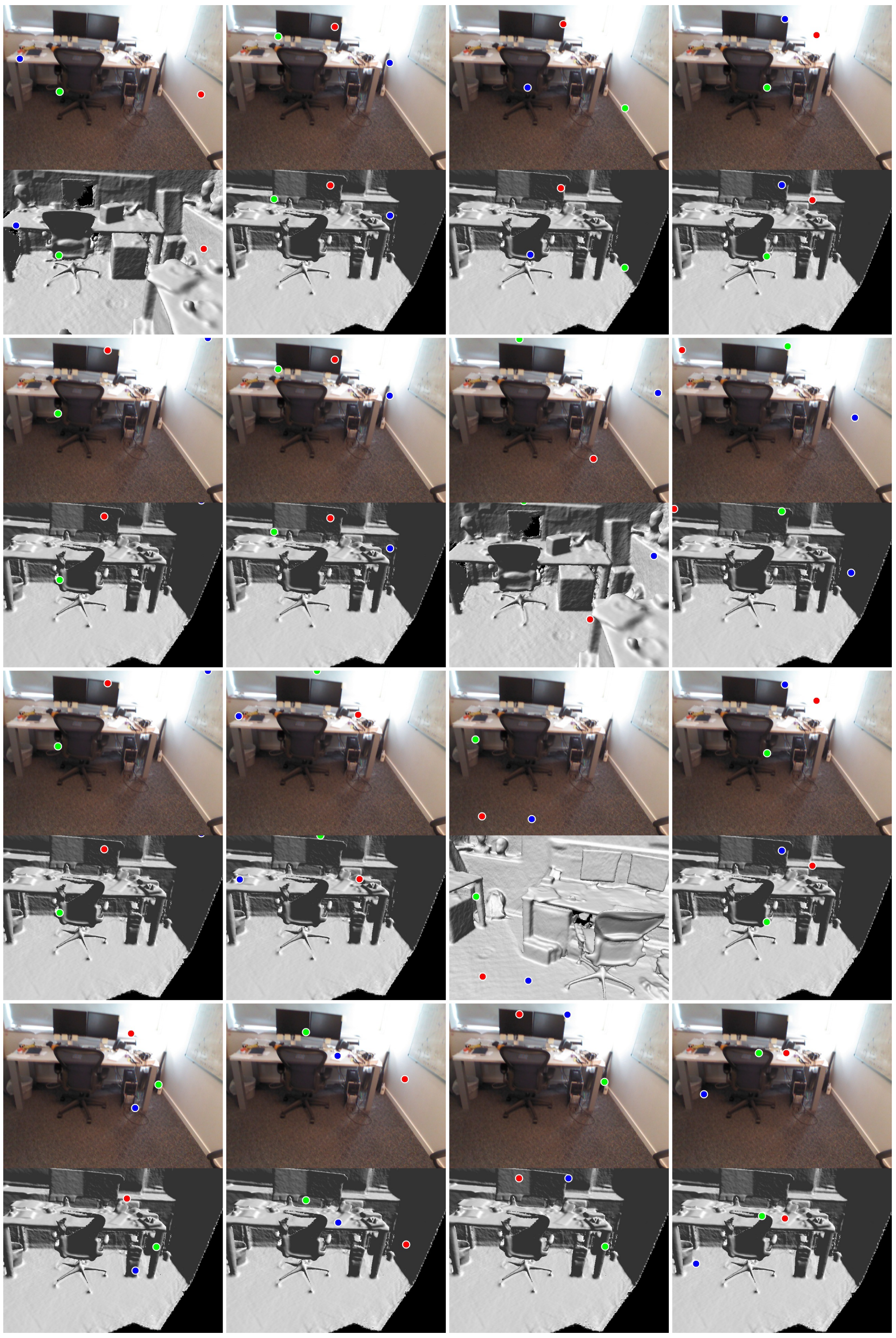
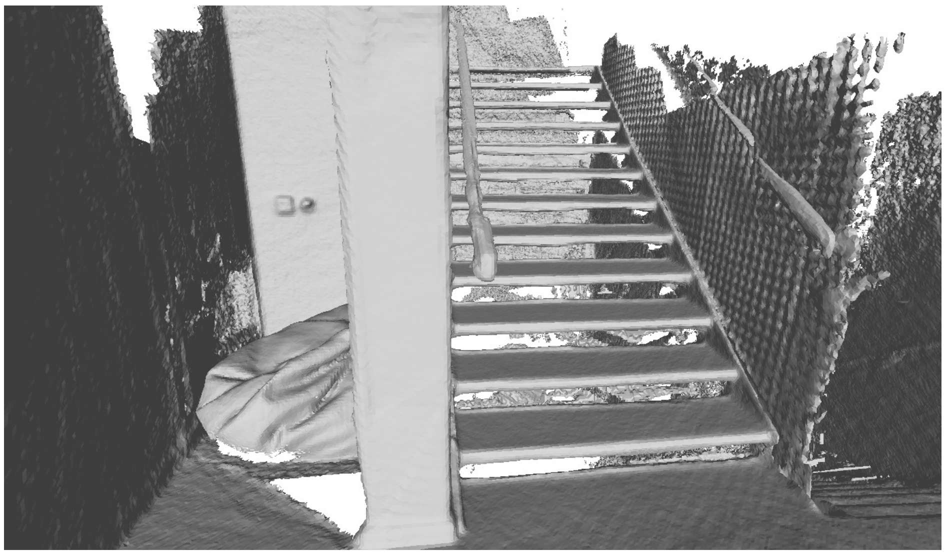
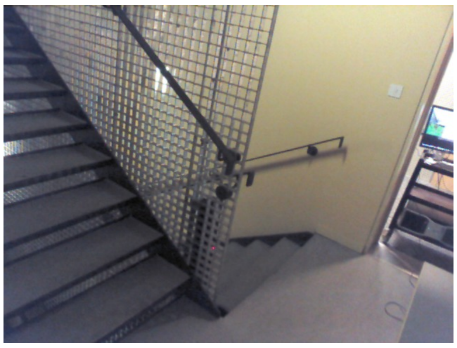
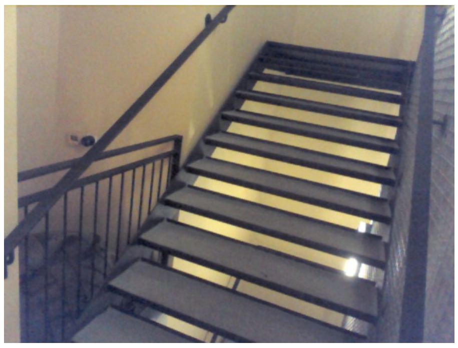


Figure 6.7: The top 16 pose candidates (left-to-right, top-to-bottom) corresponding to the failure case on the *Office* scene shown in Figure 6.6d. The coloured points indicate the 2D-to-3D correspondences that are used to generate the initial pose hypotheses. Note that whilst the top candidate selected by RANSAC relocalises the camera to face the wrong desk, any of the next five candidates would have relocalised the camera correctly.


(a)


(b)


(c)

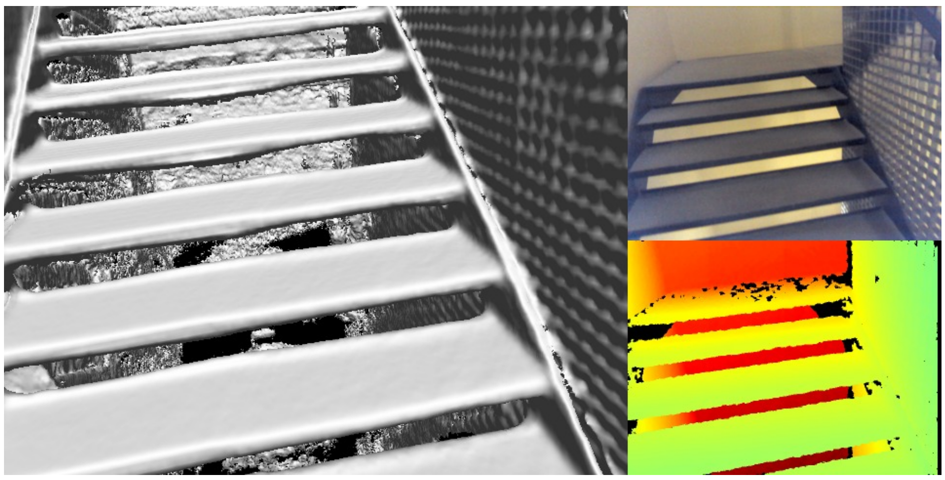

(d)

Figure 6.8: The *Stairs* scene from the 7-Scenes dataset [104]. (a) is notoriously difficult, containing a staircase that consists of numerous visually-identical steps (see (b) and (c)). In (d), many of the 2D-to-3D correspondences predicted by the forest are likely to be of a low quality, since it is hard to distinguish between similar points on different steps. This significantly reduces the probability of generating good initial hypotheses, leaving RANSAC trying to pick a good hypothesis from an initial set that only contains bad ones.

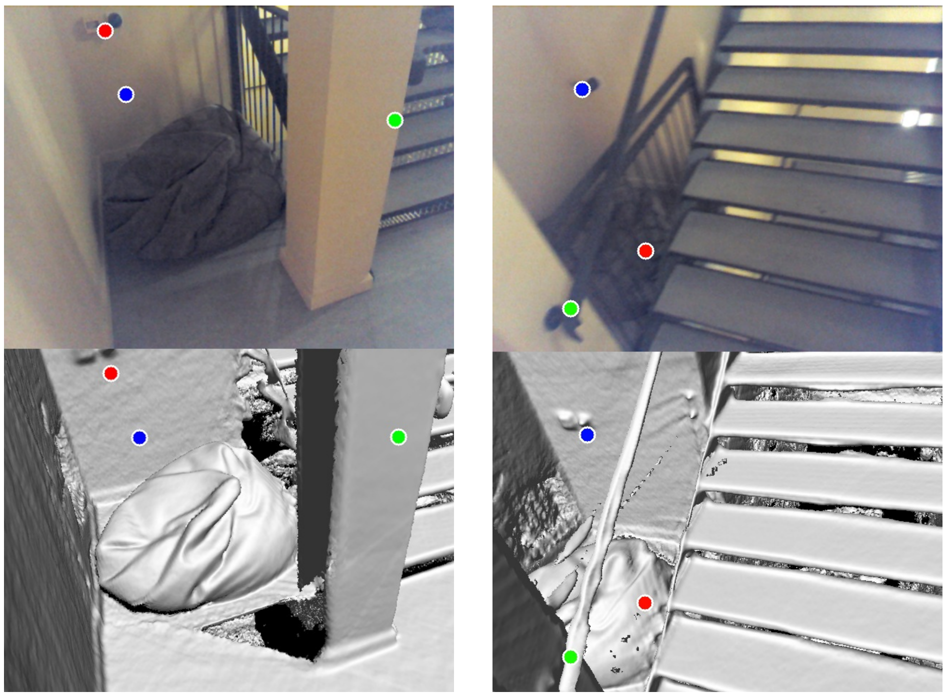


Figure 6.9: From certain angles in the *Stairs* scene, the relocaliser is able to rely on points in the scene that can be identified unambiguously to estimate the pose.

As in the previous section, we can visualise the top 16 camera pose hypotheses for this instance to pin down what has gone wrong (see Figure 6.10). It is noticeable that in this case, none of the top 16 hypotheses would have successfully relocalised the camera. As suggested by the points predicted in the 3D scene for each hypothesis (which are often in roughly the right place but on the wrong step), this is because the points at the same places on different steps tend to end up in similar leaves, making the modes in the leaves less informative (see also Figure 6.11) and significantly reducing the probability of generating good initial hypotheses.

Unlike in the *Office* case, the problem here cannot be fixed by a late-stage consistency check, since none of the last few surviving hypotheses are of any use. Instead, one potential way of fixing this might be to improve the way in which the initial set of hypotheses is generated so as to construct a more diverse set and increase the probability of one of the initial poses being in roughly the right place. An alternative might be to adaptively increase the number of hypotheses generated in difficult conditions.

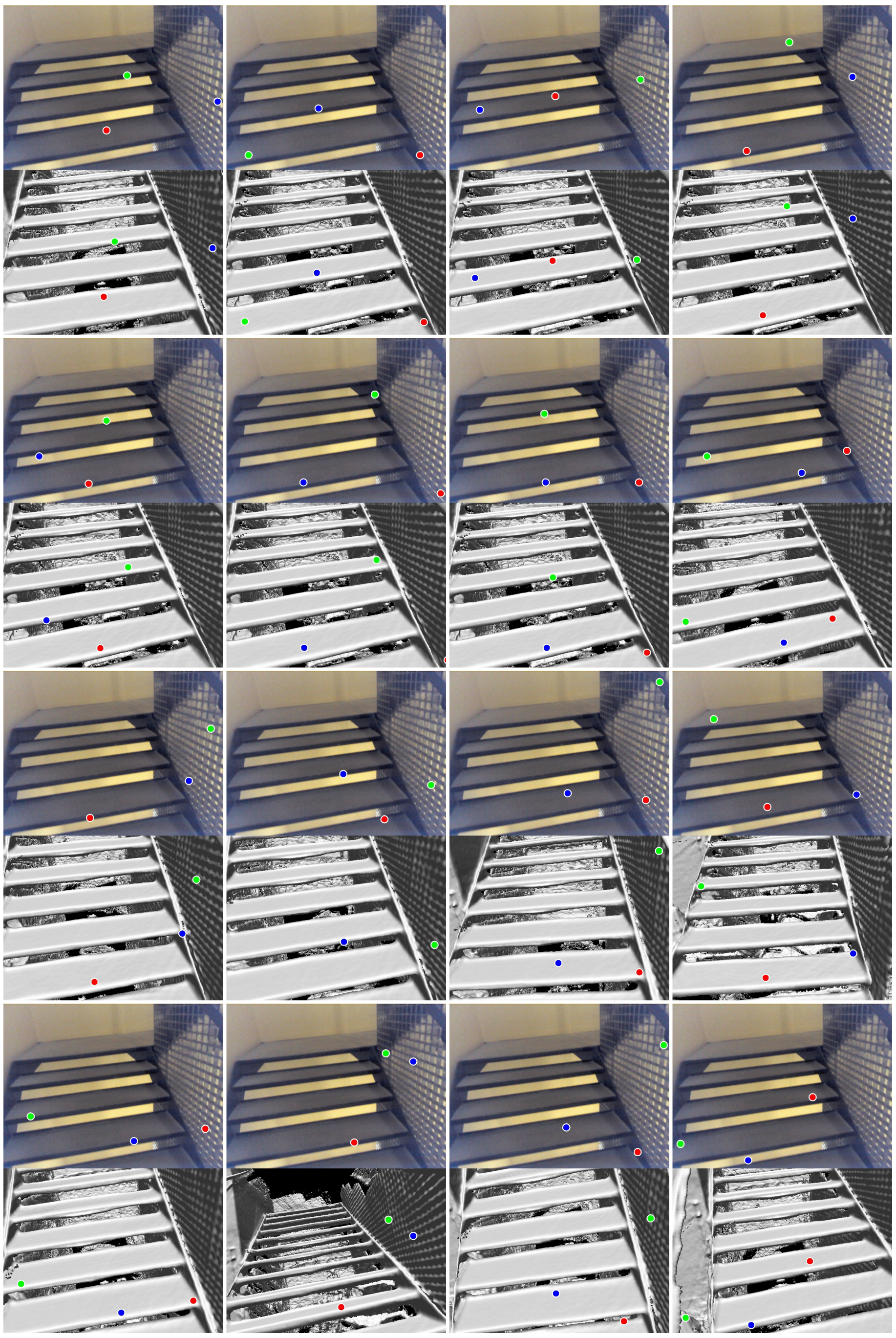


Figure 6.10: The top 16 pose candidates (left-to-right, top-to-bottom) corresponding to the failure case on the *Stairs* scene shown in Figure 6.8d. The coloured points indicate the 2D-to-3D correspondences that are used to generate the initial pose hypotheses. Note that in this case, none of the candidates would relocalise the camera successfully.

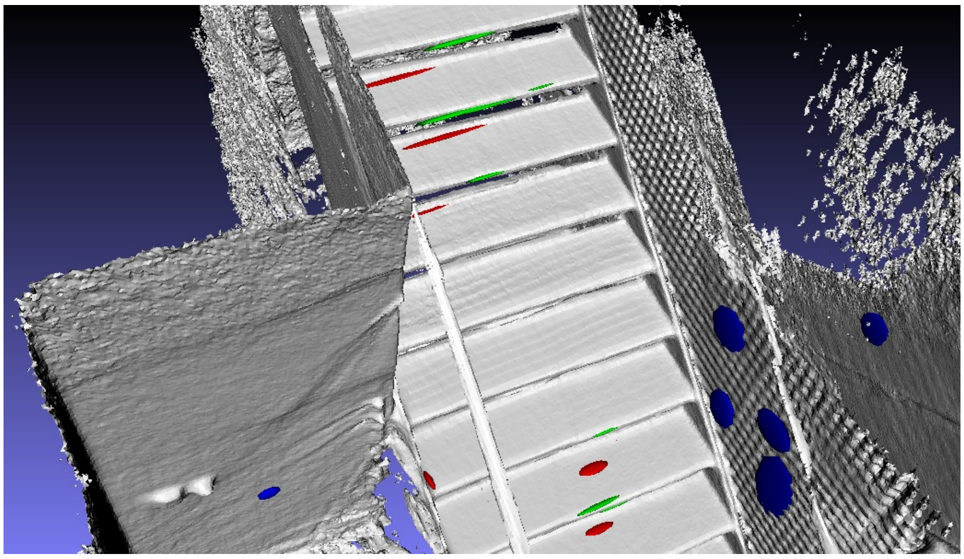


Figure 6.11: The modal clusters contained in the leaves for the points in the optimal camera pose hypothesis from Figure 6.10. It is noticeable that points at the same places on different steps end up in the same leaves, making the distributions in those leaves less informative.

6.4.6 *Further Successful Examples*

Some further examples of successful relocalisation, this time in the *Fire* scene from the 7-Scenes dataset [104], can be seen in Figure 6.12. As in Figure 6.9, it is noticeable that the relocaliser tries to rely on points in the scene that can be identified unambiguously where these are available, something that is clearly easier in sequences such as *Fire* that contain many easily-distinguished objects.

6.5 FINAL REMARKS

In recent years, offline approaches based on regression forests to predict 2D-to-3D correspondences [12, 55, 82, 104, 116] have been shown to achieve state-of-the-art camera relocalisation results, but their adoption for online relocalisation in practical systems such as KinectFusion [91], VoxelHashing [94] or InfiniTAM [64, 66] has been hindered by the need to train extensively on the target scene ahead of time.

In this chapter, we have shown that it is possible to circumvent this limitation by adapting offline-trained regression forests to novel

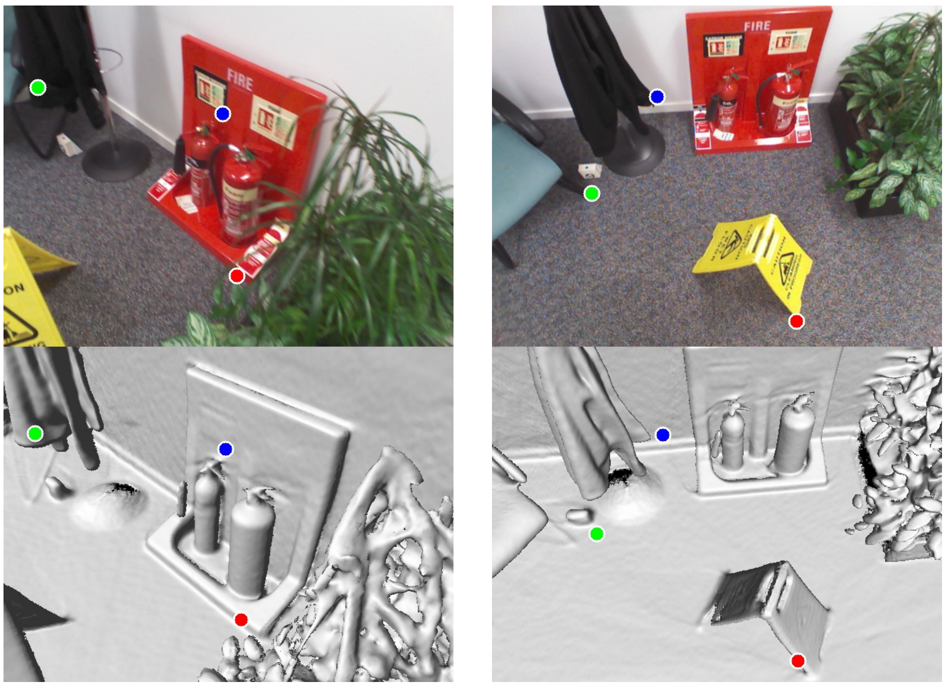


Figure 6.12: Further examples of successful relocalisation in the *Fire* scene from the 7-Scenes dataset [104]. To estimate the pose, the relocaliser tries to rely on points in the scene that can be identified unambiguously.

scenes online. Such adapted forests achieve relocalisation performance on the 7-Scenes Dataset by Shotton et al. [104] that is competitive with the offline-trained forests of existing methods, and our approach runs in under 150 ms, making it competitive for practical purposes with fast keyframe-based approaches such as random ferns [51, 64]. Unlike such approaches, we are also much better able to relocalise from novel poses, removing much of the need for the user to move the camera around when relocalising.

While not explicitly considered in this chapter, it would be feasible to also employ the adapted forests to perform semantic labelling of the input images, simultaneously to the pose regression task, by extending the reservoirs associated to the leaves to store category examples, produced, *e.g.* by a deep neural network [77] or by manual hints given by the user [52]. Such examples could then be aggregated in probability histograms for each leaf, as in SemanticPaint, and used to *quickly* label the RGB-D input frames: the relocalisation task as a whole takes less than 150 ms, but the majority of such time is spent in the Preemptive RANSAC estimation, while the feature computation and forest evaluation tasks take $\approx 3\,\mathrm{ms}$, thus potentially allowing the labelling of input frames in real time.

A robust relocalisation system such as the one described here is a desirable feature for 3D reconstruction pipelines (either based on semantic clues or not), allowing an easier interaction with users of such software: the need for slow and smooth camera movements is somewhat reduced if tracking failures caused by the lack of them can be reliably recovered from.

Part III

CONCLUSIONS

7 CONCLUSIONS

This thesis has been concerned with the topic of Semantic Simultaneous Localisation and Mapping. While in the last decades several research works tackled the theme of SLAM, resulting in the development of effective software pipelines, only recently the interest on high level informations has spurred research on the topic of *Semantic SLAM*.

This work has been divided in two logical parts, each presenting research on an aspect of localisation and mapping systems: in the first part, we focused our efforts on the *core* elements of Semantic SLAM systems, *i.e.* semantic camera tracking and reconstruction; whilst in the second part an ancillary – but nevertheless important – component, *i.e.* that performing camera relocalisation, was described.

3D reconstruction systems based on the processing of visual data such as those here described, in general, allow the attainment of geometrically accurate models of the observed environment, optionally endowed with colour informations. Only few recent works, mentioned in Section 2.3, started including semantic information in the generated reconstructions, by deploying manual hints given by the users of the system or machine learning techniques. Accordingly, in Chapter 3, we present a pipeline allowing the automatic labelling of generated models, by coupling the output of a state-of-the-art neural network for semantic segmentation of images with a 3D reconstruction pipeline such as the well known KinectFusion [91]. We show how, by properly integrating semantic labels for pixels pertaining to different images acquired over time, the categories associated with each element of the final model become robust to single (or relatively infrequent) errors by part of the semantic segmentation system.

We then extend the just described system, in Chapter 4, to deploy high level informations stored in the model being reconstructed during the camera pose estimation phase. By accounting for the semantics of items being reconstructed, we are able to overcome limitations of the original, geometry-based, camera tracking algorithm. Geometric tracking systems, such as the projective ICP approach by Newcombe et al. [91] or the direct alignment method by Bylow et al. [14], struggle in estimating the correct camera pose when the observed surface present few geometrically distinctive features (*e.g.* when the camera is observing a flat wall or floor). Conversely, by integrating a term accounting for the *semantic alignment* between the observed surface and the reconstructed model, we are able to successfully reconstruct scenes where the prior, geometry-based, algorithm would fail (*e.g.* flat walls adorned by flat paintings) or would perform poorly, exhibiting substantial amounts of drift.

By exploiting the presence of semantic information in both the tracking and mapping phases of the SLAM pipeline we are thus able to realise a virtuous "semantic loop", wherein the availability of high level informations by an image labelling algorithm allows to generate semantically accurate reconstruction that, in turn, improve the accuracy of camera pose estimation, allowing the subsequent fusion of additional semantic informations in the map, resulting – finally – in an overall improved model of the observed scene.

In Chapter 5 we tackle two shortcomings of the Semantic SLAM systems described earlier, namely the non real-time nature of the algorithms, due to high processing requirements of the semantic labelling neural network, and the limited extents of attainable reconstructions, due to the choice of a dense voxel-grid representation for the map (as in KinectFusion). By replacing the dense voxel grid used to store the reconstruction with a dynamically managed, hash-based, data structure, we are able to model large scale environments. Simultaneously, by deploying the labelling process on a separate GPU accelerator and migrating data between the main memory of the system and two video cards used by our software, we can attain a pipeline processing data at interactive rates, thus resulting in a pleasant experience for the final user of the software.

In the second and final part of this thesis we present an accurate camera relocalisation system, relying on the on-the-fly adaptation of pre-trained pose regression forests to novel scenes. In recent years, several research works proved that the deployment of decision forests, paired with robust implementations of RANSAC methods, is a viable tool to accurately estimate the position and bearings of a camera observing a known scene. Drawback of such methods, however, is the requirement for the forests to be trained in advance on the scene of interest, over the course of several minutes/hours, thus preventing the employment of those systems in the main loop of real-time reconstruction pipelines. In Chapter 6 we show how, instead of performing an ad-hoc training process for each observed scene, a *single* pre-trained decision forest can be *adapted*, *online*, to new scenes, with relocalisation performances on par with those of offline trained (and state-of-the-art) methods.

The availability of an accurate camera relocalisation algorithm is a desirable property for any simultaneous localisation and mapping pipeline for several reasons. While reliable, current camera pose estimation algorithms are not without any defects as they, sometimes, fail in their task, thus preventing the continuation of the SLAM loop. In such cases a forest-based relocalisation method, such as the one we propose, can take over and identify the likely camera pose, so as to restart the localisation and mapping iterations. Another state wherein the availability of a relocaliser could bring benefits to a SLAM pipeline is when the user wants to restart the reconstruction of a partially explored environment: the proposed system will be able to reliably provide an initial sensor pose and then cede control to the main loop. Additionally, even though we don't consider it explicitly in this work, a pose regression forest such as that we deploy in the relocalisation system is well suited to multiple tasks and could be adapted to *jointly* perform semantic labelling or object recognition, by relying on the feature vocabulary used by the pre-existing system.

Recent interest for topics such as virtual or augmented reality, self driving vehicles, and mobile mapping fostered the deployment of ever-improving SLAM pipelines. We posit that the integration of semantic informations in such systems will lead to improved performance, either due to the pure availability of such high level data in the maps (for processing by separate reasoning pipeline, *e.g.* to detect specific instances of objects of interest) or to the exploitation of such clues to improve quality and accuracy of the reconstructions.

By focusing the work performed in this thesis on the topic of simultaneous localisation and mapping and deploying high-level, semantic, information as core elements of the SLAM loop we hope to provide useful insights and deployable techniques to future researchers in the field.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

[1] Motilal Agrawal and Kurt Konolige. 'Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive gps'. In: *IEEE International Conference on Pattern Recognition*. Vol. 3. 2006, pp. 1063–1068 (cit. on p. 13).

[2] Nicholas Ayache and Olivier D. Faugeras. 'Building, Registrating, and Fusing Noisy Visual Maps'. In: *The International Journal of Robotics Research* 7.6 (1988), pp. 45–65 (cit. on p. 12).

[3] Hyojoon Bae, Michael Walker, Jules White, Yao Pan, Yu Sun and Mani Golparvar-Fard. 'Fast and scalable structure-from-motion based localization for high-precision mobile augmented reality systems'. In: *The Journal of Mobile User Experience* 5.1 (2016), pp. 1–21 (cit. on p. 95).

[4] Tim Bailey and Hugh Durrant-Whyte. 'Simultaneous Localization And Mapping (SLAM): Part II'. In: *IEEE Robotics & Automation Magazine* 13.3 (2006), pp. 108–117 (cit. on p. 12).

[5] Sid Yingze Bao, Mohit Bagra, Yu-Wei Chao and Silvio Savarese. 'Semantic Structure From Motion with Points, Regions, and Objects'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2703–2710 (cit. on p. 16).

[6] Sid Yingze Bao, Mohit Bagra and Silvio Savarese. 'Semantic Structure From Motion with Object and Point Interactions'. In: *International Conference on Computer Vision*. 2011, pp. 982–989 (cit. on p. 16).

[7] Sid Yingze Bao and Silvio Savarese. 'Semantic Structure from Motion'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2011, pp. 2025–2032 (cit. on p. 16).

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool. 'Speeded-up robust features (SURF)'. In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359 (cit. on pp. 13, 16).

[9] Paul Besl and Neil McKay. 'A Method for Registration of 3-D Shapes'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (1992), pp. 239–256 (cit. on pp. 14, 56, 106, 109).

[10] Gérard Blais and Martin D. Levine. 'Registering Multiview Range Data to Create 3D Computer Objects'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995), pp. 820–824 (cit. on p. 78).

[11] Jose-Luis Blanco. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Tech. rep. University of Malaga, 2010 (cit. on pp. 58, 79, 105).

[12] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold and Carsten Rother. 'Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. on pp. 98, 107, 110, 114, 121).

[13] Erik Bylow, Carl Olsson and Fredrik Kahl. 'Robust Camera Tracking by Combining Color and Depth Measurements'. In: *IEEE International Conference on Pattern Recognition*. 2014, pp. 4038–4043 (cit. on pp. 15, 19, 32, 53, 62, 63, 78).

[14] Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl and Daniel Cremers. 'Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions'. In: *Robotics: Science and Systems*. Vol. 9. 2013 (cit. on pp. 24, 25, 30, 56, 57, 78, 128).

[15] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid and John J. Leonard. 'Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age'. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332 (cit. on p. 12).

[16] Michael Calonder, Vincent Lepetit, Christoph Strecha and Pascal Fua. 'BRIEF: Binary Robust Independent Elementary Features'. In: *European Conference on Computer Vision*. 2010, pp. 778–792 (cit. on p. 96).

[17] Daniel R. Canelhas, Todor Stoyanov and Achim J. Lilienthal. 'SDF Tracker: A Parallel Algorithm for On-line Pose Estimation and Scene Reconstruction from Depth Images'. In: *International Conference on Intelligent Robots and Systems*. 2013, pp. 3671–3676 (cit. on pp. 24, 56, 78).

[18] Robert O. Castle, Georg Klein and David W. Murray. 'Video-rate Localization in Multiple Maps for Wearable Augmented Reality'. In: *IEEE International Symposium on Wearable Computers*. 2008, pp. 15–22 (cit. on p. 95).

[19] Robert O. Castle, Georg Klein and David W. Murray. 'Combining monoSLAM with object recognition for scene augmentation using a wearable camera'. In: *Image and Vision Computing* 28.11 (2010), pp. 1548–1556 (cit. on p. 16).

[20] Raja Chatila and Jean-Paul Laumond. 'Position referencing and consistent world modeling for mobile robots'. In: *International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 138–145 (cit. on p. 12).

[21] Jiawen Chen, Dennis Bautembach and Shahram Izadi. 'Scalable real-time volumetric surface reconstruction'. In: *ACM Transactions on Graphics* 32.4 (2013), p. 113 (cit. on p. 15).

[22] Yang Chen and Gérard Medioni. 'Object Modelling by Registration of Multiple Range Images'. In: *Image and Vision Computing* 10.3 (1992), pp. 145–155 (cit. on p. 79).

[23] Howie Choset and Keiji Nagatani. 'Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization'. In: *IEEE Transactions on robotics and automation* 17.2 (2001), pp. 125–137 (cit. on p. 13).

[24] Ondřej Chum, Jiří Matas and Josef Kittler. 'Locally Optimized RANSAC'. In: *Joint Pattern Recognition Symposium*. 2003, pp. 236–243 (cit. on p. 104).

[25] Javier Civera, Dorian Gálvez-López, Luis Riazuelo, Juan D. Tardós and J. M. M. Montiel. 'Towards Semantic SLAM using a Monocular Camera'. In: *International Conference on Intelligent Robots and Systems*. 2011, pp. 1277–1284 (cit. on p. 16).

[26] Javier Civera, Oscar G. Grasa, Andrew J. Davison and J. M. M. Montiel. '1-Point RANSAC for EKF Filtering. Application to Real-Time Structure from Motion and Visual Odometry'. In: *Journal of Field Robotics* 27.5 (2010), pp. 609–631 (cit. on pp. 13, 16).

[27] David M. Cole and Paul M. Newman. 'Using Laser Range Data for 3D SLAM in Outdoor Environments'. In: *International Conference on Robotics and Automation*. 2006, pp. 1556–1563 (cit. on p. 13).

[28] James L. Crowley. 'World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging'. In: *International Conference on Robotics and Automation*. 1989, pp. 674–680 (cit. on p. 12).

[29] Brian Curless and Marc Levoy. 'A Volumetric Method for Building Complex Models from Range Images'. In: *Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques*. 1996, pp. 303–312 (cit. on pp. 24, 26).

[30] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi and Christian Theobalt. 'BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration'. In: *arXiv preprint arXiv:1604.01093* (2016) (cit. on p. 16).

[31] Andrew J. Davison. 'Real-Time Simultaneous Localisation and Mapping with a Single Camera'. In: *International Conference on Computer Vision*. 2003, pp. 1403–1410 (cit. on p. 13).

[32] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton and Olivier Stasse. 'MonoSLAM: Real-Time Single Camera SLAM'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067 (cit. on p. 13).

[33] Lei Deng, Zhixiang Chen, Baohua Chen, Yueqi Duan and Jie Zhou. 'Incremental image set querying based localization'. In: *Neurocomputing* 208 (2016), pp. 315–324 (cit. on p. 97).

[34] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich and Shahram Izadi.

'Fusion4D: Real-time Performance Capture of Challenging Scenes'. In: *ACM Transactions on Graphics* 35.4 (2016), p. 114 (cit. on p. 16).

[35] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon and Shahram Izadi. '3D Scanning Deformable Objects with a Single RGBD Sensor'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 493–501 (cit. on p. 16).

[36] H.F. Durrant-Whyte. 'Uncertain Geometry in Robotics'. In: *IEEE Journal on Robotics and Automation* 4.1 (1988), pp. 23–31 (cit. on p. 12).

[37] Hugh Durrant-Whyte and Tim Bailey. 'Simultaneous Localization and Mapping: Part I'. In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110 (cit. on p. 12).

[38] Ethan Eade and Tom Drummond. 'Monocular SLAM as a Graph of Coalesced Observations'. In: *International Conference on Computer Vision*. 2007, pp. 1–8 (cit. on p. 13).

[39] David Eigen and Rob Fergus. 'Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture'. In: *International Conference on Computer Vision*. 2015, pp. 2650–2658 (cit. on pp. 18, 73).

[40] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers and Wolfram Burgard. '3-D mapping with an RGB-D camera'. In: *IEEE Transactions on Robotics* 30.1 (2014), pp. 177–187 (cit. on p. 14).

[41] Nicola Fioraio, Gregorio Cerri and Luigi Di Stefano. 'Towards Semantic KinectFusion'. In: *International Conference on Image Analysis and Processing*. 2013, pp. 299–308 (cit. on pp. 17, 19, 52).

[42] Nicola Fioraio and Luigi Di Stefano. 'Joint Detection, Tracking and Mapping by Semantic Bundle Adjustment'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. June 2013, pp. 1538–1545 (cit. on pp. 17, 19).

[43] Nicola Fioraio and Luigi Di Stefano. 'SlamDunk: Affordable Real-Time RGB-D SLAM'. In: *Workshop at the European Conference on Computer Vision*. 2014, pp. 401–414 (cit. on p. 14).

[44] Nicola Fioraio, Jonathan Taylor, Andrew Fitzgibbon, Luigi Di Stefano and Shahram Izadi. 'Large-Scale and Drift-Free Surface Reconstruction Using Online Subvolume Registration'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4475–4483 (cit. on pp. 15, 16, 95).

[45] Martin A. Fischler and Robert C. Bolles. 'Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography'. In: *Communications of the ACM* 24.6 (1981), pp. 381–395 (cit. on pp. 97, 100).

[46] Georgios Floros and Bastian Leibe. 'Joint 2D-3D Temporally Consistent Semantic Segmentation of Street Scenes'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2823–2830 (cit. on p. 17).

[47] Brian Fulkerson and Stefano Soatto. 'Really quick shift: Image segmentation on a GPU'. In: *European Conference on Computer Vision*. 2010, pp. 350–358 (cit. on p. 103).

[48] Dorian Gálvez-López and Juan D. Tardós. 'Real-Time Loop Detection with Bags of Binary Words'. In: *International Conference on Intelligent Robots and Systems*. 2011, pp. 51–58 (cit. on pp. 96, 97).

[49] Dorian Gálvez-López and Juan D. Tardós. 'Bags of Binary Words for Fast Place Recognition in Image Sequences'. In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197 (cit. on p. 97).

[50] Andrew P. Gee and Walterio Mayol-Cuevas. '6D Relocalisation for RGBD Cameras Using Synthetic View Regression'. In: *British Machine Vision Conference*. 2012, pp. 1–11 (cit. on pp. 96, 97).

[51] Ben Glocker, Jamie Shotton, Antonio Criminisi and Shahram Izadi. 'Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding'. In: *IEEE Transactions on Visualization and Computer Graphics* 21.5 (May 2015), pp. 571–583 (cit. on pp. 97, 99, 106, 109, 111–114, 122).

[52] Stuart Golodetz, Michael Sapienza, Julien P. C. Valentin, Vibhav Vineet, Ming-Ming Cheng, Victor A. Prisacariu, Olaf Kaehler, Carl Yuheng Ren, Anurag Arnab, Stephen L. Hicks, David W. Murray, Shahram Izadi and Philip H. S. Torr. 'SemanticPaint: Interactive Segmentation and Learning of 3D Worlds'. In: *SIGGRAPH Emerging Technologies (Demo)*. 2015 (cit. on pp. 17–19, 70, 95, 122).

[53] Saurabh Gupta, Pablo Arbelaez and Jitendra Malik. 'Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 564–571 (cit. on p. 43).

[54] Saurabh Gupta, Ross Girshick, Pablo Arbeláez and Jitendra Malik. 'Learning Rich Features from RGB-D Images for Object Detection and Segmentation'. In: *European Conference on Computer Vision*. 2014, pp. 345–360 (cit. on pp. 18, 60, 73).

[55] Abner Guzman-Rivera, Pushmeet Kohli, Ben Glocker, Jamie Shotton, Toby Sharp, Andrew Fitzgibbon and Shahram Izadi. 'Multi-Output Learning for Camera Relocalization'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1114–1121 (cit. on pp. 98, 107, 110, 114, 121).

[56] Ankur Handa, Thomas Whelan, John McDonald and Andrew J. Davison. 'A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM'. In: *International Conference on Robotics and Automation*. 2014, pp. 1524–1531 (cit. on pp. 59, 62, 63).

[57] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd. Cambridge University Press, 2004 (cit. on p. 97).

[58] Peter Henry, Dieter Fox, Achintya Bhowmik and Rajiv Mongia. 'Patch Volumes: Segmentation-based Consistent Mapping with RGB-D Cameras'. In: *International Conference on 3D Vision*. IEEE. 2013, pp. 398–405 (cit. on p. 16).

[59] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren and Dieter Fox. 'RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments'. In:

International Journal of Robotics Research 31.5 (2012), pp. 647–663 (cit. on p. 14).

[60] Alexander Hermans, Georgios Floros and Bastian Leibe. 'Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images'. In: *International Conference on Robotics and Automation*. 2014, pp. 2631–2638 (cit. on p. 17).

[61] In: *Stochastic Processes and Filtering Theory*. Ed. by Andrew H. Jazwinski. Vol. 64. Elsevier, 1970 (cit. on pp. 12, 16).

[62] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama and Trevor Darrell. 'Caffe: Convolutional Architecture for Fast Feature Embedding'. In: *International Conference on Multimedia*. 2014, pp. 675–678 (cit. on p. 43).

[63] Wolfgang Kabsch. 'A solution for the best rotation to relate two sets of vectors'. In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–923 (cit. on pp. 97, 100, 104).

[64] Olaf Kähler, Victor A. Prisacariu and David W. Murray. 'Real-Time Large-Scale Dense 3D Reconstruction with Loop Closure'. In: *European Conference on Computer Vision*. 2016, pp. 500–516 (cit. on pp. 15, 16, 91, 95, 102, 106, 109, 111, 113, 114, 121, 122).

[65] Olaf Kähler, Victor A. Prisacariu, Julien Valentin and David W. Murray. 'Hierarchical Voxel Block Hashing for Efficient Integration of Depth Images'. In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 192–197 (cit. on p. 15).

[66] Olaf Kähler*, Victor A. Prisacariu*, Ren C. Yuheng, Xin Sun, Philip H. S. Torr and David W. Murray. 'Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices'. In: *IEEE Transactions on Visualization and Computer Graphics* 21.11 (2015), pp. 1241–1250 (cit. on pp. 15, 18, 52, 76, 95, 109, 121).

[67] Rudolph Emil Kalman. 'A new approach to linear filtering and prediction problems'. In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45 (cit. on p. 12).

[68] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich and Andreas Kolb. 'Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion'. In: *International Conference on 3D Vision*. 2013, pp. 1–8 (cit. on p. 13).

[69] Alex Kendall, Matthew Grimes and Roberto Cipolla. 'PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization'. In: *International Conference on Computer Vision*. 2015, pp. 2938–2946 (cit. on p. 97).

[70] Jonghyuk Kim and Salah Sukkarieh. 'SLAM aided GPS/INS Navigation in GPS Denied and Unknown Environments'. In: *The 2004 International Symposium on GNSS/GPS*. Vol. 4. 2004 (cit. on p. 13).

[71] Jonghyuk Kim and Salah Sukkarieh. 'Real-time implementation of airborne inertial-SLAM'. In: *Robotics and Autonomous Systems* 55.1 (2007), pp. 62–71 (cit. on p. 13).

[72] Young Hoon Lee and Gérard Medioni. 'RGB-D camera based wearable navigation system for the visually impaired'. In: *Computer Vision and Image Understanding* 149 (2016), pp. 3–20 (cit. on p. 95).

[73] John J. Leonard and Hugh F. Durrant-Whyte. 'Simultaneous map building and localization for an autonomous mobile robot'. In: *International Conference on Intelligent Robots and Systems Workshops*. 1991, pp. 1442–1447 (cit. on p. 12).

[74] Stefan Leutenegger, Margarita Chli and Roland Y. Siegwart. 'BRISK: Binary Robust Invariant Scalable Keypoints'. In: *International Conference on Computer Vision*. 2011, pp. 2548–2555 (cit. on p. 97).

[75] Kenneth Levenberg. 'A Method for the Solution of Certain Problems in Least Squares'. In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168 (cit. on pp. 59, 105).

[76] Shuda Li and Andrew Calway. 'RGBD Relocalisation Using Pairwise Geometry and Concise Key Point Sets'. In: *International Conference on Robotics and Automation*. 2015, pp. 6374–6379 (cit. on p. 97).

[77] Jonathan Long, Evan Shelhamer and Trevor Darrell. 'Fully Convolutional Networks for Semantic Segmentation'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440 (cit. on pp. 18, 19, 43, 44, 47, 48, 60, 69, 72, 73, 84, 86, 89, 122).

[78] William E. Lorensen and Harvey E. Cline. 'Marching cubes: A high resolution 3D surface construction algorithm'. In: *ACM SIGGRAPH Computer Graphics*. Vol. 21. 4. 1987, pp. 163–169 (cit. on pp. 34, 35, 84).

[79] David G. Lowe. 'Distinctive Image Features from Scale-Invariant Keypoints'. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110 (cit. on p. 13).

[80] Guoyu Lu, Yan Yan, Li Ren, Jingkuan Song, Nicu Sebe and Chandra Kambhamettu. 'Localize Me Anywhere, Anytime: A Multi-task Point-Retrieval Approach'. In: *International Conference on Computer Vision*. 2015, pp. 2434–2442 (cit. on p. 98).

[81] Donald W. Marquardt. 'An Algorithm for Least-Squares Estimation of Nonlinear Parameters'. In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963) (cit. on pp. 59, 105).

[82] Daniela Massiceti, Alexander Krull, Eric Brachmann, Carsten Rother and Philip H S Torr. 'Random Forests versus Neural Networks – What's Best for Camera Localization?' In: *arXiv preprint arXiv:1609.05797* (2016) (cit. on pp. 98, 114, 121).

[83] John McCormac, Ankur Handa, Andrew Davison and Stefan Leutenegger. 'SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks'. In: *arXiv preprint arXiv:1609.05130* (2016) (cit. on pp. 17, 18).

[84] Ondrej Miksik, Vibhav Vineet, Morten Lidegaard, Ram Prasaath, Matthias Nießner, Stuart Golodetz, Stephen L. Hicks, Patrick Pérez, Shahram Izadi and Philip H. S. Torr. 'The Semantic Paintbrush: Interactive 3D Mapping and Recognition in Large Outdoor Spaces'. In: *Annual ACM Conference on Human Factors in Computing Systems*. 2015, pp. 3317–3326 (cit. on pp. 17–19).

[85] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun and Alan Yuille. 'The Role of Context for Object Detection and Semantic Segmentation in the Wild'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 891–898 (cit. on p. 60).

[86] Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós. 'ORB-SLAM: A Versatile and Accurate Monocular SLAM System'. In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1147–1163 (cit. on pp. 14, 97).

[87] Raúl Mur-Artal and Juan D. Tardós. 'Fast Relocalisation and Loop Closing in Keyframe-Based SLAM'. In: *International Conference on Robotics and Automation*. 2014, pp. 846–853 (cit. on p. 95).

[88] Kevin P. Murphy. 'Bayesian Map Learning in Dynamic Environments'. In: *Annual Conference on Neural Information Processing Systems*. 1999 (cit. on p. 12).

[89] Richard A. Newcombe and Andrew J. Davison. 'Live Dense Reconstruction with a Single Moving Camera'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2010, pp. 1498–1505 (cit. on p. 13).

[90] Richard A. Newcombe, Dieter Fox and Steven M. Seitz. 'DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 343–352 (cit. on pp. 16, 52).

[91] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges and Andrew Fitzgibbon. 'KinectFusion: Real-Time Dense Surface Mapping and Tracking'. In: *International Symposium on Mixed and Augmented Reality*. 2011, pp. 127–136 (cit. on pp. 14, 17, 19, 23, 24, 26, 29, 32, 34, 51, 52, 56, 74, 78, 95, 106, 121, 127, 128).

[92] Richard A. Newcombe, Steven J. Lovegrove and Andrew J. Davison. 'DTAM: Dense Tracking and Mapping in Real-Time'. In: *International Conference on Computer Vision*. 2011, pp. 2320–2327 (cit. on p. 13).

[93] Paul M. Newman, David M. Cole and Kin Ho. 'Outdoor SLAM using Visual Appearance and Laser Ranging'. In: *International Conference on Robotics and Automation*. 2006, pp. 1180–1187 (cit. on p. 13).

[94] Matthias Nießner, Michael Zollhöfer, Shahram Izadi and Marc Stamminger. 'Real-time 3D Reconstruction at Scale using Voxel Hashing'. In: *ACM Transactions on Graphics* 32.6 (2013), p. 169 (cit. on pp. 15, 18, 52, 68, 70, 74, 76, 90, 106, 121).

[95] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen and Peter-Pike Sloan. 'Interactive ray tracing for isosurface rendering'. In: *Proceedings of the Conference on Visualization*. 1998, pp. 233–238 (cit. on p. 34).

[96] Rémi Paucher and Matthew Turk. 'Location-based augmented reality on mobile phones'. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2010, pp. 9–16 (cit. on p. 95).

[97] Nicolas Loy Rodas, Fernando Barrera and Nicolas Padoy. 'Marker-less AR in the Hybrid Room using Equipment Detection for Camera Relocalization'. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2015, pp. 463–470 (cit. on p. 95).

[98] Henry Roth and Marsette Vona. 'Moving Volume KinectFusion'. In: *British Machine Vision Conference*. 2012, pp. 1–11 (cit. on p. 15).

[99] Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski. 'ORB: an efficient alternative to SIFT or SURF'. In: *International Conference on Computer Vision*. IEEE. 2011, pp. 2564–2571 (cit. on pp. 13, 97).

[100] Szymon Rusinkiewicz and Marc Levoy. 'Efficient variants of the ICP algorithm'. In: *International Conference on 3-D Digital Imaging and Modeling*. IEEE. 2001, pp. 145–152 (cit. on pp. 14, 17).

[101] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H. J. Kelly and Andrew J. Davison. 'SLAM++: Simultaneous Localisation and Mapping at the Level of Objects'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1352–1359 (cit. on pp. 17, 19).

[102] Torsten Sattler, Bastian Leibe and Leif Kobbelt. 'Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 99 (2016) (cit. on p. 97).

[103] Toby Sharp. 'Implementing Decision Trees and Forests on a GPU'. In: *European Conference on Computer Vision*. 2008, pp. 595–608 (cit. on p. 103).

[104] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi and Andrew Fitzgibbon. 'Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2930–2937 (cit. on pp. 98, 103, 104, 107, 108, 110, 113–116, 118, 121, 122).

[105] Nathan Silberman, Derek Hoiem, Pushmeet Kohli and Rob Fergus. 'Indoor Segmentation and Support Inference from RGBD Images'. In: *European Conference on Computer Vision*. 2012, pp. 746–760 (cit. on p. 60).

[106] Randall C. Smith and Peter Cheeseman. 'On the Representation and Estimation of Spatial Uncertainty'. In: *International Journal of Robotics Research* 5.4 (1986), pp. 56–68 (cit. on p. 12).

[107] Randall Smith, Matthew Self and Peter Cheeseman. 'Estimating Uncertain Spatial Relationships in Robotics'. In: *Autonomous robot vehicles*. Springer, 1990, pp. 167–193 (cit. on p. 12).

[108] H. W. Sorenson. *Kalman Filtering: Theory and Application*. IEEE Press selected reprint series. IEEE Press, 1985 (cit. on pp. 12, 16).

[109] Frank Steinbrücker, Jürgen Sturm and Daniel Cremers. 'Real-Time Visual Odometry from Dense RGB-D Images'. In: *International Conference on Computer Vision Workshops*. 2011, pp. 719–722 (cit. on p. 78).

[110] Hauke Strasdat. 'Local Accuracy and Global Consistency for Efficient Visual SLAM'. PhD thesis. Imperial College London, 2012 (cit. on pp. 58, 79, 105).

[111] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard and Daniel Cremers. 'A benchmark for the evaluation of RGB-D SLAM systems'. In: *International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580 (cit. on pp. 59, 62, 63, 66, 67).

[112] Ivan E. Sutherland. 'Three-dimensional data input by tablet'. In: *Proceedings of the IEEE* 62.4 (1974), pp. 453–461 (cit. on p. 97).

[113] Sebastian Thrun, Wolfram Burgard and Dieter Fox. *Probabilistic robotics*. MIT Press, 2005 (cit. on p. 12).

[114] Sebastian Thrun and John J. Leonard. 'Simultaneous Localization And Mapping'. In: *Springer handbook of robotics*. Springer, 2008, pp. 871–889 (cit. on p. 12).

[115] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip H. S. Torr, Shahram Izadi and Cem Keskin. 'Learning to Navigate the Energy Landscape'. In: *arXiv preprint arXiv:1603.05772* (2016) (cit. on p. 97).

[116] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi and Philip H.S. Torr. 'Exploiting Uncertainty in Regression Forests for Accurate Camera Relocalization'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4400–4408 (cit. on pp. 98–100, 102, 104, 105, 107, 109, 110, 114, 121).

[117] Julien Valentin, Vibhav Vineet, Ming-Ming Cheng, David Kim, Jamie Shotton, Pushmeet Kohli, Matthias Nießner, Antonio Criminisi, Shahram Izadi and Philip Torr. 'SemanticPaint: Interactive 3d Labeling and Learning at your Fingertips'. In: *ACM Transactions on Graphics* 34.5 (2015), p. 154 (cit. on pp. 17–19, 70, 90, 95, 102).

[118] Jeffrey Scott Vitter. 'Random Sampling with a Reservoir'. In: *ACM Transactions on Mathematical Software* 11.1 (1985), pp. 37–57 (cit. on p. 102).

[119] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J. Leonard and John McDonald. 'Robust Real-Time Visual Odometry for Dense RGB-D Mapping'. In: *International Conference on Robotics and Automation*. 2013, pp. 5724–5731 (cit. on p. 78).

[120] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard and John McDonald. *Kintinuous: Spatially Extended Kinectfusion*. Tech. rep. Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, 2012 (cit. on p. 15).

[121] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker and Andrew J. Davison. 'ElasticFusion: Dense SLAM Without A Pose Graph'. In: *Robotics: Science and Systems*. 2015 (cit. on pp. 13, 18, 95).

[122] Brian Williams, Georg Klein and Ian Reid. 'Automatic Relocalization and Loop Closing for Real-Time Monocular SLAM'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.9 (Sept. 2011), pp. 1699–1712 (cit. on p. 97).

[123] Jianxiong Xiao, Andrew Owens and Antonio Torralba. 'SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels'. In: *International Conference on Computer Vision*. 2013, pp. 1625–1632 (cit. on pp. 17, 22, 38, 40–47, 60, 65).

[124] Ming Zeng, Fukai Zhao, Jiaxiang Zheng and Xinguo Liu. 'Octree-based fusion for realtime 3D reconstruction'. In: *Graphical Models* 75.3 (2013), pp. 126–136 (cit. on p. 15).

[125] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang and Philip H. S. Torr. 'Conditional Random Fields as Recurrent Neural Networks'. In: *International Conference on Computer Vision*. 2015, pp. 1529–1537 (cit. on pp. 18, 73).

[126] Qian-Yi Zhou and Vladlen Koltun. 'Dense Scene Reconstruction with Points of Interest'. In: *ACM Transactions on Graphics* 32.4 (2013), p. 112 (cit. on pp. 16, 88, 89).

[127] Qian-Yi Zhou and Vladlen Koltun. ‘Depth Camera Tracking with Contour Cues’. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 632–638 (cit. on pp. 15, 19, 52, 53, 64).

[128] Qian-Yi Zhou, Stephen Miller and Vladlen Koltun. ‘Elastic Fragments for Dense Scene Reconstruction’. In: *International Conference on Computer Vision*. 2013, pp. 473–480 (cit. on pp. 16, 88).