Imperial College London
Department of Computing

# Neural Scene Representations for Dense-Semantic SLAM

Edgar Sucar

16th October 2023

Supervised by Prof. Andrew Davison

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

**Copyright Declaration**

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial-No Derivatives 4.0 International Licence (CC BY-NC-ND).

Under this licence, you may copy and redistribute the material in any medium or format on the condition that; you credit the author, do not use it for commercial purposes and do not distribute modified versions of the work.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

**Abstract**

An important challenge in visual Simultaneous Localisation and Mapping (SLAM) has been on the design of scene representations that allow for both robust inference and useful interaction. The rapid progression of semantic image understanding powered by deep learning has led to SLAM systems that enrich geometric maps with semantics, which increases the range of applications possible. However, a core challenge remains in how to tightly integrate geometry and semantics for 3D reconstruction; we believe that their joint representation is the right direction for actionable and robust maps. In this thesis we will address the central question on designing efficient scene representations by the use of compressive models, which can represent detail with the least number of parameters. We then demonstrate that compressive models offer a solution for the joint representation of geometry and semantics, where semantics provide priors for robust reconstruction and geometric compression informs scene decomposition.

This work focuses on using generative neural networks, a category of compressive representations, for incremental dense SLAM. We develop a volumetric rendering formulation for the use of compressive models in generative inference from multi-view images, enabling two novel SLAM systems. First, we learn class-level code descriptors for object shape from aligned 3D models. At test time, the code and object pose are optimised for efficient and complete object reconstruction from instances of the learned categories. This method relaxes the assumption of fixed templates and allows for intra-class shape variation. We demonstrate the usefulness of semantic priors for complete and precise reconstruction in a robotic packing application. Second, we present a scene-specific multi-layered perceptron (MLP) neural field for full generative dense SLAM. Our results show that it allows for efficient mapping, automatic hole-filling, and joint optimisation of camera trajectory and 3D map. Last, we demonstrate that the MLP's automatic scene compression discovers underlying scene structures that are revealed with sparse labeling.

**Acknowledgements**

First and foremost I would like to thank my supervisor Andy for giving me the opportunity to have a unique and amazing PhD experience. His brilliant guidance taught me how to identify and pursue important research questions. His desire to tackle the hard problems coupled with his enthusiasm for research motivated for me to pursue and develop my interests in SLAM. And in particular his trust in my abilities and excitement for my work gave me the confidence to try new ideas and be willing to venture into unexplored territory. I learned from him the usefulness and fun of live demonstrations. I also greatly appreciate his fostering of a collaborative environment in the lab which made it a great place to be in.

I would particularly like to acknowledge and thank my close colleagues and friends Kentaro Wada and Joe Ortiz from whom I collaborated and learned a lot. And in general I would like to thank the other members of the Dyson Robotics Lab whom I had great discussions and learned a lot but also formed good friendships: Tristan Laidlow, Eric Dexheimer, Dorian Henning, Ivan Kapelyukh, Xin Kong, Zoe Landgraf, Shikun Liu, Hidenobu Matsuki, Kirill Mazur, Aalok Patwardhan, Marwan Taher, Anagh Malik, and Shuaifeng Zhi.

I would like to thank the lab manager Iosifina for all her help in making my PhD a smooth experience. I would like to thank Dyson and CONACyT for sponsoring my PhD.

Finally I would like to thank my parents and sister for their unconditional love and support throughout. I could always feel their encouragement, which kept me going even in the difficult times. I would particularly would like to thank my father for fostering in me a a curiosity and love for learning.

# Contents

# List of Tables

# List of Figures

Chapter 1

# Introduction

Contents



## 1.1 Spatial Artificial Intelligence

The goal of *artificial intelligence* is to develop autonomous programs which aid humans to either automate processes or make decisions. This could be in a variety of domains such as medicine, economy, translation, recommendation systems, factory automation, transportation, etc. This thesis is centered on a sub area of *artificial intelligence* we call Spatial Artificial Intelligence or **Spatial AI**. The main characteristic of Spatial AI is that we are concerned about making decisions about a physical spatial environment.

The key element of an embodied Spatial AI system is a moving platform with sensors to capture information about its environment and possibly actuators to interact and change its surroundings. Two significant instances of a Spatial AI applications are *augmented reality AI assistants* and *robotics*, see Figure 1.1. The goal of an augmented reality AI assistant is to augment the spatial capabilities of a human through a wearable device such as augmented reality glasses. This could include capabilities such as a spatial memory of the objects which a person has interacted with to retrieve lost items, navigation instructions in a new indoor space, or a virtual sports coach.

In robotics we are interested in automating a physical process through a robotic platform. This could be a passive task where no interaction is involved such as inspection of a factory by a flying drone, but normally the most useful robotic tasks involve some sort of scene interaction and manipulation. With interactive robotics there is a diversity of environments and tasks. At one hand of the spectrum we have specialized settings, where the range of objects and/or interaction types is limited. One example is warehouse robotics where the range of objects to be manipulated may be limited, for example boxes of different sizes or products of a given category, as well as the range of interaction types such as packing or stacking.

However, most of the biggest challenges in Spatial AI research are towards general purpose robotics. One example of such a setting is a commercial future house hold robot helper. This robot should work in a wide variety of homes with completely different layouts and object instances. It should be able to perform varied tasks such as tidying a room, loading and unloading a dishwasher, wiping and vacuuming surfaces with different materials, or cooking a meal, and it should interact with humans, for example to manually assist them. Additionally the robot should be able to continuously learn new things and adapt, to recognize a new object it had not previously seen, to perform a new tasks from a human demonstration, or to fix a mistake in its performance through a user correction.

Figure 1.1: (a) A futuristic house hold humanoid cleaning robot. (b) Augmented reality glasses used in collaborative architecture.

## 1.2 Scene Understanding

Two high level capabilities necessary for most Spatial AI robotic systems are *navigation* and *manipulation*. Navigation is the ability to move around a space, to go from point A to a point B, and this could be navigation on a 2D surface for a ground mobile robot or in 3D space for a flying robot. Manipulation is the ability to interact with objects, to change their position and configuration in order to achieve a goal, and is done with a robotic arm end defector, such as a gripper. The design of these capabilities is normally broken up into the following modules: *scene understanding, motion planning*, and *control*.

Scene understanding consists of inferring and modeling what is in the environment such as what is the geometry of the space, how can it be broken down into movable objects, or what are the physical properties of the objects such as mass and material. Motion planning is then about figuring out a set of actions to achieve a desired goal; in navigation, for example, motion planning is about finding a set of valid configurations to go from pose A to pose B while avoiding obstacles. Finally control is about going from high level actions into low level motor command for the robot. This pipeline can be executed in an *open loop* fashion where for a given goal or subgoal each module is performed sequentially and independently, or in a *closed loop*, where scene understanding, planning and control are continuously running jointly with feedback, this tends to be more robust as errors in one module can be corrected.

The scope of this thesis will be the scene understanding or perception component of Spatial AI. Improvements in this area have the potential of unlocking new capabilities in planning and control.

## 1.3 Machine Learning

Machine Learning ML has played an important role in the development of AI systems in recent years. ML contrasts with *hand crafted* methods in that data is used to *learn* a functional mapping from inputs to outputs. This is done by tuning the parameters

of a black box model, a process referred to as *training*, to optimise its performance on a set of samples, *the training data set*, where the mapping from inputs to outputs is known, with the purpose of generalising to new inputs with unknown outputs. One popular choice for the black box model, and which we will employ in different aspects of the thesis, is a *Neural Network*, which consists of stacked blocks of linear functions followed by a nonlinear activation function.

With the flexibility of black box models, a Spatial AI pipeline (Scene Understanding, Planning, and Control) can be broken into modules in varied ways. With a module defined as a subroutine where inputs and outputs are interpretable variables. Therefore a Spatial AI system can be composed of a combination of both designed and learned components. At one end of the spectrum are approaches that encompass a full robotic stack with a single learned module that maps sensor observations directly to motor controls, trained *end to end* either on demonstrations of the task to be performed (*imitation learning*), or through an indirect downstream *reward function* judging the performance of the task (*reinforcement learning*).

The complete lack of structural knowledge in end-to-end methods implies that they require more training data, and the lack of modules limits the re-usability in different settings. We argue that this is why these approaches are not practical in many problems outside a tightly constrained task. A modular approach with intermediate representations, including objects, scenes, etc., provides more flexibility, so that the learned components can be applied to different tasks. It also facilitates the integration of prior knowledge, such as physical laws and geometrical properties, at different stages in this pipeline.

## 1.4 3D Representation

There is one common element in the Spatial AI systems we have described, the need to move and take decisions in a 3D space, which requires a persistent *memory* to plan and reason jointly about the different elements of a scene. This is the reason that we argue that a persistent 3D scene representation should be explicitly

modeled in a Spatial AI pipeline. This representation should allow a device to operate for extended periods in the same space, plan and execute varied tasks, and allow communication with a human user. There are a lot of options about on how to design a representation to meet these requirements, with a combination of both designed and learned elements. The thesis will explore this research area: **3D Scene Representation**.

A 3D representation is the design choice of a **3D map** or *scene summary* storing the spatial contents of a scene. As a robot moves around a scene it needs to both find its position with respect to the current map: *localisation*, and update the state of the map with new observations: *mapping*. In this thesis we will concentrate on visual sensing, where observations are coming from a moving camera. This is because of the richness of information captured by cameras and their affordability.

As images are a projection of the underlying 3D geometry of the environment, mapping is related to inverting the physical image formation process. Our 3D representation must also therefore model the relationship between the 3D map and the visual observations. *Inference* is the process of recovering a complete 3D map from a set of 2D observations. There are generally 2 strategies for inference: *discriminative* and *generative* approaches. In discriminative or bottom up (from pixels to map) approaches the image formation model is not explicitly modeled, but inverted through machine learning. An example this could be an object detector mapping from images to 3D object bounding boxes. More common in 3D reconstruction are generative methods, which explicitly model a *measurement function* that transforms the 3D map into a 2D observation, and then inference is done through optimisation. Many practical systems have a combination of discriminative and generative components as in the system we present in Chapter 4.

## 1.5 Simultaneous Localisation and Mapping

For many applications, a robot must be reactive to its environment and take decisions in real-time. For example a flying drone cannot take seconds to plan if an

object is coming towards it, or time constraints may be of importance, such as in warehouse robotics where efficiency is critical. This adds additional constraints on the scene inference problem, it must be solved *incrementally*, that is from a stream of incoming images rather than a batch of data, and *online*; the computation time for each new image should be fast enough to match the rate of change of the desired application. Additionally the localisation and mapping problems are simultaneous objectives; in order to recover its position the robot must have a map of the environment, and in order to extend the map with new observations the robot must have an estimate of its position. This problem, referred to as *Simultaneous Localisation and Mapping* or *SLAM*, is a central component in most robotic systems, and will encapsulate the methods developed in this thesis.

A key property of SLAM is that it forms a closed loop system, where there is continuous feedback between tracking (estimating the pose and orientation of the sensor) and mapping. This is enabled by a persistent map summarising past observations, and the process of *data association*, between the latest observation and the corresponding elements in the map. SLAM contrasts with open loop localisation systems, where estimation is performed frame to frame without a persistent map, and drift accumulates even when observing a co-visible region.

The choice of scene representation for SLAM systems is tied to the estimation methods used for scene inference, and the applications it enables. Sparse 3D point clouds have been shown to enable very precise localisation though non-linear optimisation, but lack expressiveness for other tasks that involve interaction or navigation. Dense SLAM which aims to represent the full geometry of a scene opens up applications in navigation and motion planning, but present challenges in memory resources and normally resort to optimisation approximations. Finally, semantic SLAM methods augment dense maps with class information which is generally useful for going beyond navigation towards interaction with a scene.

## 1.6 Semantics Abstractions

For model-based robotics the available information in the scene representation is directly related to the ability to plan how to solve a given task. For example, if a robot aims to pick and place an object, the map must delineate its extents and separation from the rest of the scene; if the robot is to align two objects of the same class it must know their pose; if a mobile platform must navigate around a room it must identify the surface it can move on: the floor; or if the robot aims to push a box it must infer its weight to plan the force required to move the object. The map then serves as a simulator of the world a robot can use to test for the result of its actions to plan how to achieve a certain goal. In order to reason efficiently about the elements in the scene the robot must use *abstractions* which are groupings of finer elements with similar characteristics or a shared property. For example *objects* are abstractions of elements that move coherently, or *semantic classes* are grouping of objects that share a common function.

Machine learning has been used to perform abstractions on data through the process of *supervised learning*. This is done by creating datasets, where manually label elements are grouped together (commonly in images), of either object instances or semantic classes. 3D shape estimation and abstraction are then normally treated as independent, loosely coupled modules, for example by fusing semantic image predictions from a CNN network on top of an already reconstructed 3D map.

**Box 1.1: Research hypothesis**

The efficient representation (or compression) of a scene, that is, how to represent detail with the fewest parameters is closely related to the ability to abstract or group elements which share an underlying structure. Further, the ability to be efficient in representation by abstraction, either reusing common elements or finding underlying geometric structures, should aid in regularising noisy observations or completing missing data.

The central question we will be looking at in this thesis is, how can we jointly

model scene abstraction/compression and geometric shape inference, with feedback between each other. We pose the hypothesis shown in Box 1.1.

Using abstractions for shape inference has been explored in previous works we will discuss in Section 2. However the main technical contributions of this thesis revolve around how to use a new category of compressive representations, *generative 3D neural networks*, within the framework of incremental and real-time dense SLAM. Thus, we will explore the use of **Neural Scene representation**, to build a new line of SLAM systems where abstraction and shape are modeled jointly, which we define as **Tightly-Coupled Dense Semantic SLAM**. We will investigate the design and properties of this concept in three main projects outlined below:

- **NodeSLAM: Neural Object Descriptors** (Chapter 4) A generative auto-encoder neural network is trained to compress a dataset of 3D shapes of known semantic classes (common in table top settings: *mugs*, *bowls*, *cans*, and *bottles*), into a compact and optimisable latent space representation. We formulate a probabilistic volume rendering measurement function for multi-view shape optimisation with respect to masked depth images (obtained with an object detector), from which we build a SLAM system with object landmarks. We demonstrate the ability of the system to complete partial observations of the trained classes enabling robotic planning tasks such as object packing and stacking.

- **iMAP: iMAP: Neural Fields for Dense SLAM** (Chapter 5) A Multi-Layer Perceptron (MLP) neural network representation is used for reconstructing a room scale scene. The network is trained from scratch incrementally and in real-time to fit depth data observations within a dense SLAM framework. We demonstrate the representation capabilities for automatic scene compression with dynamic level of detail and the capacity to fill in holes where data is missing.

- **iLabel: Interactive Neural Scene Segmentation** (Chapter 6) Building

on top of iMAP, we demonstrate that the compressive properties of the MLP representation present an inherent decomposition of the scene into coherent objects which can be revealed through interactive sparse user annotations.

## 1.7 Thesis Structure

The remainder of this thesis is structured as follows:

**Chapter 2** provides an overview of the representative literature works in sparse and dense real-time SLAM, compressive representations, and semantic mapping; it discusses their characteristics and open challenges and places our work within their context.

**Chapter 3** introduces basic notation, the concepts used for optimisation in SLAM, and provides a primer on deep neural networks and volume rendering.

**Chapter 4** presents *NodeSLAM*, a system for object level mapping with learned class-level shape descriptors.

**Chapter 5** describes *iMAP*, a real-time SLAM system with a scene-specific MLP neural field representation.

**Chapter 6** builds on top of iMAP and shows that scene specific compression discovers underlying structures revealed with sparse labeling, demonstrated with the interactive system *iLabel*.

**Chapter 7** concludes the thesis with a discussion of the research presented and suggestions for future work.

## 1.8 Publications

The work described in this thesis resulted in the following publications:

- Sucar, E., Wada, K., Davison, A. (2020), **NodeSLAM: Neural Objects**

**Descriptors for Multi-View Shape Reconstruction**. In *Proceedings of the International Conference on 3D Vision (3DV)*. [Sucar et al., 2020].

- Sucar, E., Liu, S., Ortiz, J., Davison, A. (2021), **iMAP: Implicit Mapping and Positioning in Real-Time**. In *Proceedings of the International Conference on Computer Vision (ICCV)*. [Sucar et al., 2021].

- Zhi, S.*, Sucar, E.*, Mouton, A., Haughton, I., Laidlow, T., Davison, A. (2022). **iLabel: Revelaing Objects in Neural Fields**. In *Proceedings of the IEEE Robotics and Automation Letters (RA-L))*. [Zhi et al., 2022]. (* denotes joint first author.)

- Haughton, I., Sucar, E., Mouton, A., Johns, E., Davison, A. (2022). **Real-time Mapping of Physical Scene Properties with an Autonomous Robot Experimenter**. In *Proceedings of the Conference on Robot Learning (CoRL))*. [Haughton et al., 2022].

While not described directly, the following publications were done in conjunction with this thesis:

- Wada, K., Sucar, E., James, S., Lenton, D., Davison, A. (2020) **MoreFusion: Multi-object Reasoning for 6D Pose Estimation from Volumetric Fusion**. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [Wada et al., 2020].

- Ortiz, J., Evans, T., Sucar, E., Davison, A. (2022) **Incremental Abstraction in Distributed Probabilistic SLAM Graphs**. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Ortiz et al., 2022].

- Matsuki, H., Sucar, E., Laidlow, T., Wada, K., Scona, R., Davison, A. (2023) **iMODE: Real-time Incremental Monocular Dense Mapping using Neural Field**. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Matsuki et al., 2023].

- Mazur, K., Sucar, E., Davison, A. (2023) **Feature-realistic neural fusion for real-time, open set scene understanding**. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [Mazur et al., 2023].

The following video material provides a visualisation of the algorithms developed in this thesis:

- NodeSLAM: Neural Objects Descriptors for Multi-View Shape Reconstruction, https://youtu.be/zPzMtXU-0JE.
- iMAP: Implicit Mapping and Positioning in Real-Time, https://youtu.be/c-zkKGArl5Y.
- iLabel: Revelaing Objects in Neural Fields, https://youtu.be/bL7RZaMhRbk.

Chapter 2

# Literature Review

## Contents



In the thesis we build around different concepts of sparse and dense real-time SLAM, compressive representations, and semantic mapping. In this section we will provide an overview of representative work introducing these concepts, discussing their characteristics and open challenges.

The problem of recovering the 3D scene structure along with camera poses from a collection of 2D images has been traditionally explored in the *structure from motion*

research area in the computer vision community, normally performed in an offline and batch setting. Visual SLAM has focused on the problem of online and real-time performance by combining techniques from structure from motion and Bayesian Filtering [Kalman, 1960]. There has normally been a separation in the estimation algorithms used for structure from motion and Visual SLAM techniques depending on the choice of representation, between *sparse* and *dense* systems. The methods in this thesis will borrow design elements from both paradigms

## 2.1 Sparse Reconstruction

In sparse reconstruction systems scene geometry is abstracted as a collection of 3D points, that lie in the surface of objects and is normally in the order of 1000's of points for small scale scenes (room or indoor spaces). A first process in the reconstruction pipeline, the *front end*, consists of extracting and matching a set of 2D salient points in the images. Salient points, called *keypoints* or *features*, are selected with the goal of maximising the ability to recognise them across different viewpoints. Image *corner points* have this property in contrast with regions of high ambiguity such as uniform regions or edges. [Harris and Stephens, 1988, Shi and Tomasi, 1994, Moravec, 1977] presented pioneering algorithms for corner detection based on patch auto-correlation.

The next part of a visual SLAM pipeline, feature matching is the process of identifying keypoint correspondences across different images. This is done through comparison of feature *descriptors*. Descriptors are vector representations of the local image appearance of a given feature. Early SLAM systems mostly relied on patch similiarity. A pioneering algorithm for designing improved feature descriptors was SIFT, presented in [Lowe, 1999]. The method is based on histogram of image gradients, and is designed to be scale and rotation invariant. A big emphasis was placed on making keypoint detection and description more efficient, which led to a variety of methods [Rublee et al., 2011, Calonder et al., 2010, Leutenegger et al., 2011, Bay et al., 2008, Alcantarilla et al., 2012].

When a 3D point is projected to a 2D image position, its location depends on three variables, the camera *intrinsic geometry parameters*, the *camera motion*, and the *3D position* of the corresponding landmarks. Initial work on recovering camera motion from pair of images relied on solving linear systems derived from epipolar geometry constraints, such as the 8 point algorithm [Hartley, 1995] or extensions to multi-camera setups such as the trifocal tensor methods [Hartley, 1994, Armstrong et al., 1996]. The position of the 3D landmarks is then recovered from triangulation. These methods suffer from degenerate solutions and stability issues, in particular with small baselines or noisy inputs. More relevant to us are nonlinear approaches solved through iterative optimisation. In Horn [Horn, 1986] the fundamentals for two-frame nonlinear structure from motion were established, where 3D landmark position as well as relative camera orientation and translation are solved for through nonlinear least squares optimisation. Optimisation is performed using the 2D associations with a camera projection measurement function, which we will define in Section 3. This approach can be directly extended to multiple images with batch optimisation over the full set of measurements, process referred to as *Bundle Adjustment* [Szeliski and Kang, 1993, Kumar et al., 1989, Vidal et al., 2001]. See Figure 2.1 for an illustration of a sparse structure from motion pipeline.

Later methods scaled up inference techniques from structure for motion to handle sequential and real-time operation, giving rise to *visual odometry* and *visual SLAM*. There are two categories for doing this which we will describe next, *filtering* methods and *local bundle adjustment*.

### 2.1.1 Filtering approaches

Filtering approaches initially developed for estimating the position of a ground robot moving in a plane. [Smith and Cheeseman, 1986] introduced the first probabilistic formulation for SLAM. The states of the robot position and 3D landmarks are modeled as a joint Gaussian distribution, which accounts for the correlations between map and position imposed by re-observation across time. Incremental state estimation is done using the *Extended Kalman Filter EKF* taking into account meas-

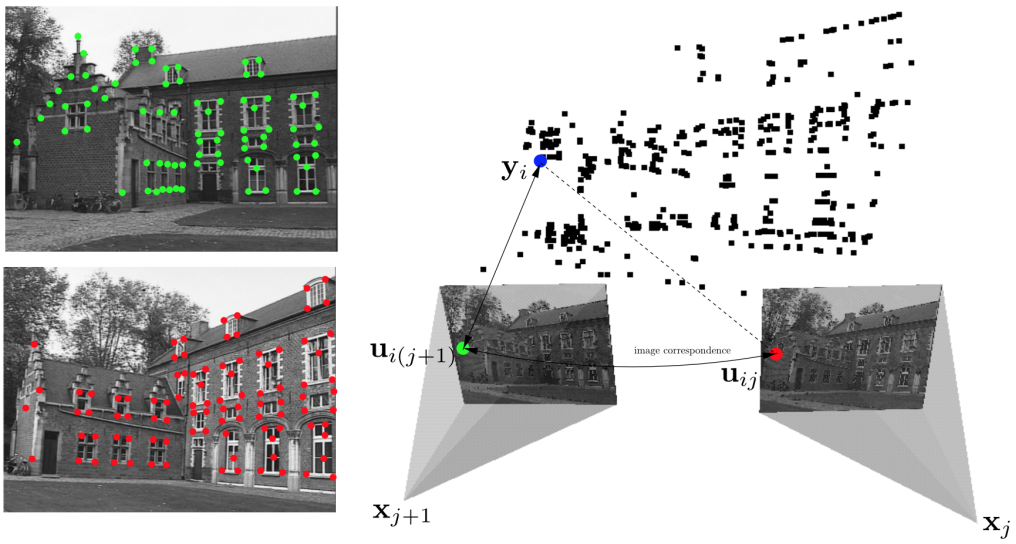



Figure 2.1: Illustration of sparse structure from motion pipeline: image features $\mathbf{u}$ are first detected and matched on the image pair, then camera poses $\mathbf{x}$ are jointly estimated with 3D points $\mathbf{y}$ via non-linear optimisation to minimise the distances between the projection of the 3D points and the detected ones. (Image adapted from [Pollefeys et al., 1999].)

urement uncertainty. This formulation became standard for robot SLAM, and was later developed in different robot applications [Leonard and Whyte, 1991, Castellanos, 1998, Davison, 1998, Newman, 1999].

Robot SLAM methods were constrained to 2D maps and used robot wheel odometry for motion estimation and range measurements for landmarks. [Harris and Pike, 1987] was the first approach to tackle the general 3D problem with a free moving camera. They presented a system capable of tracking the camera position incrementally with respect to an estimated 3D point cloud. The system performed iterative inference using a Kalman filter with the associated Harris features, however they assumed independent states and did not model correlations. [Broida et al., 1990] later proposed a system which modeled the full state covariance, and included an initial Bundle Adjustment step for initialisation, however they assumed all the feature correspondences were given. [Chiuso et al., 2002] presented a system where inserting new features was taking into account, however this was done is a sep-

arate filter and limited application to a compact space of 20-40 landmarks, and did not account for re-observing landmarks. [Davison, 2003] pioneered the first full, automatic monocular visual SLAM system, MonoSLAM. MonoSLAM used the joint uncertainty for efficient feature tracking, which allowed re-detection of unobserved features. It also developed an automatic top-down feature initialisation procedure by modeling a depth distribution, and a map management strategy for discarding spurious landmarks and keeping a reasonable sized state. MonoSLAM showed unprecedented performance with respect to localisation with fast camera motion, and the ability to not accumulate drift under continious operation in a room scale scene.

### 2.1.2 Bundle adjustment

The limitation of filtering based SLAM are the errors introduced by accumulating linearisation of non linear measurements, and the quadratic continuous time and space complexity with respect to state size, which limits the number of landmarks used. To counter these limitations a new evolution of SLAM methods relying on local bundle adjustment emerged. [Nistér et al., 2004] proposed a *visual odometry* system for guiding a robotic outdoor car. Motion was solved for a group of frames, in a *sliding window* of 3 frames, by the 5-point algorithm [Nistér, 2004] with Random Sample Consensus RANSAC [Fischler and Bolles, 1981] followed by bundle adjustment using tracked 2D features, from which 3D feature positions are then triangulated. However this system does not take into account landmark re-observation, leading to drift in motion. [Engels et al., 2006] further expands this framework by performing local bundle adjustment with the Levenberg-Marquardt optimiser [Levenberg, 1944] for a window of 20 frames, achieving real-time performance by efficient optimisation leveraging the sparsity structure in the factor graph. To avoid the windowed optimisation to diverge from past past measurements, the gauge is fixed by fixing the first camera pose in the local window. They show improved stability over long term sequences preventing gross failure modes.

[Mouragnon et al., 2006] concurrently presented a systems scaling to 1000s of 3D points for city scale trajectories and introducing the concept of *keyframes*. Key-

frames are selected subset of frames used for efficiency local bundle adjustment, chosen when few matches are obtained or when pose uncertainty is high. Finally Parallel Tracking and Mapping (PTAM) [Klein and Murray, 2007] consolidated local bundle adjustment developments and added new innovations to form the basis of modern monocular sparse SLAM systems. First they introduced the concept of parallel tracking and mapping, where they separate camera tracking given the map, and map building with bundle adjustment into separate threads. Tracking runs at a higher frame rate which is important for most applications, while map building at slower rate, allowing more accurate and bigger maps. Second they associate 3D landmark descriptors to the keyframe image where they were initialised, allowing reobservation of features and simplified feature tracking. PTAM demonstrated that keyframe bundle adjustement presents much higher accuracy and robustness than filter based methods.

[Strasdat et al., 2012] presents further analysis comparing both techniques, highlighting that the benefits of local bundle adjustment come from the ability to use many more map points by leveraging factor graph sparsity, in contrast to filter methods that introduce variable dependencies through marginalisation. However, filtering systems still remain used in the visual odometry systems, specifically when combined with inertial measurement units (IMU) as in [Mourikis and Roumeliotis, 2007].

## 2.2 Dense Reconstruction

The sparse methods presented were designed with the main objective of precise and efficient camera localisation, and thus model the 3D scene with a sparse point cloud. In applications such as robotics a much richer and detailed geometric reconstruction is necessary. This led to a new category of methods for *Dense Reconstruction* and SLAM. In this section we will give an overview of the 3D scene representations and estimation methods developed for dense reconstruction.

### 2.2.1 Multi-View Stereo

The first developments in dense visual reconstruction came about by tackling the *stereo correspondence problem*; an overview is given in [Scharstein and Szeliski, 2001]. This category of methods aim at establishing a dense correspondence between a pair of calibrated images. In contrast to sparse methods, they aim to establish correspondence in a dense manner, introducing the concept of a *disparity image* (analogous to inverse depth image) [Okutomi and Kanade, 1993]. The disparity image defines for every pixel in a reference frame the vector to the pixel corresponding to the same 3D point in the other image, across the epipolar line. A big family of algorithms exist for stereo correspondence, but rely on the two main concepts of *photo-metric matching cost* and *regularisation*. The photo-metric cost measures the appearance agreement between corresponding pixels, and is normally aggregated across a small local window, such as with Sum of Squared Differences *SSD* [Anandan, 1989]. Because of the under-constrained solution space, especially in smoothly textured areas, a regularisation penalty to encourage local smoothness is added [Yang et al., 1993]. The disparity image is solved for to maxisimise the photo-metric cost given the regularisation. A wide variety of optmisation algorithms exist for solving two view disparity optimisation, including quantised local optmisation such as plane-sweep [Collins, 1996], combinatorial global optimisation as graph-cut [Kolmogorov and Zabih, 2001], dynamic programming [Cox et al., 1996], and cooperative algorithms [Zitnick and Kanade, 2000].

### 2.2.2 Dense Monocular SLAM

[Pollefeys et al., 1999] introduced one of the first full automatic dense reconstruction systems, using sparse local bundle adjustment for camera pose estimation, followed by dense multi-view stereo matching with dynamic programming, then fusing the depth maps into a volumetric map [Curless and Levoy, 1996] and extracting a a full geometric mesh with marching cubes [Lorensen and Cline, 1987]. [Pollefeys et al., 2008] extended the system for real-time incremental reconstruction using GPU hardware acceleration. DTAM [Newcombe et al., 2011b] pioneered dense SLAM, showing

unprecedented real-time reconstruction with a monocular video stream, by combining primal-dual non convex optimisation [Chambolle and Pock, 2011] with cost volumes using GPU hardware acceleration [Hosni et al., 2013]. Furthermore camera tracking was done within the dense representation using Lukas-Kanade non-linear least squares optimisation [Lucas and Kanade, 1981].

In this thesis we will not directly tackle the multi-view stereo problem, since the development of *depth camera* commodity sensors provides accurate dense depth image measurements for indoor environments by using active infrared projection, with the first commercial product introduced by Kinect [Microsoft Corp, 2010].

### 2.2.3 Depth Fusion

Given a set of depth image measurements, the next problem in dense reconstruction is measurement fusion, that is how to combine the underlying depth measurements into a coherent 3D map. This is to build a persistent scene model, avoiding geometric redundancy of depth measurements averaging out measurement noise. We will next describe different 3D representations with their corresponding estimation algorithms for dense fusion. A common representation is to discretise space into a uniformly spaced grid. This technique was originally developed for building occupancy grid maps for robotic localisation and navigation from range measurements coming from sonar, ultra-sound or stereo camera sensors [Moravec and Elfes, 1985, Thrun, 1998, Buhmann et al., 1995]. Originally these methods were developed for 2D grid maps representing horizontal slices parallel to the ground plane, but were later generalised to full 3D voxel grids [Martin and Moravec, 1996]. An occupancy grid map stores for each cell the probability of it being occupied by an object, and these are estimated with inverse sensor models, back projecting range measurements into the grid and solving a an estimation problem for each cell independently.

3D voxel grid maps were then extended to store signed distance, which is more informative than occupancy as it represents the distance to the closest surface, and allow for recovering if desired more precise explicit surface geometry. [Curless and

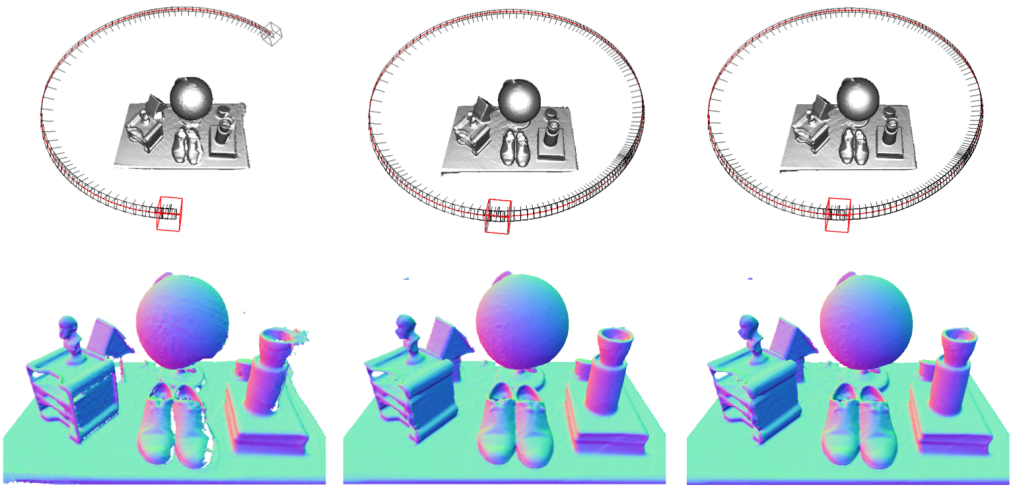


Figure 2.2: Normals render of dense reconstruction from KinectFusion obtained by fusing depth image measurements into a signed distance function voxel grid. (Image adapted from [Newcombe et al., 2011a].)

Levoy, 1996] presented an incremental algorithm for incremental fusion of range measurements into a *Signed Distance Function (SDF)* grid. The method works by using the range measurement to approximates the signed distance function near the surface, under a truncation distance, and accumulating or fusing measurements through a weighted average for each voxel, where the weight takes into account geometric properties to reflect how well the range measurement approximated the SDF. KinectFusion [Newcombe et al., 2011a], see Figure 2.2, builds on top of this technique a full Dense SLAM system than runs in real-time on a desktop computer with parallel GPU implementation and using a Kinect depth sensor camera. Through ray-casting [Parker et al., 1998] of the SDF volume grid, they can *render* or predict a depth image from a given camera pose. Given the rendered image, camera tracking is performed in an alternating manner with map fusion, via point-plane Iterative Closest Point ICP [Chen and Medioni, 1992] with projective data association. By doing model to frame camera tracking KinectFusion displays less drift than frame to frame depth tracking such as in [Rusinkiewicz et al., 2002].

Dense SLAM methods with voxel grids cannot scale to bigger spaces because of cubic memory and computation complexity. To tackle this, methods leveraging data

structures for efficient spatial decomposition, such as hierarchical octrees [Wurm et al., 2010, Vespa et al., 2018] or hash-tables [Nießner et al., 2013a, Kahler et al., 2015], expanded on voxel-based dense SLAM. Another category of Dense SLAM methods directly model surface geometry as a map representation, in contrast to occupancy or SDF which are *implicit surface* representations. In ElasticFusion [Whelan et al., 2015] the map is represented as a dense collection of *surfels* (similar to a point cloud), which are small oriented discs. ElasticFusion adds a deformation graph on top of the surfels, which allows non-rigid deformation to correct for local loop closures. BAD SLAM [Schops et al., 2019] extends this, and allows an alternating bundle adjustment optimisation of surfels and camera poses, by limiting surfels to move along the normal direction.

The algorithmic simplicity with natural parallelisation, and reconstruction accuracy has made voxel fusion based dense SLAM systems dominant in the robotics community. However, several of the assumptions made impose important limitations as outlined in the future work section of [Newcombe, 2012]. First, tracking and mapping are separated into independent alternating optimisation procedures. This is because the full dense probabilistic SLAM problem considering the joint distribution of camera trajectory and map is not modeled. Instead the problem is split into independent estimation problems for each cell and each frame. This causes drift and map error accumulation, as the solution with this method is only optimal when each tracking and mapping step also yields an optimal state. More over, the assumption that each voxel is independent disregards spatial relationships between them and leads to maps with holes.

### 2.2.4 Generative dense reconstruction

A generative formulation for dense reconstruction requires the definition of a *forward sensor model*, for computing the the probability of the observation (depth or colour images) given the full map model. One setting where generative approaches have been developed is surface reconstruction. In [Cheeseman et al., 1996, Morris et al., 2001] the surface is modeled using a discrete uniform 2D grid storing elevation

and surface emitance. A likelihood function is defined for intensity images given surface parameters, which is solved for using a Bayesian approach which also models smoothness constraints between adjacent cells. [Zienkiewicz et al., 2016] further developed this method into an incremental and real-time formulation also integrating camera tracking, by instead using depth image measurement images obtained from stereo. OpenDR [Loper and Black, 2014] provided a suite for differential rendering of arbitrary meshes by approximating derivatives of the standard graphics rendering pipeline, which can serve as a generative model for solving computer vision problems, a process referred to as *inverse graphics*.

For implicit reconstruction, [Thrun, 2003] presents an approach for 2D occupancy mapping using a sonar sensor with forward sensor models, by defining a Gaussian mixture model over possible distances along each each measurement cone. The likelihood is then optimised using an alternating Expectation Maximisation EM approach. [Liu and Cooper, 2010, Liu and Cooper, 2011, Liu and Cooper, 2014] present a Bayesian formulation for 3D reconstruction through inverse ray-tracing. The map is represented by a grid of voxels which store opacity and color. For MAP estimation of the voxel parameters a ray Markov Fields is formulated, in which factors connect a pixel color to the voxels along the back projected ray, and are derived from the rendering equation [Kajiya, 1986]. Additional factors between voxels are added to impose smoothness constraints. Inference is done with an efficient implementation of loopy belief propagation, and this runs in an offline batch setting.

One challenge with generative approaches for volumetric implicit 3D reconstruction is the expensive memory and computation cost of a full voxel grid. Hierarchical representations such as octrees can be difficult to employ because the pruning operations are not differentiable. An alternative representation which has been explored for representing implicit volumes is the use of a continuous function, without any explicit discretisation. One example combining hierarchical space decomposition with a basis of continuous functions is Poisson Surface Reconstruction [Bolitho et al., 2009]. This method represents space through a linear combination of cubic b-splines at different hierarchical levels. The coefficients are solved for given a point

cloud with normals by solving the Poission equations they define using non-linear optimisation. By modeling spatial continuity the method can fill in surface holes in the data. Hilbert Maps [Ramos and Ott, 2016] are a method for 2D occupancy mapping without any discretisation of space by modeling reconstruction as logistic regression of occupancy on point coordinates. The regression is learned on a semi-dense point cloud, where the datastet is such that the 2D coordinates are the input to the classifier and binary occupancy is the output. The method demonstrates that the formulation allows better generalisation in areas with no measurments and robustness to outliers. To be able to model complex scenes regression is performed on Fourier features of the 2D coordinates, approximating Kernel regression. More recently and relevant to iMAP, Chapter 5, NeRF [Mildenhall et al., 2020a] uses an MLP (which maps coordinates to volume density and color) as a continious 3D scene representation for novel view synthesis. The MLP is supervised through a dataset of RGB images with known camera poses through a differential formulation of the volume rendering equation, achieving unprecedented novel view synthesis quality.

In **iMAP**, Chapter 5, we will present a full generative dense SLAM formulation for room-scale reconstruction by combining a continuous 3D map represented with an MLP with a probabilistic differential volume renderer of depth and color. We present a system which runs in incrementally real-time on a GPU from a stream of depth images using the following design components: keyframe selection, parallel tracking and mapping, and active sparse optimisation. Our method enables joint graph optimisation, automatic hole filling and map compression with dynamic level of detail.

## 2.3 Semantics

To move beyond pure geometric understanding of a scene, a useful representation must contain information about abstractions, such as a decomposition into objects or information about *semantics*. Furthermore the ability to abstract a dense map into object models offers opportunities for efficient scene representation, to handle

moving objects, and for providing strong priors on reconstruction with incomplete data.

Finding what and where are the objects in an image has been widely studied in the computer vision community. *Object detection* aims to estimate the region within the image where an object is, and *pose estimation* is finding the transformation of a 3D object with respect to the camera. Initial approaches for object detection focused on detecting specific objects through template matching [Fischler and Elschlager, 1973, Huttenlocher et al., 1993]. The basic idea is to collect a set of patch templates from an object and use this library to detect the object in the new image by using some patch correlation distance. These methods have been extended to be more robust to changes in illumination through the use of depth and edge cues, such as LINEMOD [Hinterstoisser et al., 2012], as well as to be robust to different viewpoints by using 3D CAD models to collect more templates [Hinterstoisser et al., 2013]. However these methods are still not robust in cluttered settings and with very different viewing conditions. The development of local image descriptors such as SIFT [Lowe, 1999] designed to be invariant to viewpoint and lighting led to more robust and efficient object matching algorithms, such as the system presented in MOPED [Collet et al., 2011].

In order to go beyond detecting specific object instances to detect objects within a semantic category such as *chair*, statistical machine learning methods are used. These methods consist of a first stage of image feature extraction such as Bag of Words [Csurka et al., 2004] which accumulates local descriptors into a histogram or convolving a bank of filters such as Haar wavelets [Viola and Jones, 2001], followed by applying a classifier such as SVM [Dalal and Triggs, 2005], a random forest [Bosch et al., 2007], or boosting [Torralba et al., 2004]. The classifier is learned using a dataset of labeled training images. Object detection methods normally provide a coarse estimate, such as a rectangular bounding box, of the object's position in the image. The problem of obtaining a precise pixelwise separation of the object is referred to as *image segmentation*. [Shi and Malik, 2000] formulated segmentation as a graph partitioning problem, where pixels are nodes and edge strength is determined

intensity and location similarity. A probabilistic formulation for segmentation was proposed with the use of Conditional Random Fields (CRF) [Lafferty et al., 2001], for example objcut [Kumar et al., 2005] proposes a method for object detection and segmentation by combining shape templates with CRFs.

The advent of *deep learning* has provided a unified framework for object detection, image segmentation, and object pose estimation which has dominated recent years because of its simplicity and improved accuracy. The basic backbone for modern methods is the Convolutional Neural Network architecture which was introduced by Yann LeCUn in LeNet5 [LeCun et al., 1998]. Inspired by hand crafted filter banks, CNN stack layers of optimisable or *learnable* convolutional filters followed by non linear activations with a classifier at the end. All the parameters are trained in an *end to end manner* by *supervised learning* in a dataset with images and ground truth labels. 2012 was the year when the domination of deep learning started with AlexNet [Krizhevsky et al., 2012] which outperformed all other methods in the challenging large scale ImageNet [Krizhevsky et al., 2012] classification challenge by leveraging GPU parallelisation of CNNs. VGG-net [Simonyan and Zisserman, 2015] showed that the use of smaller 3x3 filters allowed for more layers to be stacked and get better performance. This network architecture is the backbone used for pose prediction used in the NodeSLAM system, Chapter 4, presented in this thesis. This was taken even further in ResNet [He et al., 2016] by stacking 157 layers, and forms the backbone of many image recognition modules. CNNs were adapted beyond classification for object detection in works such as R-CNN [Girshick et al., 2014], and Faster-RCNN [Ren et al., 2015] by using regions of interest, and for instance segmentation in Mask-RCNN [He et al., 2017]. Mask-RCNN is used for the object detection and segmentation modules we use in NodeSLAM Chapter 4.

### 2.3.1 Geometry and semantics

For going beyond images to performing object detection or semantic segmentation in 3D maps, a process referred to as *scene labeling*, earlier approaches relied using geometric cues from an already reconstructed map to group elements. In [Nüchter

et al., 2003] planar regions are extracted from a point cloud using Random Sample Consensus RANSAC, and classified into *floor*, *wall*, or *celling* based on hand designed relative orientation and position constraints. In [Mozos et al., 2007] an Adaboost classifier is trained to separate a 2D occupancy map into semantic classes from geometric cues. [Brostow et al., 2008] leveraged a dense point cloud obtained from SfM to project geometric cues into the image and then combine with appearance for segmentation with a random forest. [Koppula et al., 2011] perform segmentation on a 3D point cloud scan obtained from an RGB-D sensor by aggregating appearance and geometric features into the 3D map, and performing classification with a Markov Random Field with log-linear nodes and edge potentials, trained on a dataset of 50 scenes. In SemanticPaint [Valentin et al., 2015], rather than pre-training on a labeled dataset, training is done interactively in for a specific scene. For this KinectFusion is run for real-time voxel-based reconstruction, then a user interactively annotates sparse labels indicating objects in the voxel map. The user labels are propagated through a CRF and fed to a random forest classifier which is trained online with geometric features concurrent to the reconstruction and labelling. The method of SemanticPaint inspired our work of iLabel in Chapter 6.

With the popularity and accuracy of image based recognition due to easier data availability, another category of methods for scene labeling first performs image segmentation and then fuses the labels into a 3D map. These methods accumulate 2D labels into a 3D structure with Bayesian updates and then regularise the labels in 3D using a Conditional Random Field (CRF). [Stückler et al., 2015, Hermans et al., 2014] both use a Random Decision Forest for 2D classification and point cloud or surfel 3D structures for label fusion. [Kundu et al., 2014] accumulates 2D labels into an octree representation which scales to bigger scenes. [Cavallari and Di Stefano, 2016] and Semantic Fusion [McCormac et al., 2017] leverage improved CNN 2D classifiers for semantic fusion jointly with dense real-time SLAM systems KinectFusion and SemanticFusion respectively.

In **iLabel**, Chapter 6, we build on top of iMAP to test the hypothesis that automatic compression of a scene should discover underlying structure in the form of

object abstractions. For this, we extend the scene representation MLP of iMAP to produce semantic outputs, thereby modeling geometry, color, and semantics with a unified generative model. The semantic outputs are then supervised interactively through sparse pixel anchors provided by the user. We show that the MLP automatically propagates the anchors into full object instance segmentations. We find that the propagation is correlated to local geometry and color, respecting object boundaries, and global color appearance, providing evidence that scene compression is enabled by scene decomposition and grouping. iLabel serves as a practical system for efficient open-set scene labeling.

### 2.3.2 Object based reconstruction

For efficient 3D scene reasoning a category of methods aim to abstract a 3D map through the detection of object templates from a pre-built dataset. [Castle et al., 2007] was one of the first sustems to integrate object templates into a SLAM pipeline. They detect the position of known planar objects such as posters through SIFT descriptor matching and insert them into the MonoSLAM system. They show that constraints the planar objects impose on 3D features improves tracking robustness and allows for scale estimation within the monocular setup. [Civera et al., 2011] goes beyond planar objects and inserts pre-scanned dense objects point-clouds detected with SURF descriptors into MonoSLAM. [Kim et al., 2012] leverages a dense scan to replace common office objects such as chairs, desks, and monitors with templates of these objects constructed using basic cubic primitives. The detection components of this method are geometric using point cloud clustering principal axis detection.

SLAM++ [Salas-Moreno et al., 2013] shown in Figure 2.3, pioneered object based SLAM, where objects, templates serve as the landmarks of a real-time SLAM system. As a first pre-processing stage a high quality dataset of objects meshes is constructed by KinectFusion scanning. Then a new scene is reconstructed under the assumption that it contains enough instances of some of the object templates. During live operation from a depth camera input stream, the 6-DoF pose of the objects is estimated by Hough forest voting with geometric Point Pair Features, and

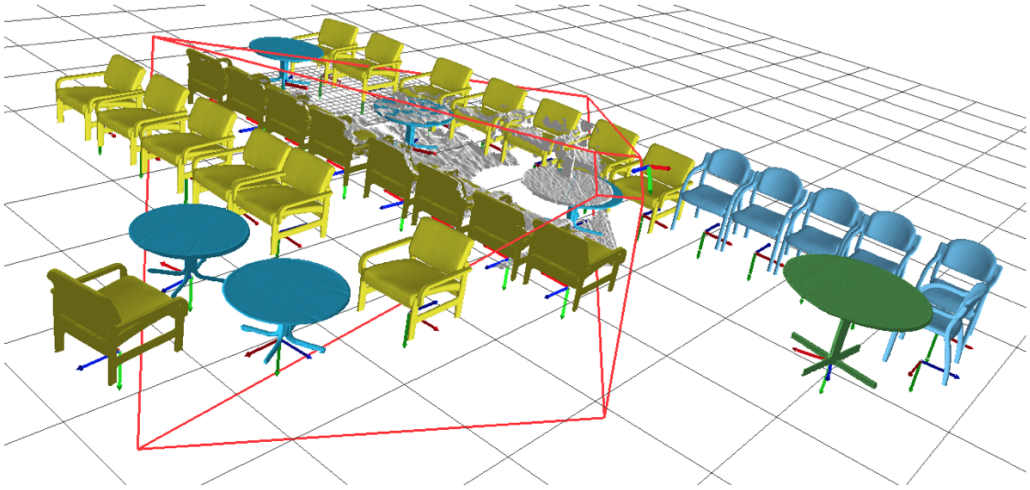


Figure 2.3: Example from SLAM++ where the 3D map is abstracted into a sparse graph of objects. The inserted objects are rigid templates from a pre-scanned dataset. (Image adapted from [Salas-Moreno et al., 2013].)

these are inserted into the map model. The camera position is tracked with ICP with respect to the map of object meshes. The full optimisation problem can be abstracted into a sparse graph with the objects and history of camera poses as variables, and ICP constraints between them, which allows a joint graph optimisation reminiscent of sparse SLAM bundle adjustment. The general pipeline of NodeSLAM in Chapter 4 is inspired by SLAM++, with object detection and joint graph optimisation components. In follow up work Fusion++ [McCormac et al., 2018], rather than use a pre-scanned template library a generic image object detector is used and SDF fusion is done in a per-object voxelgrid. The objects are then used in a joint graph optimisation similar to SLAM++.

### 2.3.3 Semantic shape priors

In order to go beyond a fixed shape template a category of methods increase flexibility in shape variations within objects of a given class (such as cars) through parametric object models. Initial parametric shape priors developed as a top down approach to guide 2D segmentation and make it more robust to noise in local statistics. In [Cremers et al., 2001] an approach is proposed where segmentation is

modeled through a spline curve with control points, referred to as *snake-based segmentation*. A prior distribution over the control points is learned from a dataset of *hand* images. Then segmentation is formulated as a variational energy minimisation problem, with a term encouraging the spline to be within the training distribution and another term measuring how well the spline segments the image, such as smoothness within the regions and following image gradients. [Dambreville et al., 2008b] extends this formulation by modeling the segmentation curve implicitly through a 2D SDF. *Principal Component Analysis PCA* is then performed on a set of aligned 2D shapes to obtain a low dimensional latent space. By modeling the contour implicitly this method has more robust optimisation and can handle topological changes. More expressive latent spaces were developed by introducing non-linearities with Kernel-PCA [Dambreville et al., 2008a] or probabilistic latent spaces such as GP-LVM [Prisacariu and Reid, 2011]. The overall category of methods where a 2D boundary is obtained by the projection of a higher level function is referred to as *level-set* based segmentation.

The later evolution of these methods obtains the 2D contour as a projection of a 3D model, thus modeling shape variation due to viewpoint explicitly. In [Tsai et al., 2003] the same variational energy approach is used for segmentation, but the contour is modeled through the projection of the 3D surface from a known model, then the energy is minimised with respect to the pose parameters of the 3D shape. To allow variation in shape, in [Yingze Bao et al., 2013] the 3D surface has a mean shape plus anchor points associated with it, which allow smooth deformations through a thin plate splice formulation. This method then uses the shape prior to densify and complete a multi-view stereo sparse reconstruction by matching the anchor points.

PWP3D [Prisacariu and Reid, 2012] proposed an energy based formulation for both 2D segmentation and real-time 3D object pose tracking of a known model. In this method a vertex based 3D model is first rendered and then converted into a 2D implicit distance transform. [Prisacariu et al., 2013] adopted an implicit voxel based representation for 3D object shape. A latent space is then learned by GP-LVM using a dataset of aligned 3D car shapes. This representation is then used for recovering

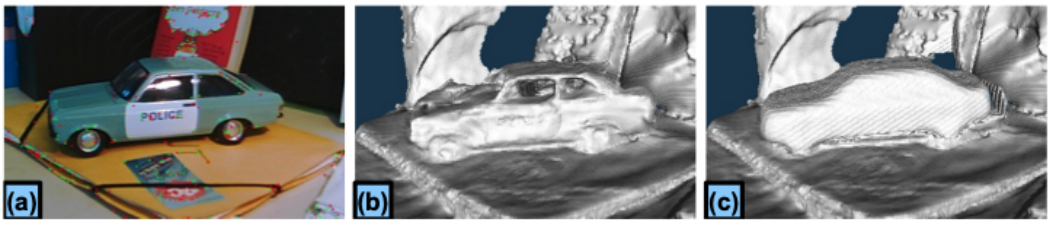



Figure 2.4: Example of using shape prior from car class to complete and regularise a dense reconstruction. (Image adapted from [Dame et al., 2013].)

3D shape and pose, and 2D segmentation from one or more images by a level set energy minimsation from the rendered contours. [Dame et al., 2013], see Figure 2.4, extended this by combining the shape priors into a monocular SLAM systems. The energy formulation for the object shape is extended to account for the estimated surface from the SLAM system. [Engelmann et al., 2016] applied similar techniques to a larger scale setting, reconstructing the shape of cars in a city scale dataset.

Neural networks were used to develop more expressive compressive generative models than linear models such as PCA. Initially these methods were developed for compression of images such as in Variational Auto-Encoders (VAEs) which learned probabilistic compact latent space through a bottleneck CNN network or a Generative Adversarial Network (GAN) which learned to generate images by competing generative and discriminative networks. These generative methods allow sampling new data that resembles the training distribution. With the development of datasets for 3D shapes such as ShapeNet [Chang et al., 2015], these methods were extend to 3D by using 3D CNNs [Wu et al., 2016] or coordinate-based MLPs as in [Park et al., 2019].

In **NodeSLAM**, Chapter 4, we present an object-level SLAM system which combines class-level parametric objects models within a sparse graph optimisation pipeline as in SLAM++. The parametric models are learned on occupancy 3D grids by a VAE neural network trained on a dataset of object shapes, and at inference time we leverage a CNN detector. The ability to go beyond the fixed templates of SLAM++ allows us to model variation within a semantic class such as *mugs*.

We show our system produces accurate and complete object reconstructions from partial observations, and demonstrate this in a robotic object packing application.

Chapter 3

# Technical Preliminaries

## Contents

In this section we will describe the technical components that will form the basis for the systems we develop throughout the thesis, encompassed by the use of *neural representation* for *dense semantic SLAM*. In Sections 3.2 and 3.3 we present the basics of 3D geometry for visual SLAM, describing the camera model and rigid transformations respectively. In Section 3.4 we describe the probabilistic language of SLAM, factor graphs, and in Section 3.5 how to do inference in them through non-linear optimisation. In Section 3.6 we talk about neural networks which we will use both as building blocks for both discriminative inference and generative representations. Finally in Section 3.9 we describe differential volumetric rendering which will serve as the measurement function we use in generative dense SLAM.

## 3.1 Notation

This thesis makes use of the following notation:

### 3.1.1 General Notation

| | |
|---|---|
| $a$ | This font is used for scalars. |

**a** This font is used for $M$-dimensional column vectors, where $a_i$ is the $i^{\text{th}}$ element of the vector:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix}, \quad \mathbf{a}^T = \begin{bmatrix} a_1 & a_2 & \dots & a_M \end{bmatrix}. \tag{3.1}$$

**A** This font is for $M \times N$-dimensional matrices, where $a_{ij}$ is the matrix element at the $i^{\text{th}}$ row and $j^{\text{th}}$ column:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \dots & a_{MN} \end{bmatrix}. \tag{3.2}$$

**I** This represents the identity matrix.

**0** This represents the zero matrix.

### 3.1.2 Probability

$p(\mathbf{x})$ This represents the probability density of $\mathbf{x}$.

$p(\mathbf{x}|\mathbf{y})$ This represents the probability density of $\mathbf{x}$ given $\mathbf{y}$.

### 3.1.3 Spaces and Manifolds

$\mathbb{R}$ This denotes the set of real numbers.

$\mathbb{R}^M$ This denotes the vector space of real $M$-dimensional vectors.

$\mathbb{R}^{M \times N}$ This denotes the vector space of real $M \times N$-dimensional matrices.

$SO(3)$ This denotes the 3D rotation group.

$SE(3)$ This denotes the Special Euclidean group.

### 3.1.4 Frames and Transformations

$\mathbf{a}_A$ The represents the vector $\mathbf{a}$ expressed in Frame A in $\mathbb{R}^3$.

$\mathbf{R}_{AB}$ This represents a 3D rotation expressed as a rotation matrix (i.e. $\mathbf{R}_{AB} \in SO(3)$).

$\mathbf{T}_{AB}$ This represents the homogeneous transformation matrix that transforms homogeneous points from frame B in $\mathbb{R}^3$ to frame A in $\mathbb{R}^3$.

### 3.1.5 Camera Models and Images

$f_x$ This represents the horizontal focal length of the camera, in pixels.

$f_y$ This represents the vertical focal length of the camera, in pixels.

$c_x$ This represents the horizontal coordinate of the camera centre, in pixels.

$c_y$ This represents the vertical coordinate of the camera centre, in pixels.

$\mathbf{K}$ This represents the intrinsic camera matrix:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.3}$$

$\boldsymbol{\pi}(\cdot)$ This denotes the perspective projection function:

$$\boldsymbol{\pi}(\mathbf{a}) = \boldsymbol{\pi}\left(\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}\right) = \frac{1}{a_3}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}. \tag{3.4}$$

$I(\mathbf{u})$ This represents the intensity at pixel coordinate $\mathbf{u}$.

$D(\mathbf{u})$ This represents the depth value corresponding to the pixel coordinate $\mathbf{u}$.

## 3.2 Camera Model

We use a *pinhole camera* to model the relation between points in the 3D world and 2D points in the image plane (modeling the captured image of a CCD type

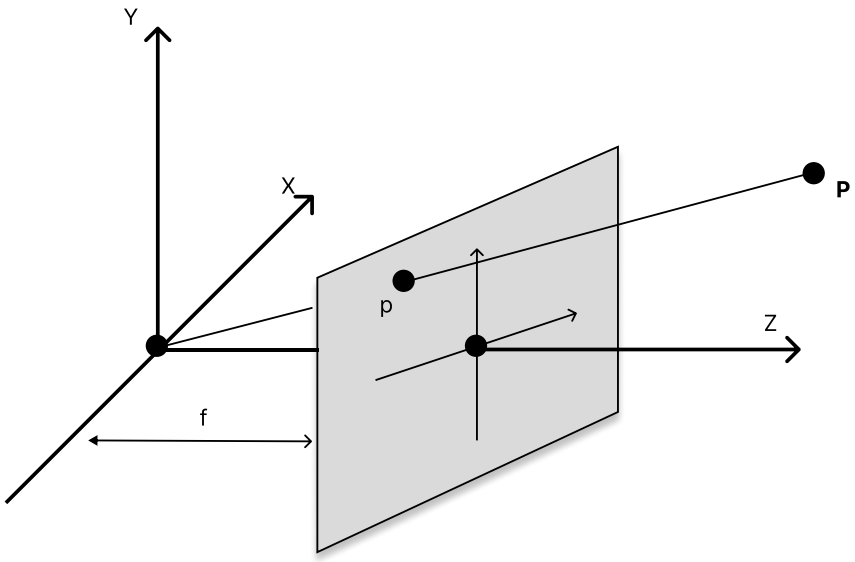



Figure 3.1: Pinhole camera model

sensor). This model will form a core component of the methods developed for 3D reconstruction: going from images to a 3D map.

**Projection:** We will derive the equations for projecting a point in a 3D Euclidean space into a 2D point in the image plane. Consider a 3D point **P** with coordinates $[x, y, z]^T$ and an *image plane* defined by $Z = f$, with $f$ defined as the focal distance, see Figure 3.1. The projection is given by the intersection of the image plane and the line joining the *camera centre* (origin of the 3D Euclidean space) and point **P**. By similar triangles we see $[x, y, z]^T$ is mapped into $[f\frac{x}{z}, f\frac{y}{z}, f]^T$, so in the image plane coordinate system $\mathbf{p} = [f\frac{x}{z}, f\frac{y}{z}]^T$. We define the line perpendicular to the image plane passing through the *camera centre* as the *principal axis* and the intersection of image plane and principal axis as the *principal point*,.

**Intrinsic matrix:** It is convenient to express this equation in matrix form as :

$$\mathbf{p} = \pi(K\mathbf{P}), \tag{3.5}$$

whose

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

is the *intrinsic matrix* describing the projection parameters of our camera sensor, and:

$$\pi\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix} \tag{3.7}$$

the projection operator.

Two extensions are needed for the general pinhole camera model. First, we model a displacement, $c_x, c_y$, between the image plane origin and the principal point. Second we model the scaling of the different image axes (due to rectangular pixels) by using different focal components for each axes $f_x, f_y$. This gives the full pinhole model equation:

$$\mathbf{KP} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x x + c_x Z \\ f_y y + c_y Z \\ z \end{bmatrix}, \tag{3.8}$$

and

$$\mathbf{p} = \pi(\mathbf{KP}) = \begin{bmatrix} f_x x + c_x \\ f_y y + c_y \end{bmatrix}. \tag{3.9}$$

**Back projection:** We are often interested in modelling the inverse operation to projection, *back projection*. As projection collapses all the points that lie in a ray into a single point (that is it is not an injective function), it is not invertible. Therefore back projection maps a point in the image plane into a 3D ray, represented by the following vector:

$$\mathbf{K}^{-1}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{u-c_x}{f_x} \\ \frac{v-c_y}{f_y} \\ 1 \end{bmatrix}. \tag{3.10}$$

We define the *depth* of a 3D point as the distance from the point to the plane the camera origin and orthogonal to the principal axis. Given the depth of a 3D point, $d$, and its projection, $[u, v]^T$ , we can recover the point as:

$$\mathbf{P} = d\begin{bmatrix} \frac{u-c_x}{f_x} \\ \frac{v-c_y}{f_y} \\ 1 \end{bmatrix} = \begin{bmatrix} d\frac{x-c_x}{f_x} \\ d\frac{y-c_y}{f_y} \\ d \end{bmatrix}. \tag{3.11}$$

## 3.3 Transformations

In the previous section we described how to model the relation between 3D points and an image, with the camera is at the world origin, which assumes the camera is static. We are interested in modelling a moving camera for which we need to model its position in space. We describe the pose of the camera using a rigid point transformation $\mathbf{T}_{CW} \in \mathbf{SE}(3)$:

$$\mathbf{T}_{CW} = \begin{bmatrix} R_{CW} & \mathbf{t}_W \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{3.12}$$

with $\mathbf{R}_{CW} \in \mathbf{SO}(3)$ and $\mathbf{t}_W \in \mathbb{R}^3$.

In this way, to project a 3D point $\mathbf{P}_W$, we first transform it from the world frame into the camera frame, and then proceed with the pinhole projection:

$$\mathbf{P}_C = \mathbf{R}_{CW}\mathbf{P}_W + \mathbf{t}_W, \tag{3.13}$$

$$\mathbf{p} = \pi(K\mathbf{P}_C). \tag{3.14}$$

**Lie Groups and Tangent Space** As we will see in future chapters, rigid transformations will be unknown variables which we solve for through optimisation techniques. As rotations cannot be modeled as Euclidean vector space, which is an assumption in optimisation, we will describe how to associate members of the Lie group of rotations, $\mathbf{SO}(3)$, into a vector space where we can use optimisation techniques, the *Lie algebra*: $\mathfrak{so}(3)$.

The Lie algebra is defined as the *tangent space* around the identity of the lie group. In the case of $\mathbf{SO}(3)$, its Lie algebra is the vector space spanned by the following basis vectors:

$$\mathbf{G}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.15}$$

We define the the $[]_\times$ operator as the mapping of the basis coefficients $\boldsymbol{\omega} \in \mathbb{R}^3$ into its corresponding element in the Lie algebra, such that:

$$[\boldsymbol{\omega}]_\times := \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{3.16}$$

$\boldsymbol{\omega}$ can be directly interpreted as a compact representation of a 3D rotation, where the direction of the vector gives the axis of rotation and the magnitude the angle of rotation. This also shows that a 3D rotatin has 3 degrees of freedom.

**Exponential and logarithmic maps:** For associating elements of $\mathfrak{so}(3)$ to elements of $\mathbf{SO}(3)$, and vice versa, we use the *exponential map* and *logarithmic map* respectively. The exponential map refers to the matrix exponential map, defined as follows:

$$\exp : \mathfrak{so}(3) \longrightarrow \mathbf{SO}(3) \tag{3.17}$$

$$\exp(\omega_\times) = \sum_{k=0}^{\infty} \frac{1}{k!}\omega_\times^k = I + \omega_\times + \frac{\omega_\times^2}{2!} + \frac{\omega_\times^3}{3!} + ... \tag{3.18}$$

.

In the case of $\mathbf{SO}(3)$ the exponential map has a closed form, known as the *Rodriguez formula*:

$$\exp(\omega_\times) = \begin{cases} I + [\boldsymbol{\omega}]_\times + \frac{1}{2}[\boldsymbol{\omega}]_\times^2 = I, & \text{for } \theta = 0 \\ I + \frac{\sin(\theta)}{\theta}[\boldsymbol{\omega}]_\times + \frac{1-\cos(\theta)}{\theta^2}\frac{1}{2}[\boldsymbol{\omega}]_\times^2, & \text{otherwise} \end{cases} \tag{3.19}$$

with

$$\theta = ||\boldsymbol{\omega}||_2 \tag{3.20}$$

.

The logarithmic map is the inverse to the exponential map, and is given by:

$$\exp : \mathbf{SO}(3) \longrightarrow \mathfrak{so}(3), \tag{3.21}$$

$$\log(\mathbf{R}) = \begin{cases} \frac{1}{2}(\mathbf{R} - \mathbf{R}^T) = 0, & \text{for } d = 1 \\ \frac{\arccos(d)}{2\sqrt{1-d^2}}(\mathbf{R} - \mathbf{R}^T), & \text{for } d \in (-1, 1) \end{cases} \tag{3.22}$$

with:

$$d = \frac{1}{2}(\text{trace}(\mathbf{R}) - 1). \tag{3.23}$$

**Derivatives:** Given the mapping from the rotation Lie group $\mathbf{SO}(3)$ into its lie algebra vector space $\mathfrak{so}(3)$, we can now perform optimisation of rotation matrices parameterised through the compact representation $\boldsymbol{\omega}$. The partial derivatives of the exponential map with respect to the k-th element of $\boldsymbol{\omega}$ is given by its corresponding generator, that is:

$$\frac{\partial}{\partial \omega_k} \exp([\boldsymbol{\omega}]_\times)\Big|_{\boldsymbol{\omega}=0} = \mathbf{G}_k. \tag{3.24}$$

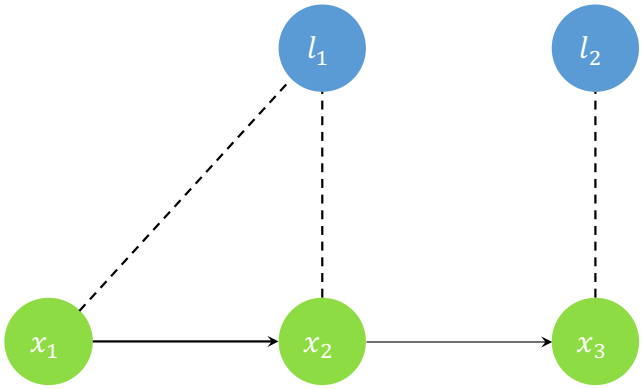



Figure 3.2: A SLAM example where a robot is moving, robot position in time shown though green nodes, and landmark position with blue nodes. The dashed lines represent indicate landmark observations.

Because the tangent space is defined around the identity, we calculate the partial of the increment around $R$ as:

$$\left.\frac{\partial \exp([\boldsymbol{\omega}]_\times)R}{\partial \omega_k}\right|_{\boldsymbol{\omega}=0} = \mathbf{G}_k\mathbf{R}. \tag{3.25}$$

## 3.4 Factor Graphs

For inference problems, *factor graphs*, a type of *probabilistic graphical model* (PGM), are a useful visual representation for the structure of the problem, illustrating the independence relationships between the variables. A full exposition on the subject can be found in [Dellaert et al., 2017].

**Bayes Network:** First we introduce *Bayes nets*, which are a directed graph where each node represents a random variable and arrows represent a factorization over the joint probability density over them. More precisely, for a set of random variables $\Theta = \theta_1, ..., \theta_n$, a Bayes net defines the joint probability distribution as:

$$p(\Theta) = \prod_j p(\theta_j|\pi_j), \tag{3.26}$$

where $\pi_j$ are the parents of $\theta_j$ in the graph.

**SLAM example:** We illustrate the definition of Bayes net with an abstract SLAM example shown in Figure 3.2, this problem will have the same core structure to the systems we will present later (Sections 4, 5). We have a robot moving through space with a sensor that allows it to make measurements of landmarks in the scene, visualised in Figure 3.2. In this case the we are interested in estimating both the pose of the robot at discrete timestamps $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ and the landmarks positions $\mathbf{l}_1, \mathbf{l}_2$, given measurements $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4$. So in this case $\Theta = \mathbf{X}, \mathbf{Z}$, with landmarks and poses $\mathbf{X}$, and measurements $\mathbf{Z}$, and $p(\mathbf{X}, \mathbf{Z}) = p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3)$. By the definition of the Bayes net visualised in Figure 3.2, the joint distribution can be factorized as:

$$\begin{aligned} P(\mathbf{X}, \mathbf{Z}) = & \, p(\mathbf{x}_1) p(\mathbf{x}_2|\mathbf{x}_1) p(\mathbf{x}_3|\mathbf{x}_2) \\ & * p(\mathbf{l}_1) p(\mathbf{l}_2) \\ & * p(\mathbf{z}_1|\mathbf{x}_1) \\ & * p(\mathbf{z}_2|\mathbf{x}_1, \mathbf{l}_1) p(\mathbf{z}_3|\mathbf{x}_2, \mathbf{l}_1) p(\mathbf{z}_4|\mathbf{x}_3, \mathbf{l}_2) \end{aligned} \tag{3.27}$$

.

This factorization helps with providing a qualitative description of the problem:

- $p(\mathbf{x}_1) p(\mathbf{x}_2|\mathbf{x}_1) p(\mathbf{x}_3|\mathbf{x}_2)$ is a Markov chain on the robot poses, and describes our knowledge of the robot movement dynamics.
- $p(\mathbf{l}_1) p(\mathbf{l}_2)$ is a prior distribution on the landmark poses.
- $p(\mathbf{z}_1|\mathbf{x}_1)$ describes an absolute measurement of the first robot pose.
- Finally $p(\mathbf{z}_2|\mathbf{x}_1, \mathbf{l}_1) p(\mathbf{z}_3|\mathbf{x}_2, \mathbf{l}_1) p(\mathbf{z}_4|\mathbf{x}_3, \mathbf{l}_2)$ describes the measurement model, which will be an important design element for the choice of representation of our SLAM system.

**Factor graph:** Factor graphs are a more general model than Bayes nets, better suited for representing inference problems as we will describe in the next section 3.5.

They make a distinction between the unknown variables $\mathbf{X}$ and the given measurements $\mathbf{Z}$. Figure 3.3 shows the factor graph for the previously described Bayes net, nodes represent the unknown variables or *states* $\mathbf{X}$ which are not directly observed and which we wish to estimate, and the *factors* are the constraints on these variables imposed by the measurements $\mathbf{Z}$, visualised as black dots. Formally a *factor graph* is bipartite graph, with two types of nodes **factors** $\phi_i$ and variables $\mathbf{x}_j$, with edges only connecting factors to variables. We define $\mathbf{X}_i$ as the set of variables adjacent to the factor $\phi_i$. Then the factor graph defines the factorisation of the following global function:

$$\phi(\mathbf{X}) = \prod_i \phi_i(\mathbf{X}_i). \tag{3.28}$$

One distinction with a Bayes net is that the factors can be un-normalised densities, we see how this distinction is useful when solving inference problems as in the next section. Factor graphs are a very general language for a probabilistic description of inference problems, they are independent of the choice of representation or measurement function.

## 3.5 Nonlinear Optimisation

Returning to the SLAM example we are interested in finding the unknown state variables $\mathbf{X}$ from the given measurements $\mathbf{Z}$. The maximum a posterior or **MAP estimate** are the values that maximise the posterior density $P(\mathbf{X}|\mathbf{Z})$. Using Bayes theorem we see that:

$$\begin{aligned} \mathbf{X}^{MAP} &= \arg\max_{\mathbf{X}} P(\mathbf{X}|\mathbf{Z}) \\ &= \arg\max_{\mathbf{X}} \frac{P(\mathbf{Z}|\mathbf{X})P(\mathbf{X})}{P(\mathbf{Z})}, \end{aligned} \tag{3.29}$$

Where $P(\mathbf{X})$ is a *prior distribution* on the states, $P(\mathbf{Z}|\mathbf{X})$ is the *measurement likelihood*, and $P(\mathbf{Z})$ is a normalisation constant. Now in this form, we can discard the

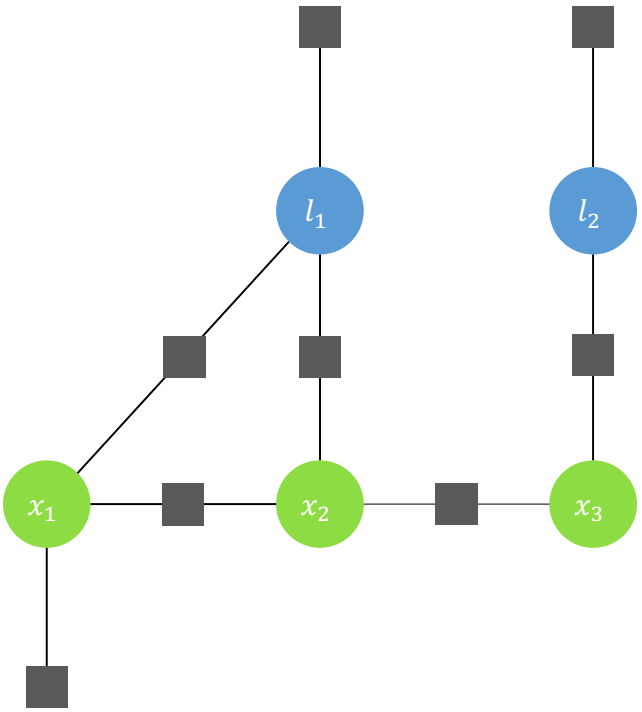



Figure 3.3: Factor graph of SLAM example, nodes represent variables with unknown state to be estimated, black squares are factors which constrain the variables on observed measurements.

term $P(\mathbf{Z})$, as the measurements are given and do not affect the optimisation result. The numerator can then be computed from our factor graph.

**Gaussian distributions:** We will now look at a particular case in which we assume that the factor densities have the form of a multinomial **Gaussian distribution**. For example, we assume the density of the measurements given the variables, has the following normal distribution:

$$p(\mathbf{Z}|\mathbf{X}) = \mathcal{N}(\mathbf{Z}, h(\mathbf{X})) \propto \exp(-(\mathbf{Z} - h(\mathbf{X}))^T \Sigma_{\mathbf{Z}}^{-1} (\mathbf{Z} - h(\mathbf{X}))) \tag{3.30}$$

. Where $h(\mathbf{X})$ is the **measurement prediction function** and $\Sigma_{\mathbf{Z}}$ is the measurement covariance matrix. When conditioned on given measurements $\mathbf{Z}$ we denote the density as $l(\mathbf{X}; \mathbf{Z}) = p(\mathbf{Z}|\mathbf{X})$ to make it explicit it is only a function of the unknown variables $\mathbf{X}$. This function is not a Gaussian distribution on $\mathbf{X}$, because of the generally non-linear function $h()$. Hence, we can now formulate the SLAM problem

introduced earlier using our factor graph definition, the likelihood is the product of all factors:

$$\begin{aligned}\Phi(\mathbf{l}_1, \mathbf{l}_2, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \phi(\mathbf{x}_1)\phi(\mathbf{x}_2|\mathbf{x}_1)\phi(\mathbf{x}_3|\mathbf{x}_2) \\ * \phi(\mathbf{l}_1)\phi(\mathbf{l}_2) \\ * \phi(\mathbf{x}_1) \\ * \phi(\mathbf{x}_1, \mathbf{l}_1)\phi(x_2, \mathbf{l}_1)\phi(x_3, \mathbf{l}_2)\end{aligned} \tag{3.31}$$

, where the factors are given by likelihood functions, as described above.

We will now introduce different optimisation techniques for solving the MAP optimisation problem for the unknown variables $\mathbf{X}$. For convenience in optimisation this is equivalent to minimizing the negative logarithm of the function:

$$\begin{aligned}\mathbf{X}^{MAP} &= \arg\max_{\mathbf{X}} \Phi(\mathbf{X}) \\ &= \arg\min_{\mathbf{X}} -log(\Phi(\mathbf{X}))\end{aligned} \tag{3.32}$$

.

Because of the nonlinear measurement function, this optimisation problem can not be solved directly, and iterative techniques must be used.

**Gradient descent:** If $\Phi()$ is differentiable we can use gradient descent to find a local minimum in the neighbourhood of an initial estimate $\mathbf{X}^{(0)}$. Given the property that the negative gradient of $\Phi()$ gives the direction of steepest descent, we perform the following iterative optimisation algorithm:

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k+1)} - \alpha_k \nabla\Phi(\mathbf{X}^{(k)}) \tag{3.33}$$

The factor $0 < \alpha_k < 1$ is the *step size*, and controls the trade-off between convergence speed and stability.

**Newton Method** A more efficient optimisation algorithm requires $\Phi$ to be twice differentiable (that is $H_\Phi$ is positive semi-definite in this neighbourhood), and that we have an estimate of $\mathbf{X}^{(0)}$ close to the local minimum $\bar{\mathbf{X}}$, which satisfies $\nabla\Phi(\bar{\mathbf{X}}) = 0$. Given that $\mathbf{X}^{(0)}$ is in the neighborhood of $\bar{\mathbf{X}}$, we can estimate $\nabla\Phi(\bar{\mathbf{X}})$ using its first order Taylor expansion:

$$\nabla\Phi(\bar{\mathbf{X}}) \approx \nabla\Phi(\mathbf{X}_0) + H_\Phi(\mathbf{X}_0)(\bar{\mathbf{X}} - \mathbf{X}_0). \tag{3.34}$$

And then following the condition $\nabla\Phi(\bar{\mathbf{X}}) = 0$ we get the iterative following update rule:

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - H_\Phi^{-1}(\mathbf{X}^{(k)})\nabla\Phi(\mathbf{X}^{(k)}). \tag{3.35}$$

This is equivalent to performing the following update:

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \boldsymbol{\delta}, \tag{3.36}$$

where $\delta$ is obtained by solving the following linear system, which were refer to as *normal equation*:

$$H_\Phi(\mathbf{X}^{(k)})\boldsymbol{\delta} = -\nabla\Phi(\mathbf{X}^{(k)}). \tag{3.37}$$

This formulation is preferred for numerical stability in implementation.

**Gauss-Newton** In the case of assuming a Gaussian density likelihood then we have:

$$\begin{aligned}
\mathbf{X}^{MAP} &= \arg\min_{\mathbf{X}} -\log(\Phi(\mathbf{X})) \\
&= \arg\min_{\mathbf{X}} -\log(\exp(-(\mathbf{Z} - h(\mathbf{X}))^T\Sigma_{\mathbf{Z}}^{-1}(\mathbf{Z} - h(\mathbf{X})))) \\
&= \arg\min_{\mathbf{X}} (\mathbf{Z} - h(\mathbf{X}))^T\Sigma_{\mathbf{Z}}^{-1}(\mathbf{Z} - h(\mathbf{X})).
\end{aligned} \tag{3.38}$$

This category of problems is referred to as *least squares*. In this case the gradient and Hessian are give by:

$$\nabla\Phi(\mathbf{X}) = J_h(\mathbf{X})^T \Sigma_{\mathbf{Z}}^{-1} r(\mathbf{X}), \tag{3.39}$$

$$H_\Phi(\mathbf{X}) = J_h(\mathbf{X})^T \Sigma_z^{-1} J_h(\mathbf{X}) + H_h(\mathbf{X}) \Sigma_{\mathbf{Z}}^{-1} r(\mathbf{X}), \tag{3.40}$$

where $J_h$ and $H_h$ are the Jacobian and Hessian matrices of $h(\mathbf{X})$ respectively, and $r(\mathbf{X}) = \mathbf{Z} - h(\mathbf{X})$, is referred to as the residual. We can therefore approximate the Hessian of $\Phi$ through:

$$H_\Phi(\mathbf{X}) \approx J_h(\mathbf{X})^T \Sigma_{\mathbf{Z}}^{-1} J_h(\mathbf{X}) \tag{3.41}$$

We can then substitute this approximation into the Newton normal equation, to get the *Gauss-Newton* update:

$$(J_h^T \Sigma_z^{-1} J_h)\boldsymbol{\delta} = -J_h^T \Sigma_z^{-1} r. \tag{3.42}$$

## 3.6 Deep Neural Networks

We will use Neural Networks (NNs) as learnable modules for machine learning, both for discriminative inference as well as generative representation in the presented systems. In this section we give a brief overview of the architectures relevant in our presented works. An in depth exposition can be found in [Goodfellow et al., 2016].

An NN is a function $F$ that maps some input $x \in \mathbb{R}^N$ to an output $y \in \mathbb{R}^M$ that depends on some learnable parameters $\Theta$, such that $y = F(x, \Theta)$. The choice of input and output depends on the application to be modeled. In classification for example $x$ is an image and $y$ is a class label. The parameters $\Theta$ are learned through gradient based optimisation (Section 3.5) to approximate a unknown function with a dataset of input output pairs $\{x_i, y_i\}$, a process known as *deep learning*.

### 3.6.1 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) or fully-connected network is the basic neural network architecture and is the model used for *neural fields* (Section 3.8.3), and the

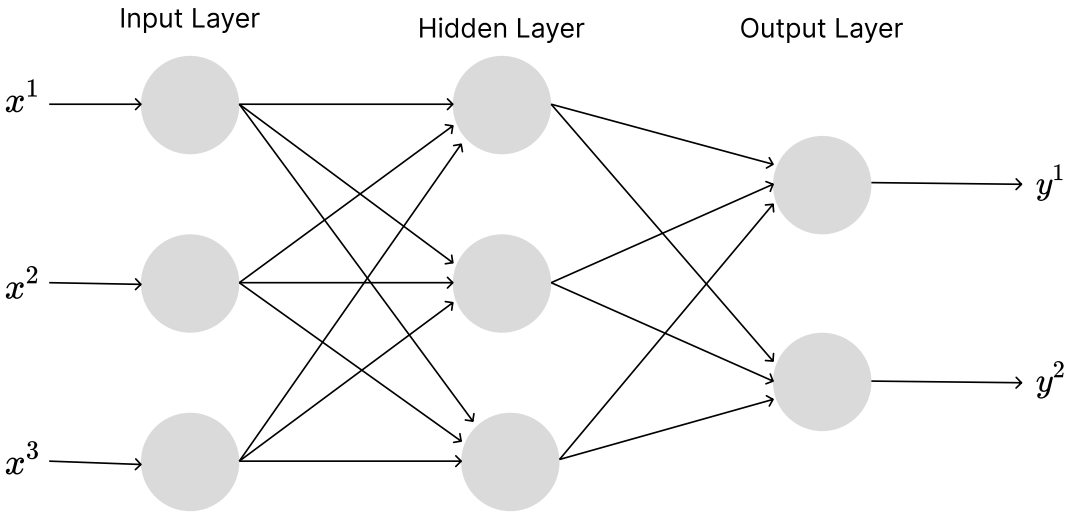



Figure 3.4: Diagram of a Multi-Layer Perceptron (MLP) with 1 hidden layer.

map representation in iMAP, Chapter 5. The MLP consists of stacked blocks of linear transformations followed by an element-wise non-linearity referred to as an *activation function*. For each of these blocks (except the last one) is referred to as a *hidden layer* and is defined as:

$$\mathbf{x}_{i+1} = f_i(\mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i), \quad i \in (1, ..., L), \tag{3.43}$$

Where $L-1$ is the number of hidden layers, $x_0$ and $x_{L+1}$ are the input and output to the neural network respectively, $\mathbf{W}_i \in \mathbb{R}^{N\times N}$ are the *weights*, $b_i \in \mathbb{R}^N$ the *biases*, and $f_i$ the activation function. The intermediate network outputs $x_i, i \in (1, ..., L)$ are referred to as features. A schematic of an MLP can be seen in Figure 3.4; the nodes are normally referred to as *neurons*.

A common choice for the activation function for hidden layers is ReLU, which is a *hockey-stick* shaped function which can "switch off" neurons below 0, defined as:

$$\text{ReLU}(\mathbf{x})_j = \max[0, \mathbf{x}_j] \tag{3.44}$$

For the output layer two common activation functions that will be relevant for us

are the *sigmoid*, used for binary classification, which squashes the output into the $[0, 1]$ range:

$$\sigma(\mathbf{x})_j = \frac{1}{1 + e^{x_j}}, \tag{3.45}$$

and *softmax* used for multi-class classification, and makes a vector sum to 1 so it becomes a discrete probability distribution:

$$\text{softmax}(\mathbf{x})_j = \frac{e^{x_j}}{\sum_k e^{x_k}}. \tag{3.46}$$

### 3.6.2 Convolutional Neural Networks

A specific case of fully-connected networks which are particularly suited for grid-like structures such as images or voxel grids is the Convolutional Neural Network (CNN). This architecture leverages the self-similar and local structure in images for efficiency through a representation with sparsity, parameter sharing, and translation equivariance. This is done by substituting the fully connected layer with a special case the discrete convolution:

$$\mathbf{I}_{i+1} = f_i(\mathbf{K}_i \circledast \mathbf{I}_i + \mathbf{b}_i), \tag{3.47}$$

where the $\circledast$ is the convolution operator. In the case where $\mathbf{I}$ is a two-dimensional image, convolving the kernel $\mathbf{K} \in \mathbb{R}^{K \times K}$ is defined as:

$$S(i, j) = \mathbf{K} \circledast \mathbf{I} = \sum_k^K \sum_l^K I(i - k, j - l)K(k, l), \tag{3.48}$$

an operation visualised in Figure 3.5.

In practice at each layer several kernels are applied, stacking the output images to form a three-dimensional grid referred to as a tensor. In this case the kernels applied thereafter are three-dimensional. Therefore the weights of a hidden layer have dimension $\mathbb{R}^{N \times M \times K \times K}$, where $N$ is the number of kernels to be applied, $M$

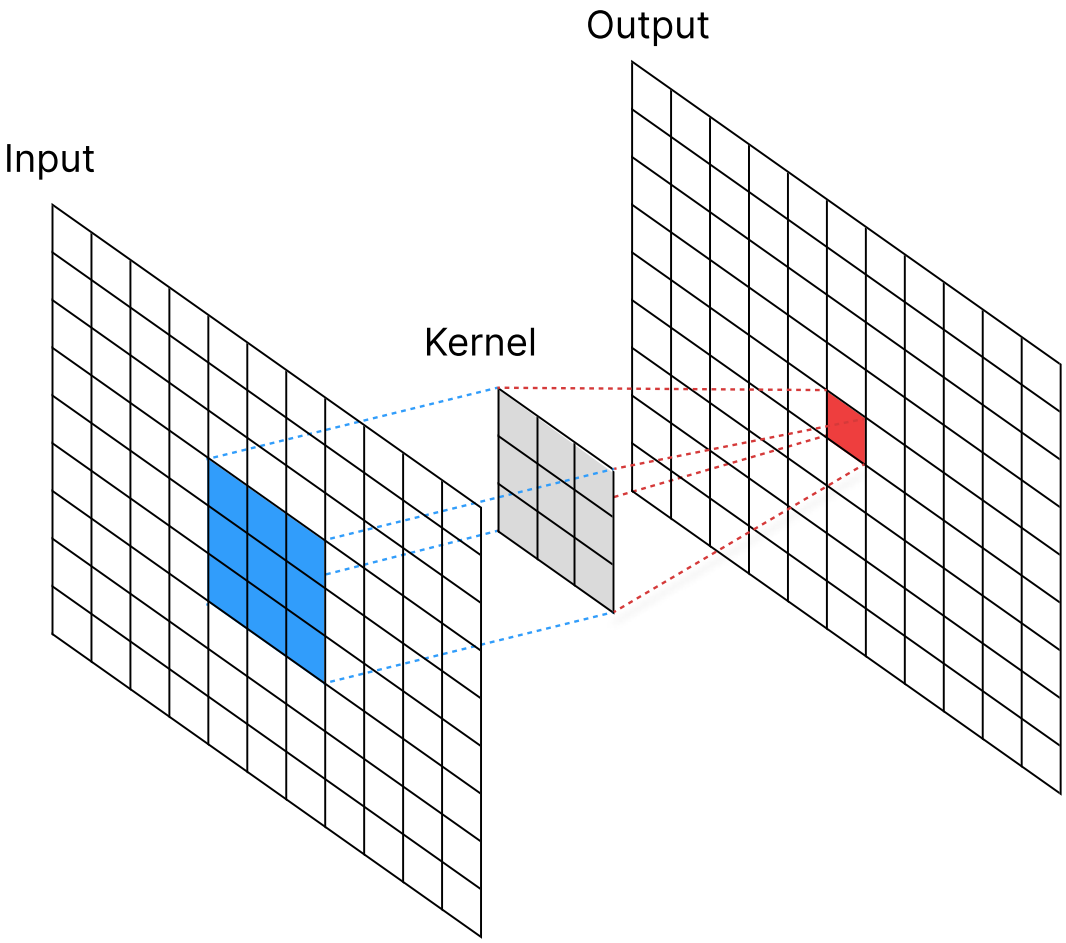



Figure 3.5: Visualisation of convolving a kernel with an input image.

the number of kernels applied in the previous layer (referred to as channel size of the input tensor), and $k$ the kernel size normally taking in the range of values ($k = 3, 5, 7$). As we can see the weight size is independent of the image size, which leads to smaller networks than using an MLP to process images.

An important property of a CNN, is that the spatial structure is preserved in intermediate feature outputs, which allows spatial resizing such *down-sampling* or *up-sampling* operations, which will be particularly relevant in the bottleneck CNN architectures discussed in Subsection 3.7.1. The resizing operation is a common fixed function (no learnable parameters) applied after the activation function of each layer in a CNN. Down-sampling reduces the spatial dimensions of the input tensor. One

common operation for this is *pooling* which summarises the local neighborhood in a local window, and is applied in a similar way to a convolution but with a fixed operation, such as taking the maximum in the window (*max-pooling*) or the average (*average-pooling*). For increasing the spatial dimension of an input *up-sampling* operations such as bi-linear interpolation or nearest neighbor are used. The output of a CNN is an image of features, which in a discriminative network is normally much smaller than the original input, and can therefore be vectorised and processed by a fully.connected layer to obtain classification scores.

## 3.7 Deep Generative Models

In Section 3.4 we described probabilistic graphical models for calculating the distribution of the observed measurements $\mathbf{Z}$ given some hidden variables or states $\mathbf{X}$, $p(\mathbf{Z}|\mathbf{X})$, where we have an explicit model of the problem. However, for certain applications we may want to learn or discover a relationship between some hidden or latent variable $\mathbf{z}$, and the observed data $\mathbf{x}$ (notice we are using the opposite notation in this case where $\mathbf{x}$ is now the observed variable). An example would be if we have a dataset of faces we may want to discover some low dimensional features, such as the shape of the nose or the distance between the eyes, to represent without explicitly modeling it. The use of neural networks for learning generative models is called Deep Generative Models (DGM), and in the next section we will describe one particular instance we use in NodeSLAM, Chapter 4, the Variational Auto-Encoder (VAE) [Kingma and Welling, 2014].

### 3.7.1 Variational Auto-Encoder

DGMs can perform different tasks such as density estimation, data generation, data compression, and data interpolation. For our applications we are interesting in discovering a latent space which is **compact** and **smooth**, to be used within an optimisation framework. For this reason we choose a probabilistic auto-encoder model, the Variational Auto-Encoder (VAE).

Given a dataset $x_{i=1}^{N}$ of samples from a random variable $\mathbf{x}$ which we assume is generated from some underlying latent variable $\mathbf{z}$ (of much smaller dimension), we wish to learn a generative model $p_\Theta(\mathbf{x}, \mathbf{z})$. For this first we define a deterministic *auto-encoder* (AE) neural network. An AE is a bottleneck network with two components: an *encoder* network $E_\Phi(x_i) = z_i$ that maps a data point to a compressed vector $z_i$, and a *decoder* network that aims to reconstruct the data point from the compressed vector $D_\Theta(z_i) = \hat{x}_i$; see Figure 4.4. This networks are optimised to maximise the conditional data likelihood, whose form for example inthe case of Bernoulli variables is given by:

$$p_\Theta(\mathbf{x}|\mathbf{z}) = \prod_i \text{Ber}(x_i|\sigma(D_\Theta(E_\Phi(z_i)))). \tag{3.49}$$

Now a VAE extends an AE so that the latent space is probabilistic, and we can sample from it (in an AE there is no structure imposed on a latent space). To do this we assume a prior on the latent space $p(\mathbf{z})$, taken to be a unit Gaussian distribution. To optimise the data likelihood given the prior, variational inference [Murphy, 2023] is used by approximating the posterior with a recognition model:

$$q_\Phi(\mathbf{z}|\mathbf{x}) = q_\Phi(\mathbf{z}|E_\Phi(\mathbf{x})) \approx p(\mathbf{x}|\mathbf{z}), \tag{3.50}$$

where $q_\Phi(\mathbf{z}|\mathbf{x})$ is a Gaussian with parameters predicted by the encoder network:

$$q_\Phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \text{diag}(\exp(\boldsymbol{\ell}))), \tag{3.51}$$

$$(\boldsymbol{\mu}, \boldsymbol{\ell}) = E_\Phi(\mathbf{z}), \tag{3.52}$$

where $\boldsymbol{\ell} = \log(\boldsymbol{\sigma})$. See Figure 4.4 for an illustration of the VAE architecture. The process of using an inference network for inverting the generative model rather than optimisation of the latent code is known as *amortised* inference. However in NodeSLAM (Section 4) we will do latent code optimisation as we do not have direct access to the data $\mathbf{x}$, but a partial measurement.

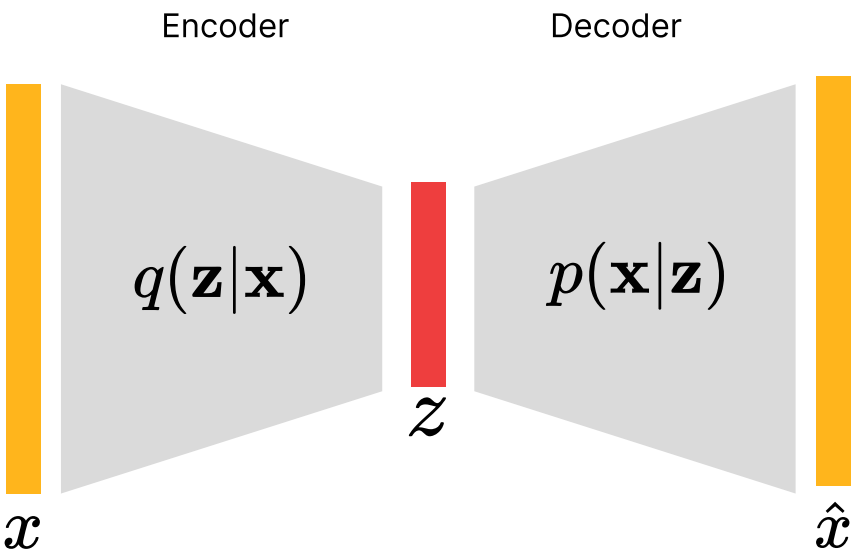



Figure 3.6: Architechture of Variational Autoencoder network.

To optimise the model parameters, the evidence lower bound (ELBO) of the joint likelihood $p(\mathbf{x}, \mathbf{z})$ is maximised, which is given by:

$$L(\mathbf{x})_{\Phi,\Theta} = \mathbb{E}[\log(p_\Theta(\mathbf{x}|\mathbf{z}))] - KL(q_\Phi(\mathbf{z}|\mathbf{x}))||p\Theta(\mathbf{z})). \tag{3.53}$$

The first therm is the conditional log-likelihood (equation 3.49) and the second term is the KL-divergence between the prior on the latent space and the inference likelihood, which in the case that the prior is a unit Gaussian is given by the closed form equation:

$$-KL(q_\Phi(\mathbf{z}|\mathbf{x}))||p_\Theta(\mathbf{z}) = \frac{1}{2}(1 + \log(\boldsymbol{\sigma}^2) - \boldsymbol{\sigma}^2 + -\boldsymbol{\mu}^2). \tag{3.54}$$

The code $\mathbf{z}$ is sampled from the distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ by the re-paremeterisation trick: $\mathbf{z} = \epsilon \odot \boldsymbol{\sigma} + \boldsymbol{\mu}$ with $\epsilon$ sampled from a unit Gaussian.

## 3.8 Geometry

In this thesis we aim at reconstructing the 3D geometry of a scene. The properties of the $3D$ representation affect both the choice of inference algorithms and the applications it enables. There are two categories of method for modeling the 3D structure of a scene *implicit* or *volumetric* and *explicit* or *parametric* surface representation. An explicit surface represents a direct parameterisation of a $2D$ manifold embedded in a $3D$ space, such that the surface $S \subset \mathbb{R}^3$ is the mapping of a $2D$ domain $\Omega \subset \mathbb{R}^2$:

$$\mathbf{f} : \Omega \rightarrow S. \tag{3.55}$$

One example of an explicit surface is a parametric plane representation:

$$g(s,t) = \mathbf{a} + s\mathbf{v_1} + t\mathbf{v_2}. \tag{3.56}$$

For representing complex shapes it is difficult to use a single function, so it is common to split the domain into patches giving a piece-wise representation. The most common piece-wise explicit surface representations are **meshes**, where the surface is represented as the union of planar triangles; an example is visualised in Figure 3.7 (a). In practice, the mesh is defined by a collection of $3D$ vertices $P = \mathbf{p}_i$ and faces $F = \mathbf{f}_i$:

$$\mathbf{p}_i = [x_i, y_i, z_i]^T \tag{3.57}$$

,

$$\mathbf{f}_i = [k, l, m], \quad k, l, m \in 1, ..., |P|. \tag{3.58}$$

Implicit surfaces are defined by the zero crossing of a scalar valued function, such that:

$$\mathbf{F} : \mathbb{R}^3 \rightarrow \mathbb{R}, \tag{3.59}$$

$$S = \{\mathbf{x} \in \mathbb{R}^3 | F(x) = 0\}. \tag{3.60}$$

To choose a scalar valued function $\mathbf{F}$ there is a choice between discretising the $3D$ domain, such as with voxel representations as described in Section 3.8.1 and seen on Figure 3.7, or using a single continuous function as described in MLP-based neural fields in section 3.8.3.

In the presented works we use implicit surface representations for shape inference because of their flexibility in representing arbitrary topologies, and ease for doing modifications such as merging or separation, which allows for integration with optimisation based techniques. In NodeSLAM, Chapter 4, we use a discretised voxel based representation for storing occupancy, for use with a 3D CNN VAE network. In iMAP, Chapter 5 we use a continuous neural field representation.

We convert implicit representations into explicit meshes as a post-processing step by using the marching cubes algorithm [Lorensen and Cline, 1987]. The benefits of a mesh representation are rapid rendering for visualisation and queries such as finding the object extrema points, useful for grasp planning as in NodeSLAM.

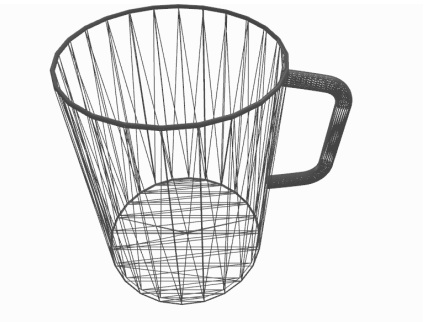
### 3.8.1 Voxel grids

Voxel representations discretise space into a 3D grid which maps a set of contiguous voxels in space to a scalar value encoding information about each voxel. A common scalar property which we use in NodeSLAM is occupancy probability. We assume an occupancy grid $\mathbf{G} \in \mathbb{R}^{N \times N \times N}$, $\mathbf{G}$ encodes a mapping so that for $i, j, k \in \{1, 2, ..., N\}$, $\mathbf{G}(i, j, k) \in [0, 1]$ stores the probability that voxel $(i, j, k)$ is occupied, an example is visualised in Figure 3.7 (c).

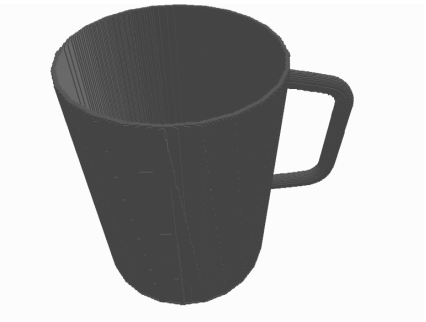
Let us consider a discretised cuboid in space (which encompasses the scene) and a mapping from the index coordinates in the cuboid $(i, j, k)$ to a voxel in space $\mathbf{V}(i, j, k) \subset \mathbb{R}^3$. Then let $\mathbf{O}(i, j, k)$ be a random variable such that:

$$\mathbf{O}(i, j, k) = \begin{cases} 1 & \text{if } \mathbf{V}(i, j, k) \text{ is occupied} \\ 0 & \text{if } \mathbf{V}(i, j, k) \text{ is free.} \end{cases} \tag{3.61}$$

An example is visualised in Figure 3.7 (b).



(a) Mesh



(b) Binary Occupancy

(c) Occupancy probability

Figure 3.7: Different representations for 3D geometry.

Now the dsicrete occupancy probability is defined as:

$$\mathbf{G}(i,j,k) = p(\mathbf{O}(i,j,k) = 1). \tag{3.62}$$

### 3.8.2 Trilinear Interpolation

In this section we will describe how to query for continuous coordinates from the discretised voxel representation. This is done by interpolating the occupancy value of a point inside a voxel grid from its 8 closest neighbours. This is called trilinear interpolation [Kang, 2006]. First we will describe the more simple case in 2D.

Suppose we have rectangle defined by its four corners $[x_0, y_0]^T$, $[x_0, y_1]^T$, $[x_1, y_0]^T$, and $[y_1, y_1]^T$ with associated values $p_{00}$, $p_{01}$, $p_{10}$, and $p_{11}$ respectively, as seen in Figure 3.8. Now we have a new point $[x, y]^T$ with $x_0 < x < x_1$ and $y_0 < y < y_1$, and we wish to interpolate its value from the given values of the four corners of the

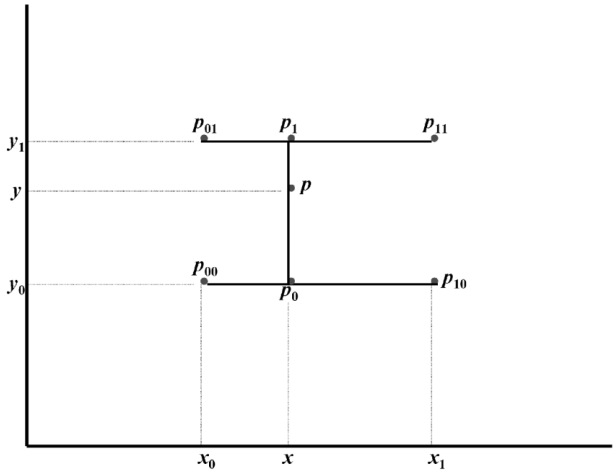



Figure 3.8: Illustration of bi-linear interpolation (adapted from [Kang, 2006]).

rectangle. This is performed by doing three linear interpolations.

First we interpolate $p_{01}$ and $p_{11}$ as:

$$p_1 := p_{01} + (p_{11} - p_{01})\frac{x - x_0}{x_1 - x_0}. \tag{3.63}$$

And $p_{00}$ and $p_{10}$ as:

$$p_0 := p_{00} + (p_{10} - p_{00})\frac{x - x_0}{x_1 - x_0}. \tag{3.64}$$

We obtain the value at $(x, y)$ by interpolating $p_0$ and $p_1$:

$$\begin{aligned} p(x, y) &= p_0 + (p_1 - p_0)\frac{y - y_0}{y_1 - y_0} \\ &= p_{00} + (p_{10} - p_{00})\frac{x - x_0}{x_1 - x_0} + (p_{01} - p_{00})\frac{y - y_0}{y_1 - y_0} \\ &+ (p_{11} - p_{01} - p_{10} + p_{00})\frac{x - x_0}{x_1 - x_0}\frac{y - y_0}{y_1 - y_0}. \end{aligned} \tag{3.65}$$

Now we will generalise the the 3D case in which we have a cuboid defined by 8 corners as illustrated on Figure 3.9. In a similar fashion we can interpolate the

value of a point $[x, y, z]^T$ inside the cuboid by performing 7 linear interpolations, obtaining the following expansion for the value at $[x, y, z]^T$:

$$p(x, y, z) = c_0 + c_1\Delta x + c_2\Delta y + c_3\Delta z + c_4\Delta x\Delta y + c_5\Delta y\Delta z + c_6\Delta x\Delta z + c_7\Delta x\Delta y\Delta z, \quad (3.66)$$

with

$$\Delta x = \frac{x - x_0}{x_1 - x_0}, \Delta y = \frac{y - y_0}{y_1 - y_0}, \Delta z = \frac{z - z_0}{z_1 - z_0}, \quad (3.67)$$

and

$$\begin{aligned}
c_0 &= p_{000}, \\
c_1 &= p_{100} - p_{000}, \\
c_2 &= p_{010} - p_{000}, \\
c_3 &= p_{001} - p_{000}, \\
c_4 &= p_{110} - p_{010} - p_{100} + p_{000}, \\
c_5 &= p_{011} - p_{001} - p_{010} + p_{000}, \\
c_6 &= p_{101} - p_{001} - p_{100} + p_{000}, \\
c_7 &= p_{111} - p_{011} - p_{101} + p_{110} + p_{100} + p_{001} + p_{010} - p_{000}.
\end{aligned} \quad (3.68)$$

For practical reasons it can be quite useful to rewrite this in matrix form. We can do this by defining the following matrices:

$$\mathbf{B} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\
1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\
-1 & 1 & 1 & -1 & 1 & -1 & -1 & 1
\end{bmatrix}. \quad (3.69)$$

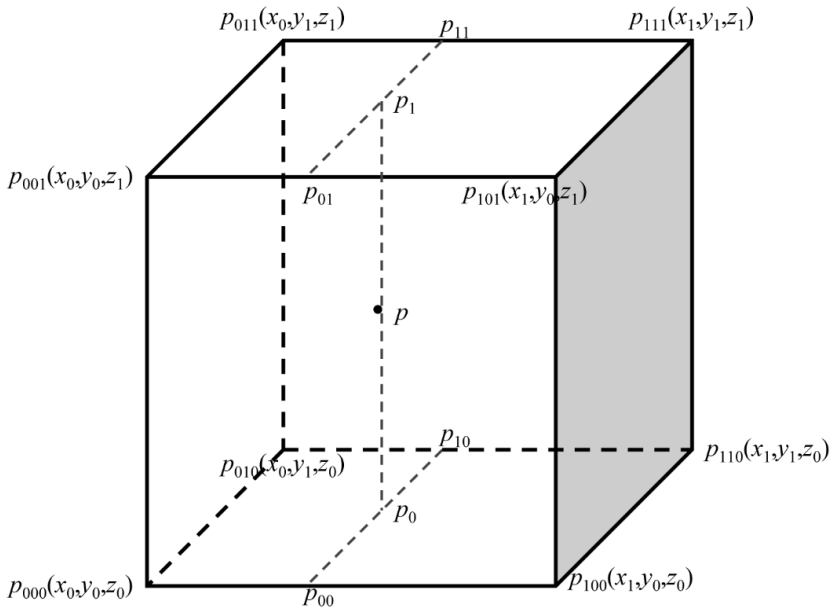



Figure 3.9: Illustration of tri-linear interpolation (adapted from [Kang, 2006]).

$$\mathbf{Q} = \begin{bmatrix} 1 & \Delta x & \Delta y & \Delta z & \Delta x\Delta y & \Delta y\Delta z & \Delta x\Delta z & \Delta x\Delta y\Delta z \end{bmatrix}^T. \tag{3.70}$$

$$\mathbf{P} = \begin{bmatrix} p_{000} & p_{001} & p_{010} & p_{011} & p_{100} & p_{101} & p_{110} & p_{111} \end{bmatrix}^T. \tag{3.71}$$

We then have:

$$p(x, y, z) = \mathbf{Q}^T\mathbf{B}\mathbf{P}. \tag{3.72}$$

In the general case we have a point $\mathbf{s}_O^i$ in object voxel grid coordinates. To calculate the coordinates of the cuboid for trilinear interpolation we take $[x_0, y_0, z_0]^T = \lfloor \mathbf{s}_O^i \rfloor$ and $[x_1, y_1, z_1]^T = [x_0, y_0, z_0]^T + [1, 1, 1]^T$. If there exist $i \in \{0, 1\}$ such that $x_i, y_i, z_i \notin \{1, ..., N\}$ ($N$ is the dimension of the voxel grid) then $o_i = 0$ since the point does not lie inside the grid. Else we take $p_{ijk} = \mathbf{G}[x_i, y_j, z_k]$, $i, j, k \in \{0, 1\}$, and use the algorithm defined above for trilinear interpolation of the occupancy value.

### 3.8.3 Neural fields

Neural fields are a continuous signal representation that avoid discretisation of space as in voxel grids. The have been explored for object reconstruction [Park et al., 2019, Mescheder et al., 2019a], object compression [Tang et al., 2020] novel view synthesis [Mildenhall et al., 2020b], and scene completion [Sitzmann et al., 2020, Chibane et al., 2020]. They form the 3D representation used in the works presented in Chapter 5 and Chapter 6 for incremental dense SLAM and interactive semantic mapping respectively.

Neural fields are defined as an MLP $F_\Theta()$ for mapping $3D$ input coordinates $\mathbf{x}$ into color and volume density scalar values $(\mathbf{c}, \sigma)$:

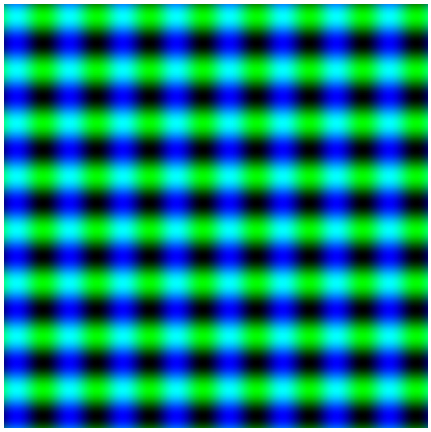
$$F_\Theta : \mathbf{x} \rightarrow (\mathbf{c}, \sigma). \tag{3.73}$$

This representation is optimised through stochastic gradient descent with respect to differentiable volumetric rendering, as described in Section 3.9 for depth and extended to color and semantics in in Chapter 5 and Chapter 6. Volume density is a generalisation of occupancy probability for rendering non solid objects as in NeRF.

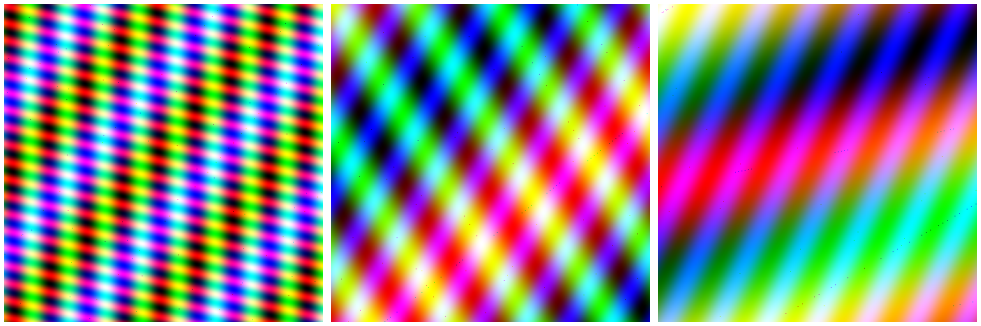
A key development in NeRF for allowing a Neural Field representation to represent higher details is **positional encoding**. Positional encoding is a decomposition of the $3D$ Euclidean space into different frequencies and is defined as:

$$\gamma(\mathbf{x}) = (sin(2^0\pi\mathbf{x}), cos(2^0\pi\mathbf{x}), ..., sin(2^{L-1}\pi\mathbf{x}), cos(2^{L-1}\pi\mathbf{x})). \tag{3.74}$$

The positional embedding is applied independently to each coordinate, and constitutes a mapping from $\mathbb{R}^3$ into a higher dimensional $\mathbb{R}^{6L}$ space, which is fed into the MLP instead of the raw coordinate. The number of frequencies $L$ controls the amount of detail that can be represented, but can lead to over-fitting of a signal. An analysis of the properties of positional encoding can be found in [Tancik et al., 2020].



(a) Axis aligned.



(b) Gaussian embedding.

Figure 3.10: Visualisation of positional encoding.

As the positional embedding is applied independently to each $3D$ coordinate this leads to a bias in axis aligned frequency representation. A generalisation of the positional encoding to remove this bias, Gaussian positional encoding, was formulated by projecting across random directions as:

$$\gamma(\mathbf{x}) = (sin(2\pi \mathbf{b_1}^T \mathbf{x}), cos(2\pi \mathbf{b_1}^T \mathbf{x}), ..., sin(2\pi \mathbf{b_m}^T \mathbf{x}), cos(2\pi \mathbf{b_m}^T \mathbf{x})), \tag{3.75}$$

Where vectors $\mathbf{b_i}$ are sampled from a Normal distribution $\mathcal{N}(0, \sigma^2)$; $m$ controls the number of projection directions and $\sigma$ the range of frequencies. Figure 3.10 shows a false color visualisation of positional encoding in $2D$ for one frequency of axis aligned, and randomly sampled frequencies from the Gaussian embedding.

## 3.9 Differential Volumetric rendering

In this section we will describe differential volume rendering, which will form our measurement function $h()$ (Section 3.5) for generative dense SLAM in both NodeSLAM (Section 4) and iMAP (Section 5). We will describe a function for predicting a depth image (as defined in Section 3.2) from an occupancy map and a camera pose. We will describe the algorithm for a map stored in an occupancy grid as in NodeSLAM, which will be generalised to an MLP map representation in iMAP.

### 3.9.1 Ray integration

We will derive the depth rendering function $d = \text{Render}(\mathbf{G}, \mathbf{T}_{OC})$, which takes as an input a voxel occupancy grid and the inverse transformation of the occuppancy grid in the camera coordinate frame, and outputs a rendered depth image of the model from the camera's perspective.

Let us assume we have an intrinsic parameter matrix $\mathbf{K}$ for the camera we are modeling. Now we will use a ray tracing algorithm to render a depth value for each pixel in the image.

For each pixel $[u, v]$ in the image do:

1. Back project pixel into a ray starting from the camera center and connecting to the center of the pixel in the image grid, $\mathbf{r} = \mathbf{K}^{-1}[u, v]^T$.
2. Sample $M$ times along the ray in the depth range $[d_{min}, d_{max}]$. Each sample $i \in \{1, ..., M\}$ has depth $d_i = d_{min} + \frac{i}{M}(d_{max} - d_{min})$ and position in the camera frame $\mathbf{s}_C^i = d_i r$.
3. Convert each sampled point into the voxel grid coordinate frame: $\mathbf{s}_O^i = \mathbf{T}_{OC}\mathbf{s}_C^i$.
4. Obtain occupancy probability $o_i := \mathbf{G}(\mathbf{s}_O^i)$ for point $\mathbf{s}_O^i$ from the occupancy grid, using trilinear interpolation as described in Section 3.8.2.
5. We will consider the depth at pixel $[u, v]$ as a random variable $D[u, v]$. Now we can calculate $p(D[u, v] = d_i)$ (that is, the termination probability at depth

$d_i$) as:

$$\phi_i := p(D[u,v] = d_i) = o_i \prod_{j=1}^{i-1}(1 - o_j). \tag{3.76}$$

for $i \in \{1..., M\}$.

6. Now we define the escape probability as:

$$\phi_{M+1} := p(D[u,v] > d_M) = \prod_{j=1}^{M}(1 - o_j). \tag{3.77}$$

It will be proven next that $\{\phi_i\}$ forms a discrete distribution.

7. We can obtain the rendered depth at pixel $[u,v]$ as:

$$\mu_{d[u,v]} := \mathbb{E}[D[u,v]] = \sum_{i=1}^{M+1} \phi_i d_i. \tag{3.78}$$

$d_{M+1}$ should be equal to $\infty$, but for practical reasons a big number is taken.

8. The uncertainty of the depth can be calculated as:

$$\sigma^2_{d[u,v]} := Var[D[u,v]] = \sum_{i=1}^{M+1} \phi_i (d_i - \mu_{d[u,v]})^2. \tag{3.79}$$

### 3.9.2 Termination Probability

We will prove by induction that $\{\phi_i\}$ is a discrete probability.

**Lemma 1** (Termination distribution)**.** *Given values $o_i \in [0,1]$ with $i \in \{1, ..., M\}$, then for $\{\phi_i\}$ as defined above we have that:*

$$\phi_i \in [0,1] \tag{3.80}$$

*for $i \in \{1, ..., M+1\}$.*

*And,*

$$\sum_{i=1}^{M+1} \phi_i = 1. \tag{3.81}$$

*Proof.* We have that $o_i \in [0,1]$ and therefore $1 - o_i \in [0,1]$. As $\phi_i$ is then a product of numbers between 0 and 1 the first proposition holds.

We will prove the second proposition by induction. For $M = 1$:

$$\begin{aligned} \sum_{i=1}^{M+1} \phi_i &= \phi_1 + \phi_2 \\ &= o_1 + (1 - o_1) \\ &= 1. \end{aligned} \tag{3.82}$$

Now let us suppose it holds for $M = N$. Let $\{\hat{\phi}_i\}$ be the distribution defined for $M = N$ and $\{\phi_i\}$ for $M = N + 1$; then:

$$\begin{aligned} \sum_{i=1}^{(N+1)+1} \phi_i &= \sum_{i=1}^{N} \phi_i + \phi_{N+1} + \phi_{N+2} \\ &= \sum_{i=1}^{N} \hat{\phi}_i + o_{N+1}\hat{\phi}_{N+1} + (1 - o_{N+1})\hat{\phi}_{N+1} \\ &= \sum_{i=1}^{N} \hat{\phi}_i + (o_{N+1} + 1 - o_{N+1})\hat{\phi}_{N+1} \\ &= \sum_{i=1}^{N+1} \hat{\phi}_i \\ &= 1 \end{aligned} \tag{3.83}$$

by the induction hypothesis.

□

### 3.9.3 Derivatives

The defined volumetric rendering function is differentiable with respect to the input voxel grid and camera pose. In this section we provide the derivative of two components of the rendering formulation with respect to the values in the occupancy voxel grid.

For a point $\mathbf{p} = [z, y, z]^T$ with interpolated occupancy $o_j = \mathbf{G}(x, y, z)$, the derivative of occupancy with respect to the value of the $l$-th voxel $o_l = \mathbf{P}_l$ (from the 8 neighbours) is given by:

$$\left.\frac{\partial o_j}{\partial o_l}\right|_{[z,y,z]^T} = \mathbf{Q}^T \mathbf{B} \begin{bmatrix} 0_1 \\ \vdots \\ 1_l \\ \vdots \\ 0_8 \end{bmatrix}, \tag{3.84}$$

with $\mathbf{Q}$, $\mathbf{P}$ and $\mathbf{B}$ as defined in Section 3.8.2.

The derivative of the termination probability $\phi_i$ is then computed by the product rule as:

$$\left.\frac{\partial \phi_i}{\partial o_l}\right|_{[z,y,z]^T} = \sum_{k=0}^{i} \frac{\partial o_j}{\partial o_l} [\prod_{j=0, j\neq k}^{i-1} (1 - o_j)] \hat{o}_i, \quad \hat{o}_i = \begin{cases} 1 & \text{if } k = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.85}$$

## 3.10 System Building

Building a SLAM system requires the integration and interfacing of different components, in particular, the systems we present at a higher level combine an optimiser library with a neural network representation and include a variety of lower level functions such as camera tracking, image object segmentation, volume rendering, bundle adjustment, data association, and pixel sampling. Also, it is often necessary to interface the SLAM systems with external modules, such as with a robotic platform in the case of NodeSLAM in Chapter 4, and with an interactive user interface in the case of iLabel in Chapter 6. For these reasons, an emphasis in our work on the systems we build is to design a modular and flexible code library.

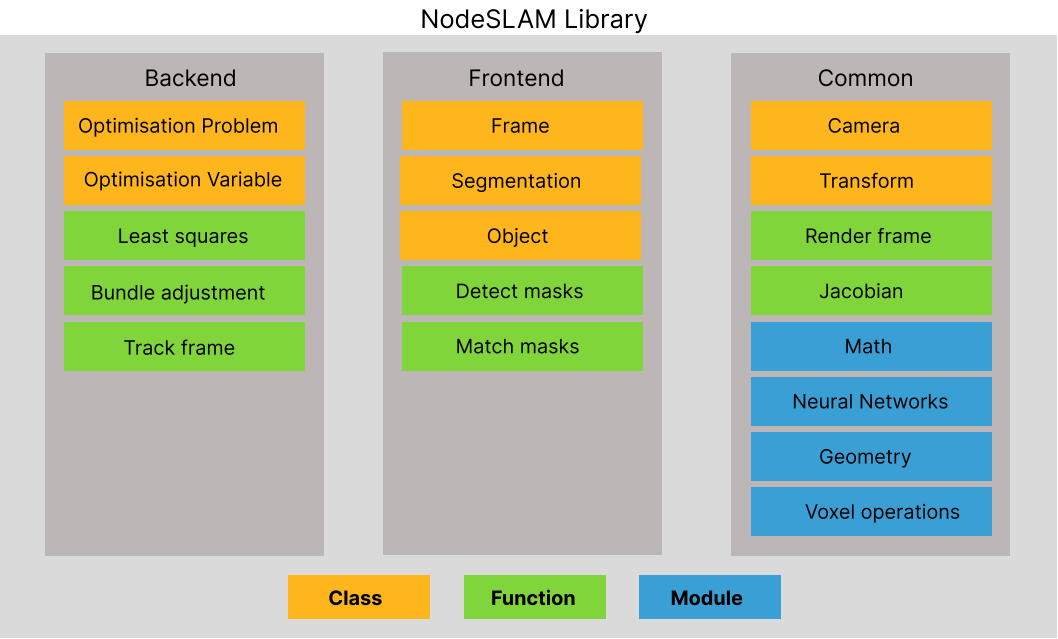



Figure 3.11: The core SLAM library for the NodeSLAM system. It is organised in three code groups: a backend used for optimisation, a frontend for defining the SLAM problem, and common for shared functions and classes.

### 3.10.1 Software Library

Our core SLAM library is implemented as a collection of classes and functions that interface with each other and are abstracted to match the SLAM problem structure. The code organisation is inspired by modern SLAM systems such as ORB-SLAM [Mur-Artal and Tardós, 2014], but uses a Python interface with GPU tensor acceleration by the PyTorch backend [Paszke et al., 2019], which allows for integration of fast matrix operations for non linear optimisation with neural networks. Figure 3.11 presents the code structure of the NodeSLAM library system, divided into three groups *backend*, *frontend*, and *common*.

The SLAM pipeline is managed by a single class object *system* which interfaces with the different elements of the library. The backend code group contains the functions and classes associated with optimisation of the constructed SLAM problem. We build an optimisation class inspired by the interface of the Pytorch *optimiser* class, but for second order non-linear squares optimisation with Gauss-Newton, Boil-

```
class OptimisationProblem:
    def __init__(self):
        self.least_squares = LeastSquares()
        self.variables = []
        self.residuals = []
        self.informations = []

    def add_variable(self, variable):
        ...

    def add_residual(self, residual):
        ...

    def solve(self):
        ...
```

Listing 3.1: Boilerplate code for custom second order optimisation class.

erplate code for this is shown in Listing 3.1. For calculation of Jacobians we leverage the auto-diff engine of PyTorch. In iMAP this optimisation library is replaced with the PyTorch optimiser for gradient descent, which shares a similar interface to our custom one.

The frontend code group contains the elements used by the system class to build the SLAM problem from the input image stream. The *frame* class encapsulates images with their associated properties such as tracked pose, and for keyframes it contains the association between detections, represented by the *segmentation* class and 3D map elements represented by the *object* class. It also contains the two main functions for interfacing with images: detection by *detect masks* and data association by *match masks*. The common code group contains different elements for representing and operating on the children objects of the frontend and backend. An important commmon function element is *render frame* which is used as the measurement function in SLAM mapping from map objects to image observations. The iMAP code has a simplified code structure as it does not include the detection and matching components.

Figure 3.12: Demonstration of iLabel system in live operation.

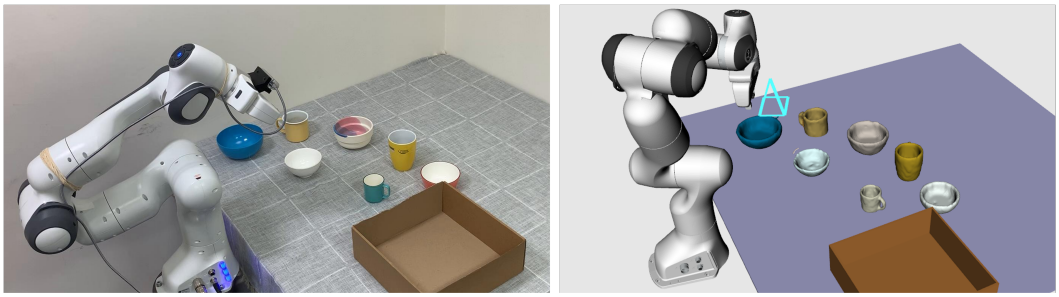


Figure 3.13: Visualisation of robotic interaction demo performed with the NodeSLAM system.

### 3.10.2 System Interfacing

For either interfacing the SLAM systems with external modules or parallelising sub-modules within them, we used a multi-processing paradigm with shared queues for resource sharing. This design allows us to standarise messages between modules independent of the communication library and to be read/write memory safe. For example for communication between our NodeSLAM (Chapter 4) system and the Franka Emika Panda robotic arm, we use the ROS API [Quigley et al., 2009], while for multiprocessing in iMAP (Chapter 5) and user interaction in iLabel (Chapter 6) we use the PyTorch multi-processing API.

Our flexible code design is demonstrated by the practical real-time applications and demos built with our SLAM libraries, such as the iLabel interactive real-time demo seen in Figure 3.12 and the robotic manipulation system presented in Figure 3.13.

Chapter 4

# NodeSLAM: Neural Object Descriptors

## Contents

## 4.1 Introduction

In this chapter we present NodeSLAM, a system which can build a 3D object graph of a scene from multi-view RGB-D images by fitting learned class-level object shape models. We build these object shape models by training a volumetric variational autoencoder (VAE) from a 3D database of aligned CAD objects of a number of known classes. At the bottleneck of the auto-encoder we obtain a small descriptor representing the range of 3D shape variation within the class. At run time, as a moving camera browses a scene and objects are detected, we add objects to our 3D scene graph frame by frame. We then perform joint optimisation of the camera trajectory, the object poses and the shape codes to minimise the difference between a rendering of our graph model and the depth data from multiple camera views.

We demonstrate our method in a table-top setting with a cluttered variety of objects from four different classes, and show that we can rapidly build an object scene graph model which is dense, precise and watertight as seen on Figure 5.1. This enables augmented reality effects such as filling bowls and cups. Compared to whole scene reconstruction methods, we obtain this whole dense model with relatively few views, by not needing to make observations all around an object to fit a watertight model. This is a strong indicator that we could also use this approach in robotics where precise object shape information is needed for grasping.

There have been two main approaches for 3D shape reconstruction from images. Classical reconstruction techniques infer geometry by minimizing the discrepancy between a reconstructed 3D model and observed data through a measurement function [Izadi et al., 2011, McCormac et al., 2018, Whelan et al., 2015]. These methods are flexible and general, but they can only reconstruct directly observed parts of a scene and are limited in accuracy when observations are weak or noisy. On the other hand, discriminative methods learn to map image measurements to 3D shape, such as through a feed-forward neural network [Gkioxari et al., 2019, Kundu et al., 2018, Wu et al., 2017, Tulsiani et al., 2017, Wang et al., 2018, Wu et al., 2015]. These methods take advantage of regularities in data for robustness but have trouble in

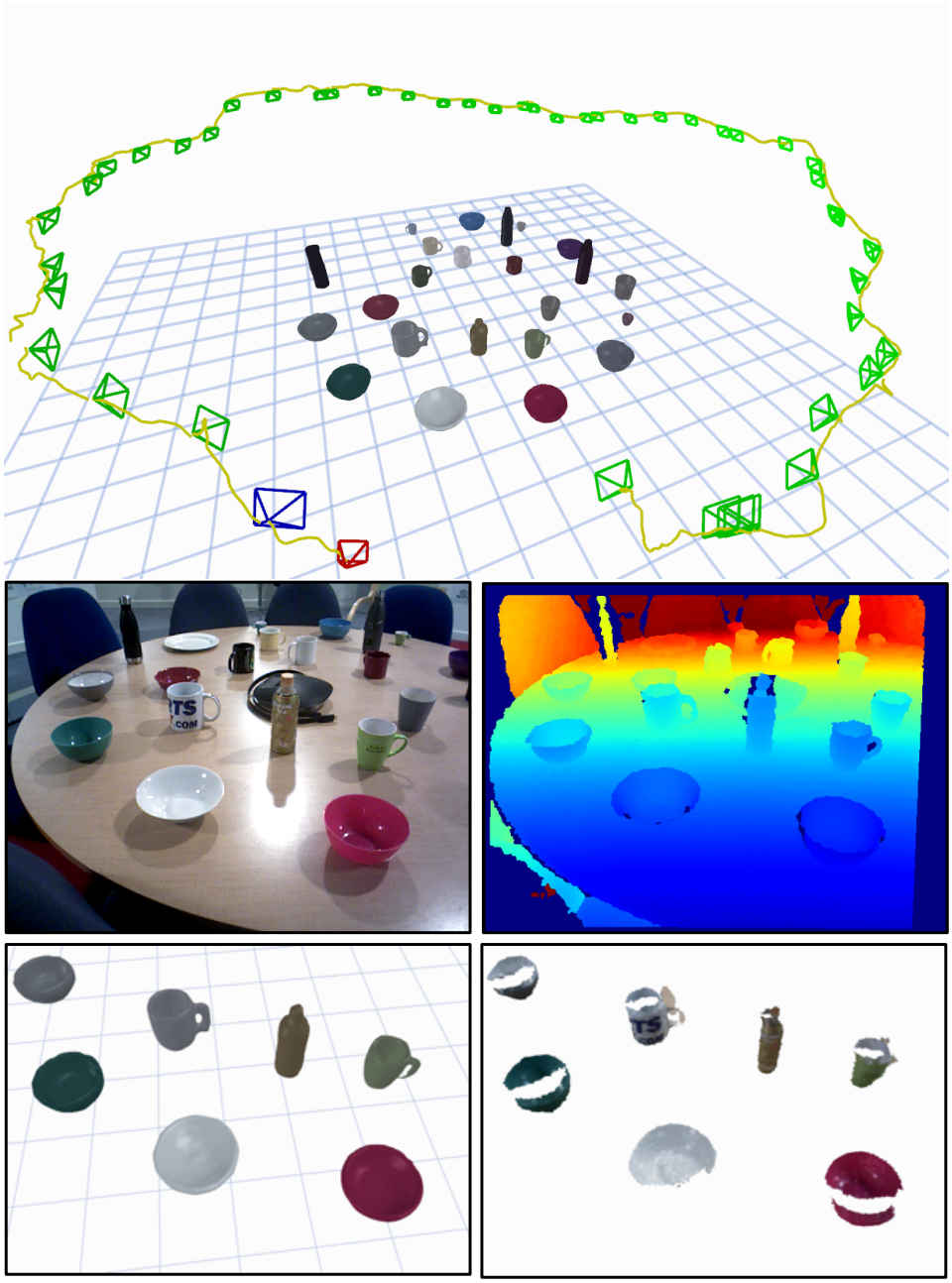


Figure 4.1: **Top:** Compact, optimisable shape models used in an object-level SLAM system which maps a real world cluttered table top scene with varied object shapes from different classes. **Bottom:** Class-level priors allow accurate and complete object reconstruction (bottom-left) even from a single image in contrast to partial reconstruction from TSDF fusion (bottom-right).

generalisation and lack the ability to integrate multiple measurements in a principled way.

Our work sits between these two approaches. We capture regularities in data though a volumetric 3D generative model represented though a class conditioned VAE, allowing us to represent object shape through a compact code. We then use the generative model for shape inference though iterative optimisation of the latent code with respect to any number of depth image measurements.

To use a generative method for inference we need a rendering function to transform 3D volumes into measurements; in our case depth images with object segmentation. The design of this function will influence optimisation speed and convergence success. Two important design considerations are (1) receptive field, the size of 3D region which influences each rendered pixel, and (2) uncertainty modeling, the confidence of each rendered pixel depth. We introduce a novel probabilistic volumetric rendering function based on these two design principles, improving the state of the art in volumetric rendering.

In scenes with many objects, our optimisable compact object models can serve as the landmarks in a SLAM system, where we use the same measurement function for camera tracking, object poses and shape optimisation. We quantitatively show that joint optimisation leads to more robust tracking and reconstruction, with comparable surface reconstruction to the data driven Fusion++ [McCormac et al., 2018], while reaching full object reconstruction from far fewer observations.

An emphasis of this work is to design object models that work robustly in the real world. We demonstrate the robustness of our proposed rendering function through qualitative demonstrations of our object-level SLAM on real world image sequences from a cluttered table-top scene obtained with a noisy depth camera, and on an augmented reality demo. Furthermore we integrate our efficient shape inference method into a real time robotic system, and show that the completeness and accuracy of our object reconstructions enable robotic tasks such as packing objects into a tight box or sorting objects by shape size.

To summarise, the key contributions of NodeSLAM are: (i) A novel volumetric probabilistic rendering function which enables robust and efficient multi-view shape optimisation. (ii) The first object-level SLAM capable of jointly optimising full object shapes and poses together with camera trajectory from real world images. (iii) The integration into a real-time robotic system that can achieve useful manipulation tasks with varied object shapes from different categories due to complete high quality surface reconstructions.

## 4.2 Class-Level Object Shape Descriptors

Objects of the same semantic class exhibit strong regularities in shape under common pose alignment. We make three key observations: (i) Given two objects of the same class, there is a pose alignment between them that allows for a smooth surface deformation between the two objects; (ii) This pose alignment is common among all instances of the same class, which defines a class-specific coordinate frame; (iii) If we select two random objects of a certain class and smoothly deform one into the other, there will be other object instances of the same class which are similar to the intermediate deformations.

We leverage these characteristics to construct a class specific smooth latent space, which allows us to represent the shape of an instance with a small number of parameters. This is motivated by the fact that the space of valid inter-class surface deformations is a much smaller sub-space than the space of all possible deformations; there are high correlations between the surface points in a valid deformation.

Rather than manually designing a parameterised shape model for a class of objects, we propose instead to learn the latent space by training a single Class-Conditional Variational Autoencoder neural network.

### 4.2.1 Network Design

3D object shapes are represented by voxel occupancy grids of dimension $32 \times 32 \times 32$, with each voxel storing a continuous occupancy probability value between 0 and 1,

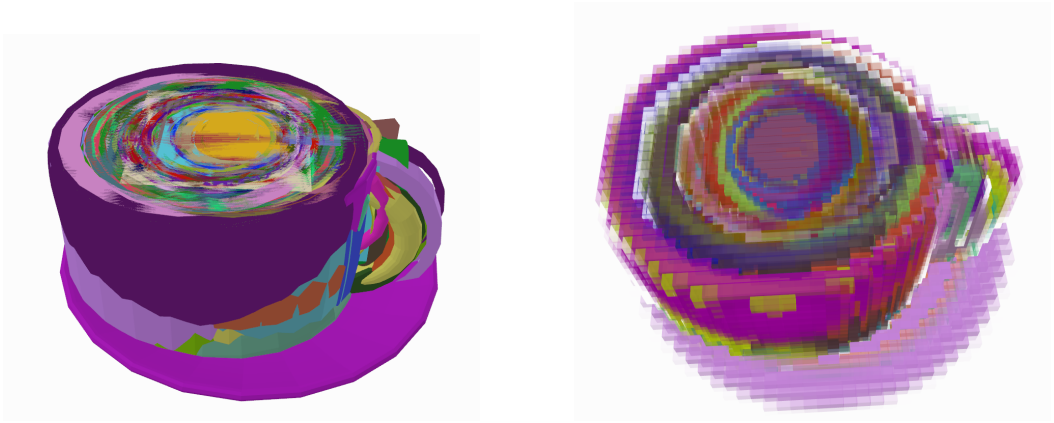


Figure 4.2: Visualisation of all aligned objects instances for training the VAE network, left: mesh models; right: occupancy grids.

see Section 3.8.1 for a technical description of voxel grids. A voxel grid was chosen to enable representation of shapes of arbitrary topology. We store occupancy values to allow a probabilistic formulation of rendering and inference.

The 3D models used were obtained from the ShapeNet database [Chang et al., 2015], which comes with annotated model alignment, in Figure 4.2 we visualise the aligned CAD and occupancy models of the *mug* category. The occupancy grids were obtained by converting the model meshes into a high resolution binary occupancy grid, and then down-sampling by average pooling. Figure 4.3 shows an example of a mug object instance with a visualisation of the mesh wire-frame (vertices and edges) and corresponding voxel occupancy grid.

A single 3D CNN Variational Autoencoder (VAE) [Kingma and Welling, 2014] was trained on objects from 4 classes: 'mug', 'bowl', 'bottle', and 'can', common table-top items, see Section 3.7.1 for a technical description of VAEs. The encoder of the network is conditioned on the class by concatenating the class one-hot vector as an extra channel to each occupancy voxel in the input, while the decoder is conditioned by concatenating the class one-hot vector to the encoded shape descriptor, similar to [Sohn et al., 2015, Tan et al., 2018]. A KL-divergence loss is used in the latent shape space, while a binary-crossentropy loss is used for reconstruction. We choose a latent shape variable of size 16. The 3D CNN (see Section 3.6.2 for definition of

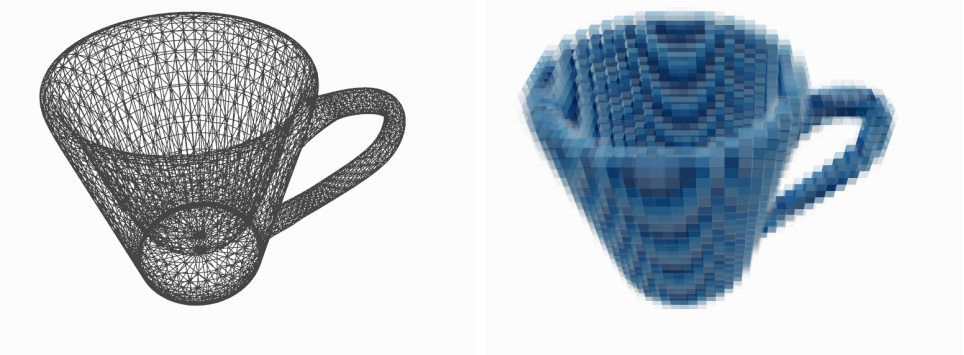


Figure 4.3: Object instance example used for training VAE: on the left is the corresponding mesh which is converted into an occupancy grid in right. Color transparency represents occupancy probability; more transparent voxels have lower probability values.

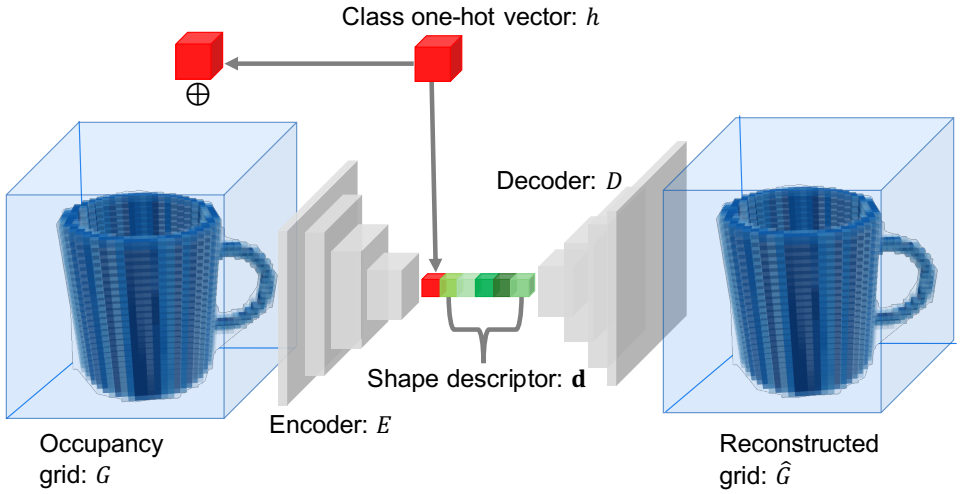



Figure 4.4: **Occupancy Variational Autoencoder**: The class one hot vector $h$ is concatenated channel-wise to each occupancy voxel in the input occupancy grid $\mathbf{G}$. The input is compressed into shape descriptor $\mathbf{d}$ by encoder network $E$. The shape descriptor and the class-one hot vector are concatenated and passed through decoder network $D$ to obtain occupancy reconstruction $\hat{\mathbf{G}}$.

CNN) encoder has 5 convolutional layers with kernel size 4 and stride 2; each layer doubles the channel size except the first one which increases it to 16. The decoder mirrors the encoder using deconvolutions.

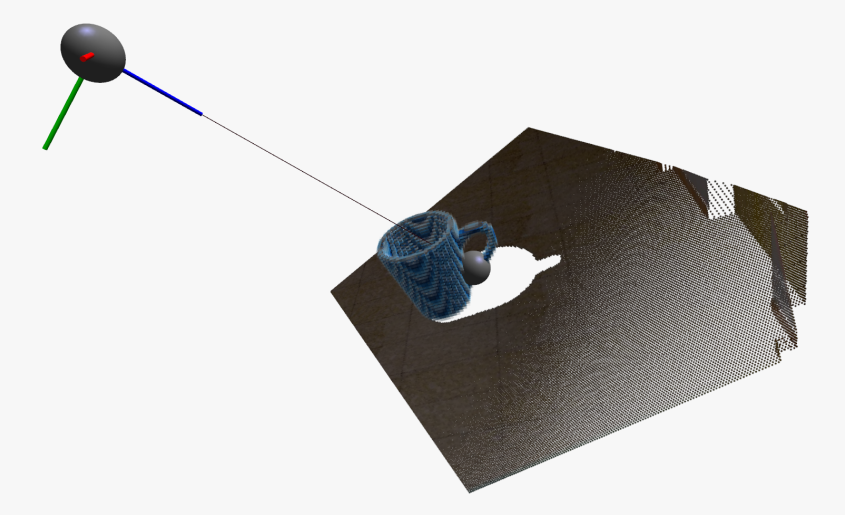


Figure 4.5: Visualisation of back-projecting a ray from the camera frame into the voxel grid of an object.

## 4.3 Probabilistic Rendering

Rendering is the process of projecting a 3D model into image space. Given the pose of the grid with respect to the camera $\mathbf{T}_{CG}$, we wish to render a depth image. We denote the rendered depth image as $\hat{\delta}_{\mu}$ with uncertainty $\hat{\delta}_{var}$, and the rendering function Render(), such that $\hat{\delta}_{\mu}, \hat{\delta}_{var} = \text{Render}(\mathbf{G}, \mathbf{T}_{GC})$. When designing our render function, we wish for it to satisfy three important requirements: to be differentiable and probabilistic so that it can be used for principled inference, and to have a wide receptive field so that its gradients behave properly during optimisation. These features lead to a robust function that can handle real world noisy measurements such as depth images.

We now describe the algorithm for obtaining the depth value for pixel $(u, v)$. See Section 3.9 for more details on the rendering algorithm

**Point sampling.** Sample $M$ points uniformly along backprojected ray $\mathbf{r}$, as seen in Figure 4.5, in depth range $[\hat{\delta}_{min}, \hat{\delta}_{max}]$. Each sampled depth $\hat{\delta}_i = \hat{\delta}_{min} + \frac{i}{M}(\hat{\delta}_{max} - \hat{\delta}_{min})$ and position in the camera frame $\mathbf{s}_C^i = \hat{\delta}_i \mathbf{r}$. Each sampled point is transformed into the voxel grid coordinate frame as $\mathbf{s}_G^i = \mathbf{T}_{GC}\mathbf{s}_C^i$.

**Occupancy interpolation.** Obtain occupancy probability $o_i = \text{Tril}(\mathbf{s}_O^i, \mathbf{G})$, for point $\mathbf{s}_O^i$ from the occupancy grid, using tri-linear interpolation from its 8 neigh-

bouring voxels. See Section 3.8.2 for technical details on tri-linear interpolation.

**Termination probability.** We denote the depth at pixel $[u, v]$ by $\mathbf{D}[u, v]$. Now we can calculate $p(\mathbf{D}[u, v] = \hat{\delta}_i)$ (that is, the termination probability at depth $\hat{\delta}_i$) as:

$$\phi_i = p(\mathbf{D}[u, v] = \hat{\delta}_i) = o_i \prod_{j=1}^{i-1} (1 - o_j) \ . \tag{4.1}$$

Figure 4.6 relates occupancy and termination probabilities.

**Escape probability.** Now we define the escape probability (the probability that the ray doesn't intersect the object) as:

$$\phi_{M+1} = p(\mathbf{D}[u, v] > \hat{\delta}_{max}) = \prod_{j=1}^{M} (1 - o_j) \ , \tag{4.2}$$

where $\{\phi_i\}$ forms a discrete probability distribution.

**Aggregation.** We obtain the rendered depth at pixel $[u, v]$ as the expected value of the random variable $\mathbf{D}[u, v]$:

$$\hat{\delta}_\mu[u, v] = \mathbb{E}[\mathbf{D}[u, v]] = \sum_{i=1}^{M+1} \phi_i \hat{\delta}_i \ . \tag{4.3}$$

$d_{M+1}$ is the depth associated to the escape probability is set to $1.1 d_{max}$ for practical reasons.

**Uncertainty.** Depth uncertainty is calculated as:

$$\hat{\delta}_{var}[u, v] = Var[\mathbf{D}[u, v]] = \sum_{i=1}^{M+1} \phi_i (\hat{\delta}_i - D[u, v])^2 \ . \tag{4.4}$$

**Mask.** Note that we can render a segmentation mask as:

$$m[u, v] = 1 - \phi_{M+1}. \tag{4.5}$$

For multi-object rendering we combine all the renders by taking the minimum depth at each pixel, to deal with cases when objects occlude each other:

$$\begin{aligned} \hat{\delta}_\mu[u, v] &= \text{Render}(\{\hat{\mathbf{G}}_i\}, \{\mathbf{T}^i_{GC}\}, \mathbf{T}_{WC})[u, v] \\ &= min\{\hat{\delta}^1_\mu[u, v], ..., \hat{\delta}^N_\mu[u, v]\} \ . \end{aligned} \tag{4.6}$$

Figure 4.6 shows the relation between rendered depth and occupancy probabilities. Additionally, we apply Gaussian blur down-sampling to the resulting rendered image at different pyramid levels (4 levels with 1 pixel standard deviation each) to

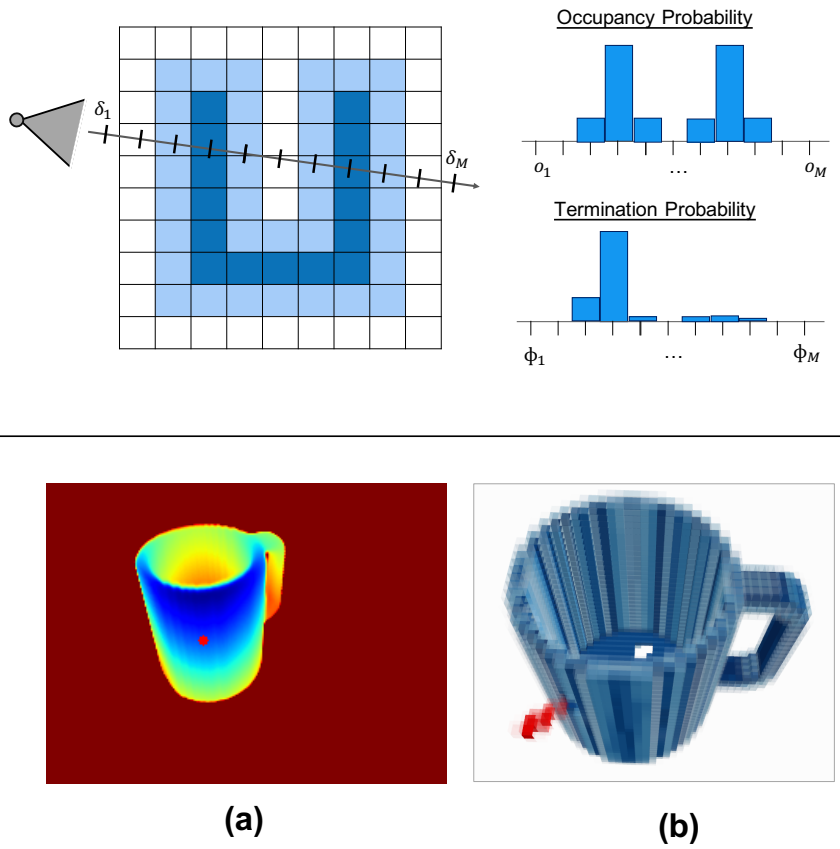



Figure 4.6: **Pixel rendering**: Each pixel is back-projected into a ray from which uniform depth samples $\delta_i$ are taken. Occupancy probability $o_i$ is obtained from the voxel grid by trilinear interpolation, and termination probability $\phi_i$ is calculated. **(a):** A $32 \times 32 \times 32$ mug occupancy grid. **(b):** The derivative of the highlighted red pixel with respect to occupancy values is shown in red.

perform coarse to fine optimisation. This increases the spatial receptive field in the higher levels of the pyramid because each rendered pixel is associated to several back projected rays.

## 4.4 Object Shape and Pose Inference

Given a depth image from an object of a known class, we wish to infer the full shape and pose of the object. We assume we have a segmentation mask and classification of the object, which in our case is obtained with Mask-RCNN [He et al., 2017]. To formulate our inference method, we integrate the object shape models developed

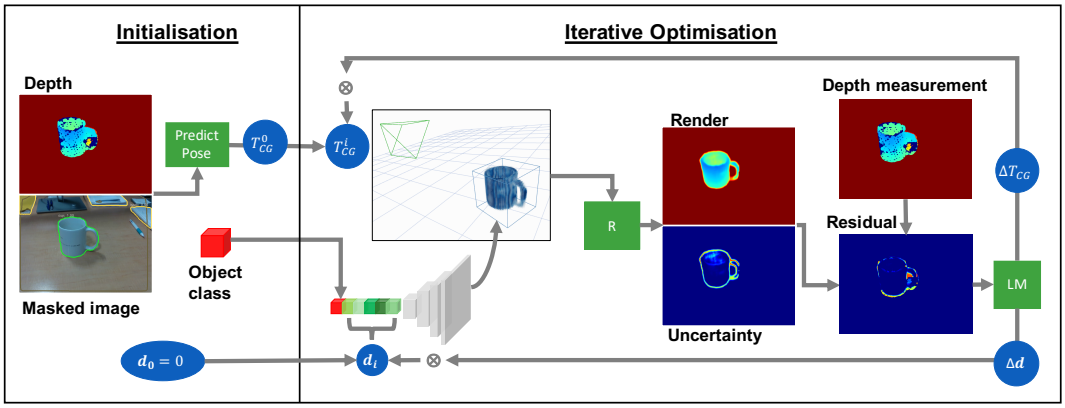



Figure 4.7: **Initialisation:** Initial object pose $\mathbf{T}_{CG}^{0}$ is estimated from a depth image and masked RGB image; object class is inferred from RGB only. The shape descriptor $\mathbf{d}$ is set to 0, representing the mean class shape. **Optimisation:** The shape descriptor is decoded into a full voxel grid, which is used with the pose to render an object depth map. The least squares residual between this and the depth is used update the shape descriptor and object pose iteratively with the Gauss-Newton algorithm.

on Section 4.2 with a measurement function, the probabilistic render algorithm outlined in Section 5.3.3. We will now describe the inference algorithm for a single object observation setup, and this will be extended to multiple objects and multiple observations in the SLAM system described in Section 4.5.

### 4.4.1 Shape and Pose Optimisation

An object's pose $\mathbf{T}_{CG}$ is represented as a 9-DoF homogeneous transform with $\mathbf{R}_{CG}$, $\mathbf{t}_{C}$, and $\mathbf{S}_{G}$ the rotation, translation and scale of the object.

The shape of the object is represented with latent code $\mathbf{d}$, which is decoded into a full occupancy grid $\hat{\mathbf{G}}$ using the decoder described in Section 4.2.

We wish to find the pose and shape parameters that best explain our depth measurement $\delta$. We consider the rendering $\mathbf{D}$ of the object as Gaussian distributed, with mean $\hat{\delta}_{\mu}$ and variance $\hat{\delta}_{var}$ calculated through the render function:

$$\begin{aligned} \hat{\delta}_{\mu}, \hat{\delta}_{var} &= R(\hat{\mathbf{G}}, \mathbf{T}_{GC}) \\ &= R(D(\mathbf{d}, h), \mathbf{T}_{GC}) \ , \end{aligned} \tag{4.7}$$

Figure 4.8: **Shape descriptor influence:** the derivative of a rendered decoded voxel grid with respect to 8 entries of the shape descriptor.

with $h$ the class one-hot vector of the detected object.

When training the latent shape space a Gaussian prior distribution is assumed on the shape descriptor. With this assumption and by taking $\hat{\delta}_{var}$ as constant, our MAP objective takes the form of least squares problem. We apply the Gauss-Newton algorithm, Section 3.5, for estimation:

$$\begin{aligned} &\min_{\mathbf{d},\mathbf{T}_{CG}} -log(p(\delta|\mathbf{d},\mathbf{T}_{CG})p(\mathbf{d})) \\ &= \min_{\mathbf{d},\mathbf{T}_{CG}} (L_{render}(\mathbf{d},\mathbf{T}_{CG}) + L_{prior}(\mathbf{d})) \\ &= \min_{\mathbf{d},\mathbf{T}_{CG}} (\sum_{u,v} \frac{(\delta[u,v] - \hat{\delta}_{\mu}[u,v])^2}{\hat{\delta}_{var}[u,v]} + \sum_{i} \mathbf{d}_i^2). \end{aligned} \tag{4.8}$$

A structural prior is added to the optimisation loss to force the bottom of the object to be in contact with the supporting plane. We render an image from a virtual camera under the object and recover the surface mesh from the occupancy grid by marching cubes. Figure 4.7 illustrates the single object shape and pose inference pipeline.

### 4.4.2 Variable Initialisation

Second order optimisation methods such as Gauss-Newton require a good initialisation. The object's translation and scale are intitialised using the back-projected

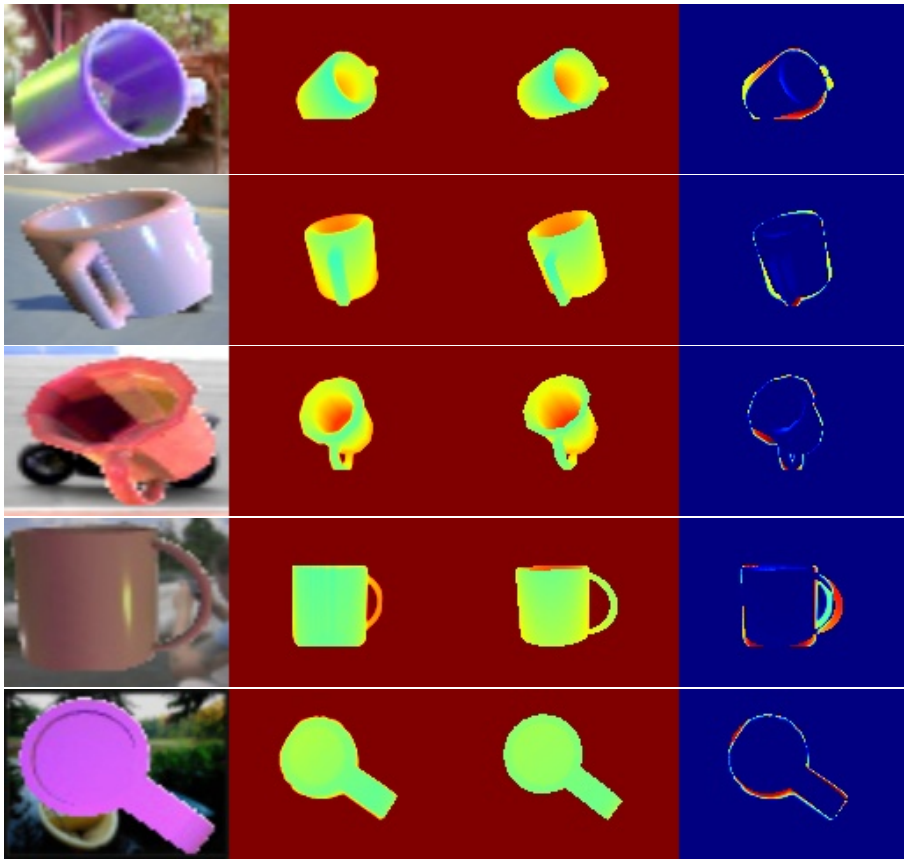


Figure 4.9: Examples of the synthetic dataset used to train the pose prediction network. First: rendered input image; second: pose prediction; third: ground truth; fourth: render discrepancy (not used for network supervision).

point cloud from the masked depth image. The first is set to the centroid of the point cloud, while the latter is recovered from the centroid's distance to the point cloud boundary. Only un-occluded objects are initialised.

For pose initialisation we use a discriminative CNN Network (Section 3.6). Given the pose of the object $\mathbf{T}_{CG}$ represented as a $4 \times 4$ homogeneous matrix, we define the orientation as the $3 \times 3$ rotation matrix $\mathbf{R}_{CG}$ of $\mathbf{T}_{CG}$. We parameterise $\mathbf{R}_{CG}$ as a rotation of magnitude $\Theta$ along axis normalised $e$. We represent $e$ in polar coordinates as $e_\theta$, $e_\phi$. The CNN orientation prediction network must predict $\Theta$, $e_\theta$, and $e_\phi$. Figure 4.9 shows some training examples from our synthetic dataset.

The CNN input is a cropped (around the object) and resized to 224×223 3-channel

color image. The first 2 convolutional and max-pool layers are taken from the VGG-11 architecture [Simonyan and Zisserman, 2015] pre-trained on ImageNet [Deng et al., 2009], with frozen weights throughout training. Following these layers are four convolutional layers with ReLu activations and batch normalisation, a kernel size of 4 and stride 2, with 256 initial channels and doubled after each layer. The network has two fully connected layers at the end with a final sigmoid activation layer which is then normalized to the corresponding angle range. The network was trained for 227375 iterations with batch size 32. Figure 4.10 shows results of the pose prediction network on real world instances, showing generalisation from the synthetic training. Our model classes ('mug', 'bowl', 'bottle', and 'can') are often found in a vertical orientation in a horizontal surface. For this reason we detect the horizontal surface using the point cloud from the depth image and initialise the object's orientation to be parallel to the surface normal, thereby taking only the vertical component of the pose prediction neural network.

The shape descriptor is initialised to $\mathbf{d} = 0$, which gives the mean class shape under the Gaussian prior of a VAE. Optimisation iteratively deforms the mean shape to best fit our observations. Figure 4.8 illustrates how changes in the shape descriptor alter the shape of the object.

## 4.5 Object-Level SLAM System

We have developed class level shape models and a measurement function that allows us to infer object shape and pose from a single RGB-D image. From stream of images we want to incrementally build a map of all the objects in a scene while simultaneously tracking the position of the camera. For this, we will show how to use the render module for camera tracking, and for joint optimisation of camera poses, object shapes, and object poses with respect to multiple image measurements. This will allow us to construct a full, incremental, jointly optimisable object-level SLAM system with sliding keyframe window optimisation.

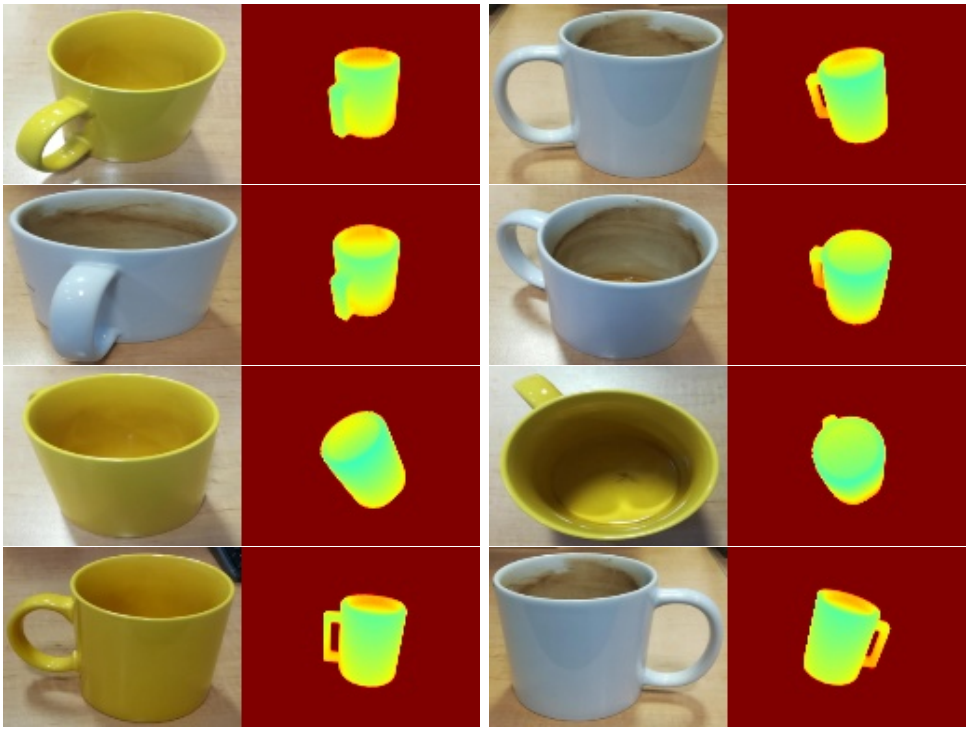


Figure 4.10: Results of pose prediction CNN (trained on a synthetic dataset) on real world data.

### 4.5.1 Data association and Object Initialisation

For each incoming image, we first segment and detect the classes of all objects in the image using Mask-RCNN [He et al., 2017]. For each detected object instance, we try to associate it with one of the objects already reconstructed in the map. This is done in a two stage process:

**Previous frame matching**: We match the masks in the image with masks from the previous frame. Two segmentations are considered a match if their IoU is above 0.2.

**Object mask rendering**: If a mask is not matched in stage 1, we try to match it directly with map objects by rendering their masks and computing IoU overlaps.

If a segmentation is not matched with any existing objects we initialise a new object as in Section 4.4.

### 4.5.2 Camera Tracking

We wish to track the camera pose $\mathbf{T}_{WC}^{j}$ for the latest depth measurement $\delta_j$. Once we have performed association between segmentation masks and reconstructed objects as described in Section 4.5.1, we have a list of matched object descriptors $\{\mathbf{d_1}, ..., \mathbf{d_N}\}$. We initialise our estimate for $\mathbf{T}_{WC}^{j}$ as the tracked pose of the previous frame $\mathbf{T}_{WC}^{j-1}$, and render the matched objects as described in Section 5.3.3:

$$\hat{\delta}_{\mu}, \hat{\delta}_{var} = \text{Render}(\{\hat{\mathbf{G}}_i\}, \{\mathbf{T}_{CG}^{i}\}, \mathbf{T}_{WC}^{j}) \ . \tag{4.9}$$

The loss between rendered and measured depth is:

$$L_{render}(\{\mathbf{d}_i\}, \{\mathbf{T}_{CG}^{i}\}, \mathbf{T}_{WC}^{j}) = \sum_{u,v} \frac{(\delta_j[u,v] - \hat{\delta}_{\mu}[u,v])^2}{\hat{\delta}_{var}[u,v]} \ . \tag{4.10}$$

Notice that this is the same loss used when inferring object pose and shape, but now we assume that the map (the object shapes and poses) is fixed and we want to estimate the camera pose $\mathbf{T}_{WC}^{j}$. As before, we use the iterative Gauss-Newton optimisation algorithm.

### 4.5.3 Sliding-Window Joint Optimisation

We have shown how to reconstruct objects from a single observation, and how to track the position of the camera by assuming the map is fixed. This will lead to the accumulation of errors, causing motion drift. Integrating new viewpoint observations for an object is also desirable, to improve its shape reconstruction. To tackle these two challenges, we wish to jointly optimise a bundle of camera poses, object poses, and object shapes. Doing this with all frames is however computationally infeasible, so we jointly optimise the variables associated to a select group of frames, called keyframes, in a sliding window manner, following the philosophy introduced by PTAM [Klein and Murray, 2007].

**Keyframe criteria:** There are two criteria for selecting a frame as a keyframe. If an object was initialised in the frame then it is selected as a keyframe, or second if the frame viewpoint for any of the existing objects is larger than 13 degrees from the frame in which the object was initialised.

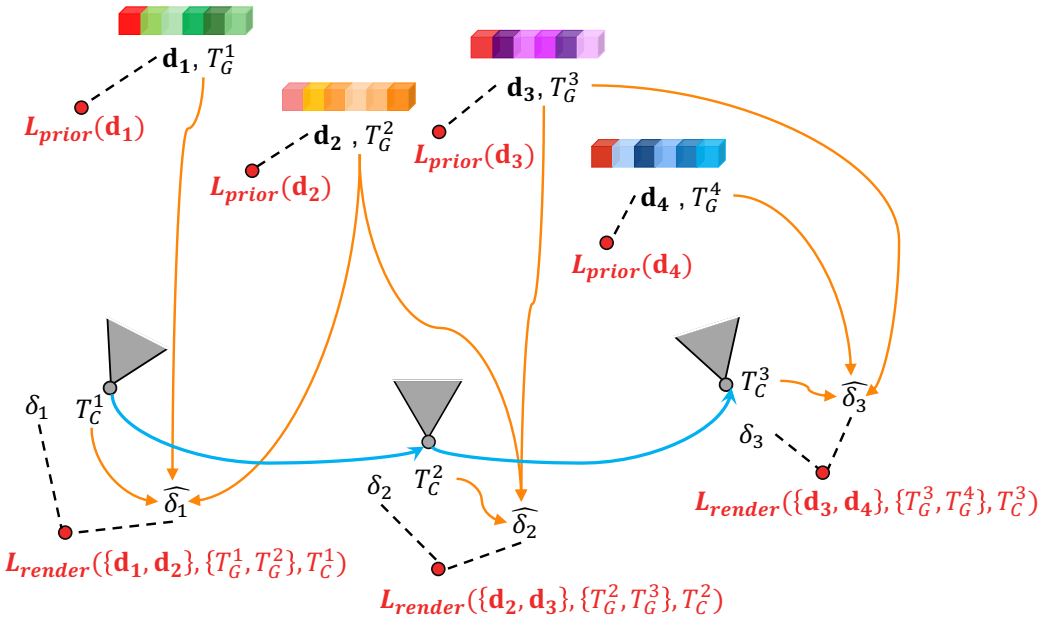



Figure 4.11: **Optimisation graph**, showing all jointly-optimised variables. Render and prior factors connect the different variables. A render factor compares object shape renders with depth measurements. Prior factors constrain how much each object shape can deviate from the mean shape of its class.

**Bundle Optimisation:** Each time that a frame is selected as a keyframe we jointly optimise the variables associated with a bundle of $N$ keyframes. In particular we select a window of 3 keyframes, the new keyframe and its two closest keyframes, with the previously defined distance.

To formulate the joint optimisation loss, consider, $\mathbf{T}_{WC}^1$, $\mathbf{T}_{WC}^2$, and $\mathbf{T}_{WC}^3$, the poses of the keyframes in the optimisation window; $\mathbf{T}_{WC}^1$ is held fixed. Now suppose $\{\mathbf{d}_i\}$ is the set of shape descriptors for the objects observed by the three keyframes. Then we can render a depth image and uncertainty for each keyframe as:

$$\hat{\delta}_{\mu}^{j}, \hat{\delta}_{var}^{j} = \text{Render}(\{\hat{\mathbf{G}}_i\}, \{\mathbf{T}_{GC}^{i}\}, \mathbf{T}_{WC}^{j}) \ , \tag{4.11}$$

with $\hat{\mathbf{G}}_i = D(\mathbf{d}_i, h_i)$. For each render we compute a loss with the respective depth measurement, $L_{render}^{j}$ as in Equation 4.10, and a prior loss, $L_{prior}^{i}$ on all codes as in Equation 4.8. Figure 4.11 illustrates the joint optimisation problem. Our final loss,

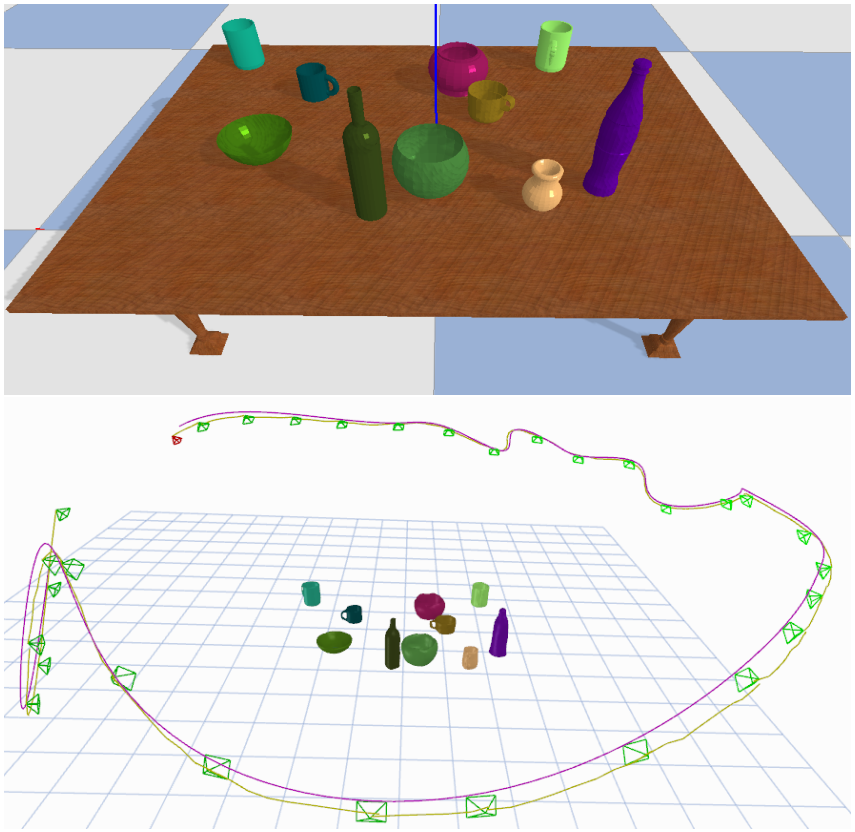


Figure 4.12: Synthetic scene example along with reconstruction and camera trajectory. Ground truth trajectory is shown in purple and tracked one in yellow, keyframes with green frustum.

optimised using Gauss-Newton, is:

$$L_{joint}(\{\mathbf{d}_i\}, \{\mathbf{T}^i_{GC}\}, \{\mathbf{T}^j_{WC}\}) = \\ \sum_j L_{render}(\{\mathbf{d}_i\}, \{\mathbf{T}^i_{GC}\}, \mathbf{T}^j_{WC}) + \sum_i L_{prior}(\mathbf{d}_i) \ . \tag{4.12}$$

**Timings:** Rendering a single object: 7ms; computing render jacobian: 100ms; object reconstruction: 1.5 seconds (15 iterations); camera tracking: 7fps; joint optimisation: 2 seconds (3 keyframe window).

Table 4.1: Shape reconstruction results for 1, 2, and 3 views. We do an ablation study of our method and compare with DVR [Niemeyer et al., 2020].

| | **Full** | **No Unc.** | **No Gauss.** | **[Niemeyer et al., 2020]** | **Mask** |
|---|---|---|---|---|---|
| **1 view** | | | | | |
| accuracy [$mm$] | 4.459 | 4.998 | 4.701 | 8.967 | 15.806 |
| chamfer-$L_1$ [$mm$] | 4.439 | 4.844 | 4.928 | 11.896 | 18.386 |
| completion [1cm] | 93.492 | 91.857 | 90.812 | 43.075 | 30.212 |
| **2 views** | | | | | |
| accuracy [$mm$] | 3.752 | 4.270 | 4.237 | 8.408 | 4.709 |
| chamfer-$L_1$ [$mm$] | 3.854 | 4.185 | 4.723 | 11.325 | 4.438 |
| completion [1cm] | 95.72 | 94.627 | 90.752 | 43.342 | 93.73 |
| **3 views** | | | | | |
| accuracy [$mm$] | 3.484 | 4.158 | 3.827 | 8.277 | 4.620 |
| chamfer-$L_1$ [$mm$] | 3.648 | 4.010 | 4.281 | 10.913 | 4.210 |
| completion [1cm] | 96.065 | 95.165 | 93 | 44.815 | 95.44 |

## 4.6 Experimental Results

### 4.6.1 Metrics

For shape reconstruction evaluation we use three metrics: chamfer-$L_1$ distance and accuracy as defined in [Mescheder et al., 2019b] and completeness (with 1cm threshold) as defined in [Li et al., 2020]. We sample 20000 points on both reconstruction and ground truth CAD model meshes.

### 4.6.2 Rendering Evaluation

In this evaluation we test the optimisation performance of our rendering formulation. We perform object shape and pose optimisation on all the objects of the 'mug' category in the ShapeNet dataset. For each instance we generate three random views of the object. Initial object pose is predicted from the first view. We perform 30 optimisation iterations for 1, 2, and 3 views. Table 4.1 shows median shape accuracy, completion, and chamfer distance after optimisation. We compare our full system with versions without uncertainty, without a Gaussian pyramid, and with a loss only between the rendered and Mask-RCNN segmentation masks. We compare with the state of the art volumetric differential rendering component in the paper Differential Volumetric Rendering (DVR) [Niemeyer et al., 2020] with our shape representation.

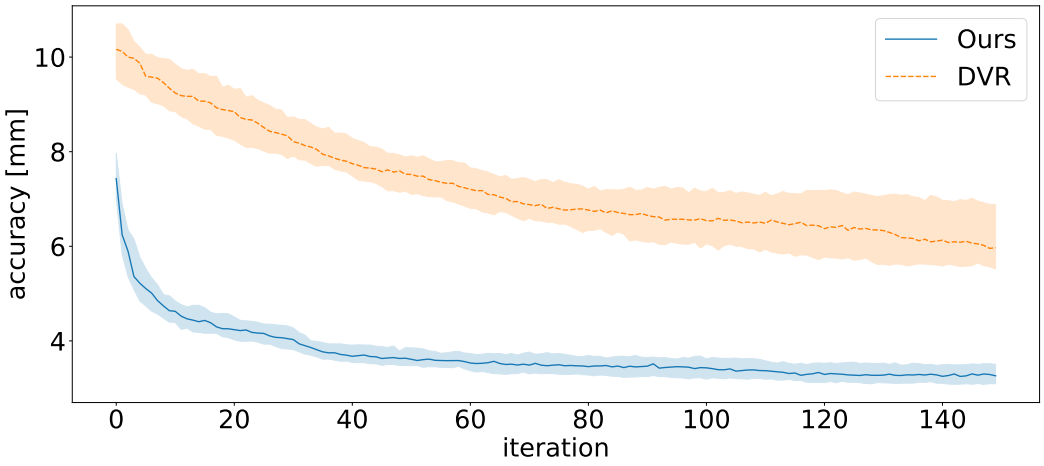



Figure 4.13: Median reconstruction accuracy (95% confidence) across 150 optimisation iterations of all 'mug' objects instances comparing our proposed renderer with [Niemeyer et al., 2020].

We observe that additional views improve shape reconstruction, more drastically in the mask optimisation because of the scale ambiguity in a single image. We also see that both the uncertainty and Gaussian pyramid are necessary for more accurate and complete shape reconstructions. Our method significantly improves on DVR, which is both less precise and has much lower shape completion, because of its local receptive field.

To further illustrate the comparison, we plot in Figure 4.13 median reconstruction accuracy across 150 optimisation iterations with all object instances against our proposed method. The plot illustrates the much faster convergence of our method and its ability to reach a lower error.

### 4.6.3 SLAM evauation

In this evaluation we evaluate our full SLAM system and how it generalises to new object instances. We create a synthetic dataset shown in Figure 4.12. Random object CAD models are spawned on top of a table model with random positions and vertical orientation. Five scenes are created with 10 different objects on each from three classes: 'mug', 'bowl', and 'bottle'. The models are obtained from the

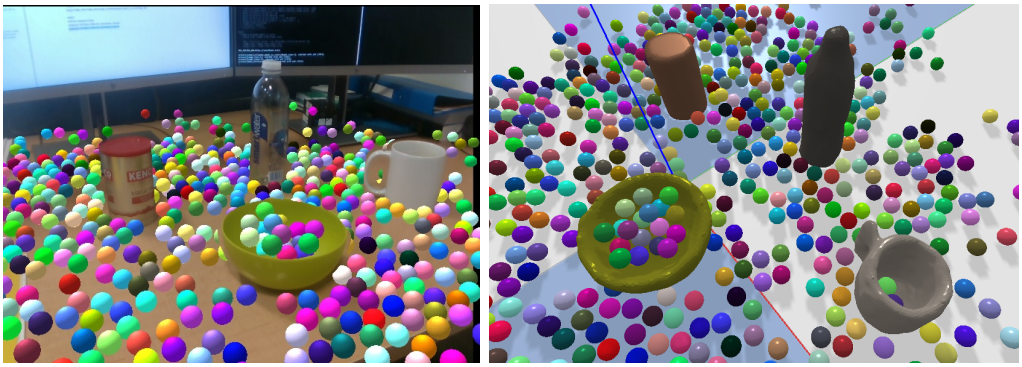


Figure 4.14: **Few-shot augmented reality:** Complete and watertight meshes can be obtained from few images due to the learned shape priors. This are then loaded into a physics engine to perform realistic augmented reality demonstrations.

ModelNet40 dataset [Wu et al., 2015] which are not used during training of the shape model.

For each scene a random trajectory is generated by sampling and interpolating random camera positions and look at points in the volume bounded by the table. Image and depth renders are obtained from the trajectory with PyBullet render, which is different rendering engine than the one used for training pose prediction.

### Fusion++ comparison

We compare our proposed method with a custom implementation of Fusion++ [McCormac et al., 2018] using open-source TSDF fusion [Zhou et al., 2018] for each object volume. In this experiment ground truth poses are used to decouple tracking accuracy and reconstruction quality. Gaussian noise is added to the depth image and camera poses (2mm, 1mm, 0.1° standard deviation for depth, translation and orientation, respectively).

We evaluate shape completion and accuracy; results are accumulated for each class from the 5 simulated sequences. Figure 4.17 shows how mean shape completion evolves with respect to frame number. This graph demonstrates the advantage of class-based priors for object shape reconstruction. With our method we see a jump to almost full completion, while TSDF fusion slowly completes the object with each

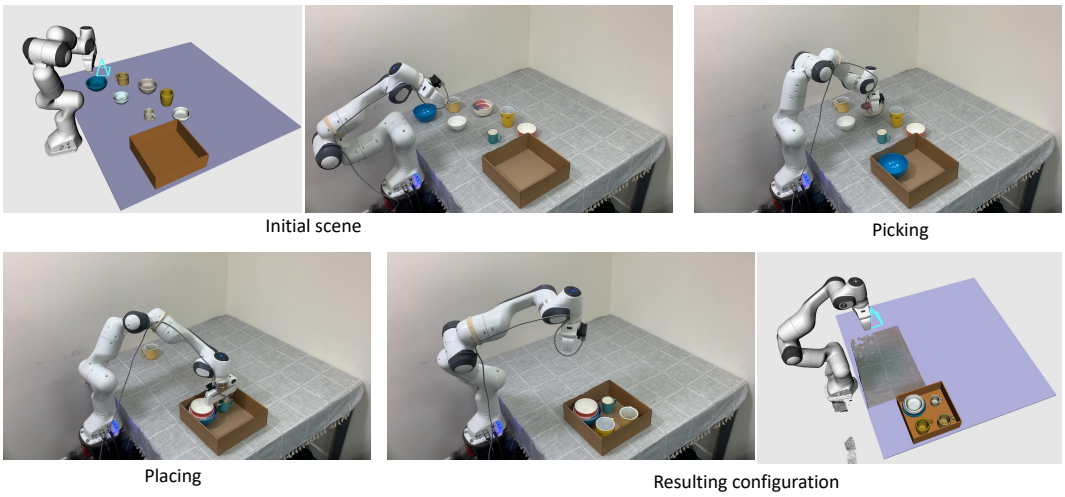



Figure 4.15: Robotic demonstration of packing of objects. The robot first captures pre-defined RGB-D images for scene reconstruction. The precise object models are used for grasp and placement planning, in this case for stacking bowls and mugs in a tight space.

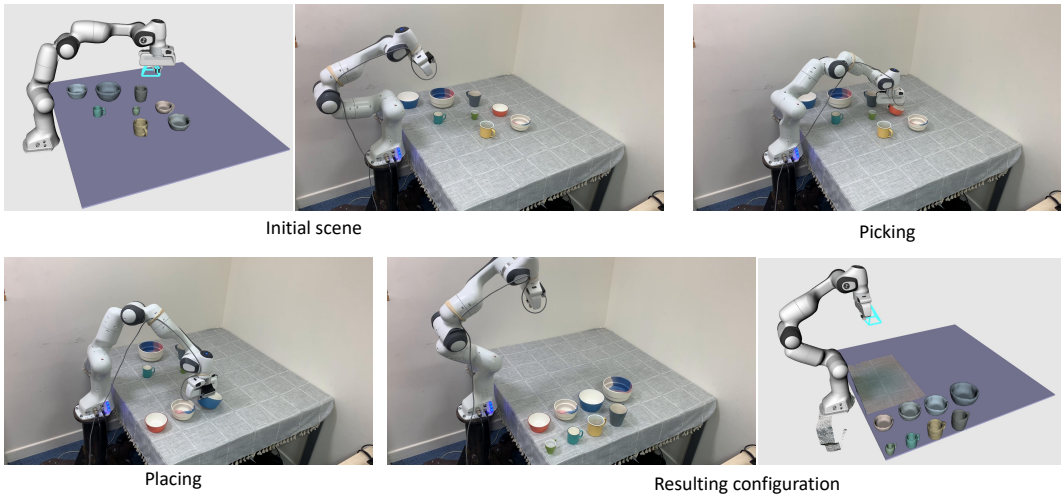



Figure 4.16: Robotic demonstration of packing sorting of objects. The robot first captures pre-defined RGB-D images for scene reconstruction. The precise object models are used for grasp and placement planning, in this case for sorting mugs and bowls according to size.

Table 4.2: Ablation study for tracking accuracy on 5 scenes, highlighting the importance of joint optimisation with uncertainty.

| **Absolute Pose Error [cm]** | Scene 1 | Scene 2 | Scene 3 | Scene 4 | Scene 5 |
|---|---|---|---|---|---|
| NodeSLAM | 1.73 | 1 | 0.81 | 1.24 | 1.15 |
| NodeSLAM no joint optim. | 8.6 | 10.17 | 0.7 | 2.14 | 1.25 |
| NodeSLAM no uncertainty | 4.37 | 3.41 | 0.88 | 3.05 | 6.99 |

new fused depth map. Fast shape completion without the need for exhaustive 360 degree scanning is important in robotic applications and in augmented reality, as shown in Figure 4.14. Figure 4.17 displays the median shape accuracy of NodeSLAM compared with TSDF fusion. We observe comparable surface reconstruction quality of close to 5mm.

### Ablation Study

We evaluate shape reconstruction accuracy and tracking *absolute pose error* on 3 different versions of our system. We compare our full SLAM system (with camera tracking) with a version without sliding window joint optimisation, and a version without uncertainty rendering. Figure 4.17 shows the importance of these features for shape reconstruction quality, with decreases in performance from 2 up to 7 mm. Table 4.2 shows mean *absolute pose error* for each version of our system for all 5 trajectories. These results prove that the precise shape reconstructions from objects provide enough information for accurate camera tracking with mean errors between 1 and 2 cm. It also shows how tracking without joint optimisation or uncertainty leads to significantly lower accuracy on most trajectories.

## 4.7 Robot Manipulation Application

We have developed a manipulation application which uses our object reconstruction system. We demonstrate two tasks: object packing, see Figure 4.15, and object sorting, see Figure 4.16. A rapid pre-defined motion is first used to gather a small

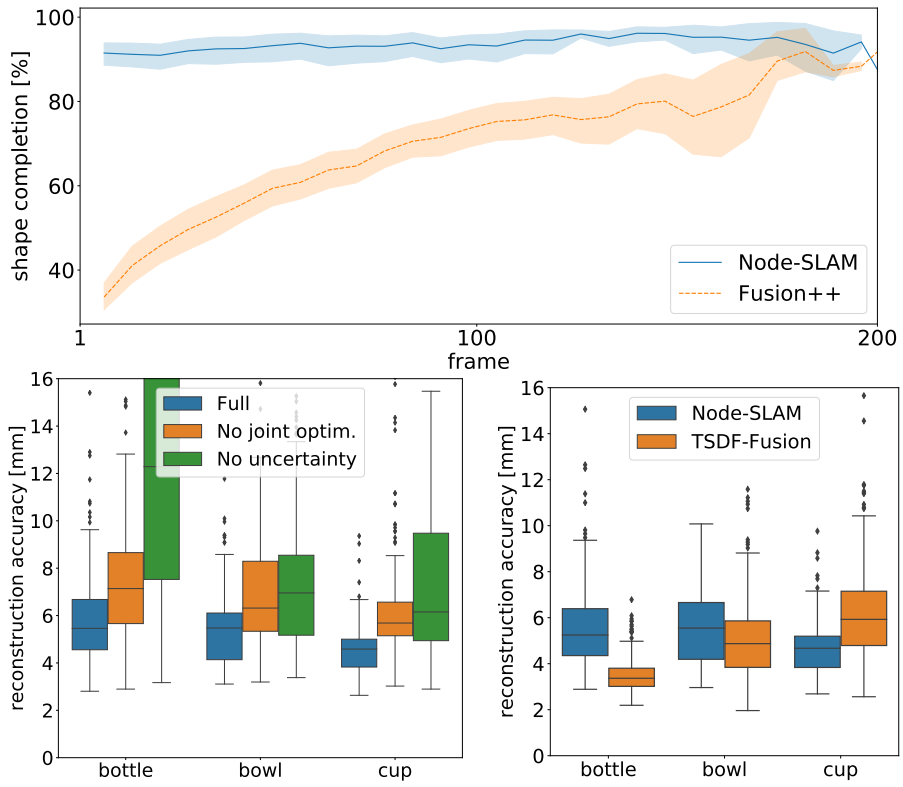



Figure 4.17: **Top:** Graph of mean object surface completion (95% confidence) comparison between NodeSLAM and TSDF fusion, with respect to the number of times an object is updated. **Bottom left:** Box plots of median surface reconstruction accuracy from our ablation study on 5 scenes with 10 objects in each. **Bottom right:** The same metric but comparing our system with Fusion++.

number of RGB-D views which our system uses to estimate the pose and shape of the objects laid out randomly on a table. Heuristics are used for grasp point selection and a placing motion based on the class and pose of the object and the shape of the reconstructed mesh. All the reconstructed objects are then sorted based on height and radius. For the packing task all the scanned objects are placed in a tight box, with bowls stacked in decreasing size order and all mugs placed inside the box with centers and orientations aligned. In the sorting task all objects are placed in a line in ascending size. In this robot application only, robot kinematics are used for camera tracking.

## 4.8 Conclusions

We have developed generative multi-class object models which allow for robust and principled multi-view shape reconstruction with the integration of semantic priors, in a subset of common tabletop object classes. This is accomplished by integrating a learned compact latent space for objects shapes with a volumetric differntiable rendering function. We demonstrated their practical use in an object-level SLAM system as well as in two robotic manipulation demonstrations and an augmented reality demo. We believe this shows evidence that semantic priors are a strong prior for complete and precise shape reconstruction, and that decomposing a scene into full object entities is a useful idea smart interaction. While our models go beyond rigid templates such as pre-defined CAD models and allow certain variation within a known semantic class, not all object classes will be well represented by the single code object VAE we used in this paper. In Chapter 5 we look at how to go beyond objects and use a neural network to represent whole scenes, and in Chapter 6 we look at how to extract an object decomposition from a compressed scene representation.

Chapter 5

# iMAP: Neural Fields for Dense SLAM

## Contents

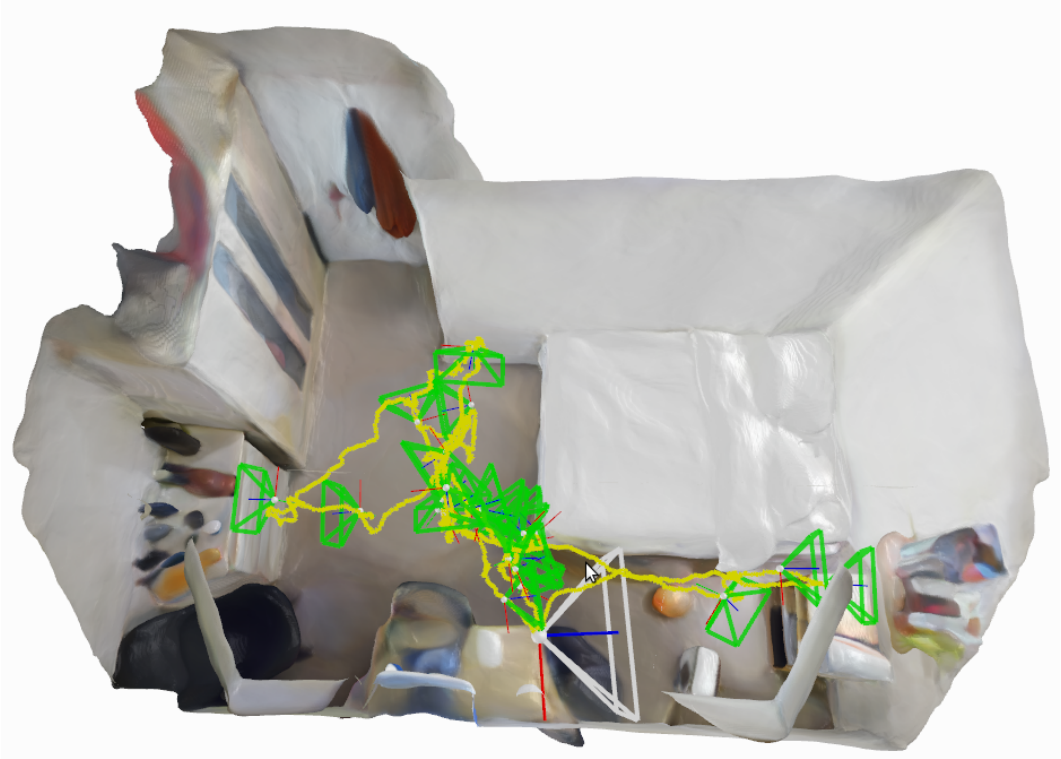


Figure 5.1: Room reconstruction from real-time iMAP with an Azure Kinect RGB-D camera, showing watertight scene model, camera tracking and automatic keyframe set.

## 5.1 Introduction

A real-time Simultaneous Localisation and Mapping (SLAM) system for an intelligent embodied device must incrementally build a representation of the 3D world, to enable both localisation and scene understanding. The ideal representation should precisely encode geometry, but also be *efficient*, with the memory capacity available used adaptively in response to scene size and complexity; *predictive*, able to plausibly estimate the shape of regions not directly observed; and *flexible*, not needing a large amount of training data or manual adjustment to run in a new scenario.

Implicit neural representations are a promising recent advance in off-line reconstruction, using a multilayer perceptron (MLP) to map a query 3D point to occupancy or colour, and optimising it from scratch to fit a specific scene. An MLP is

a general implicit function approximator, able to represent variable detail with few parameters and without quantisation artifacts. Even without prior training, the inherent priors present in the network structure allow it to make watertight geometry estimates from partial data, and plausible completion of unobserved regions.

In this chapter, we show for the first time that an MLP can be used as the only scene representation in a real-time SLAM system using a hand-held RGB-D camera. Our randomly-initialised network is trained in *live operation* and we do not require any prior training data. Our iMAP system is designed with a keyframe structure and multi-processing computation flow reminiscent of PTAM [Klein and Murray, 2007]. In a tracking process, running at over 10 Hz, we align live RGB-D observations with rendered depth and colour predictions from the MLP scene map. In parallel, a mapping process selects and maintains a set of historic keyframes whose viewpoints span the scene, and uses these to continually train and improve the MLP, while jointly optimising the keyframe poses.

In both tracking and mapping, we dynamically sample the most informative RGB-D pixels to reduce geometric uncertainty, achieving real-time speed. Our system runs in Python, and all optimisation is via a standard PyTorch framework [Paszke et al., 2019] on a single desktop CPU/GPU system.

By casting SLAM as a continual learning problem, we achieve a representation which can represent scenes efficiently with continuous and adaptive resolution, and with a remarkable ability to smoothly interpolate to achieve complete, watertight reconstruction (Figure 5.1). With around 10 - 20 keyframes, and an MLP with only 1 MB of parameters, we can accurately map whole rooms. Our scene representation has no fixed resolution; the distribution of keyframes automatically achieves efficient multi-scale mapping.

We demonstrate our system on a wide variety of real-world sequences and do exhaustive evaluation and ablative analysis on 8 scenes from the room-scale Replica Dataset [Straub et al., 2019]. We show that iMAP can make a more **complete scene reconstruction** than standard dense SLAM systems with **significantly**

**smaller memory footprint**. We show competitive tracking performance on the TUM RGB-D dataset [Sturm et al., 2012] against state-of-the-art SLAM systems.

To summarise, the key contributions of this work are:

- The first dense real-time SLAM system that uses an implicit neural scene representation and is capable of jointly optimising a full 3D map and camera poses.
- The ability to *incrementally* train an implicit scene network in real-time, enabled by automated keyframe selection and loss guided sparse active sampling.
- A parallel implementation (fully in PyTorch [Paszke et al., 2019] with multi-processing) of our presented SLAM formulation which works online with a hand-held RGB-D camera.

## 5.2 Related Work

**Visual SLAM Systems** Real-time visual SLAM systems for modelling environments are often built in a layered manner, where a sparse representation is used for localisation and more detailed geometry or semantics is layered on top. However, here we work in the 'dense SLAM' paradigm pioneered in [Newcombe et al., 2011b, Newcombe et al., 2011a] where a *unified* dense scene representation is also the basis for camera tracking. Dense representations avoid arbitrary abstractions such as keypoints, enable tracking and relocalisation in robust invariant ways, and have long-term appeal as sensor-agnostic, unified, complete representations of spaces.

Some approaches in dense SLAM explicitly represent surfaces [Keller et al., 2013, Whelan et al., 2015], but direct representation of volume is desirable to enable a full range of applications such as planning. Standard representations for volume using occupancy or signed distance functions are very expensive in terms of memory if a fixed resolution is used [Newcombe et al., 2011a]. Hierarchical approaches [Dai et al., 2017b, Vespa et al., 2018] are more efficient, but are complicated to implement and

usually offer only a small range of level of detail. In either case, the representations are rather rigid, and not amenable to joint optimisation with camera poses, due to the huge number of parameters they use.

Machine learning can discover low-dimensional embeddings of dense structure which enable efficient, jointly optimisable representation. CodeSLAM [Bloesch et al., 2018] is one example, but using a depth-map view representation rather than full volumetric 3D. Learning techniques have also been used to improve dense reconstruction but require an existing scan [Dai et al., 2020] or previous training data [Peng et al., 2020, Weder et al., 2020, Chabra et al., 2020].

**Implicit Scene Representation with MLPs** Scene representation and graphics have seen much recent progress on using implicit MLP neural models for object reconstruction [Park et al., 2019, Mescheder et al., 2019a], object compression [Tang et al., 2020] novel view synthesis [Mildenhall et al., 2020b], and scene completion [Sitzmann et al., 2020, Chibane et al., 2020]. Two recent papers [Wang et al., 2021b, Yen-Chen et al., 2020] have also explored camera pose optimisation. But so far these methods have been considered as an offline tool, with computational requirements on the order of hours, days or weeks. We show that when depth images are available, and when guided sparse sampling is used for rendering and training, these methods are suitable for real-time SLAM.

**Continual Learning** By using a single MLP as a master scene model, we pose real-time SLAM as *online continual learning*. An effective continual learning system should demonstrate both plasticity (the ability to acquire new knowledge) and stability (preserving old knowledge) [Rolnick et al., 2019, Grossberg, 1982]. Catastrophic forgetting is a well-known property of neural networks, and is a failure of stability, where new experiences overwrite memories.

One line of work on alleviating catastrophic forgetting has focused on protecting representations against new data using relative weighting [Kirkpatrick et al., 2017]. This is reminiscent of classic filtering approaches in SLAM such as the EKF [Smith

and Cheeseman, 1986] and is worth future investigation. Approaches which freeze [Rusu et al., 2016] or consolidate [Schwarz et al., 2018] sub-networks after training on each individual task are perhaps too simple and discrete for SLAM.

Instead, we direct our attention towards the *replay*-based approach to continual learning, where previous knowledge is stored either directly in a buffer [Maltoni and Lomonaco, 2019, Rolnick et al., 2019], or compressed in a generative model [Lesort et al., 2019, Shin et al., 2017]. We use a straightforward method where keyframes are automatically selected to store and compress past memories. We use loss-guided random sampling of these keyframes in our continually running map update process to periodically replay and strengthen previously-observed scene regions, while continuing to add information via new keyframes. In SLAM terms, this approach is similar to that pioneered by PTAM [Klein and Murray, 2007], where a historic keyframe set and repeated global bundle adjustment serve as a long-term scene representation.

## 5.3 iMAP: A Real-Time Implicit SLAM System

### 5.3.1 System Overview

Figure 5.2 overviews how iMAP works. A 3D volumetric map is represented using a fully-connected neural network $F_\theta$ that maps a 3D coordinate to colour and volume density (Section 5.3.2). Given a camera pose, we can render the colour and depth of a pixel by accumulating network queries from samples in a back-projected ray (Section 5.3.3).

We map a scene from depth and colour video by incrementally optimising the network weights and camera poses with respect to a sparse set of actively sampled measurements (Section 5.3.6). Two processes run concurrently: *tracking* (Section 5.3.4), which optimises the pose from the current frame with respect to the locked network; and *mapping* (Section 5.3.4), which jointly optimises the network and the camera poses of selected keyframes, incrementally chosen based on information gain

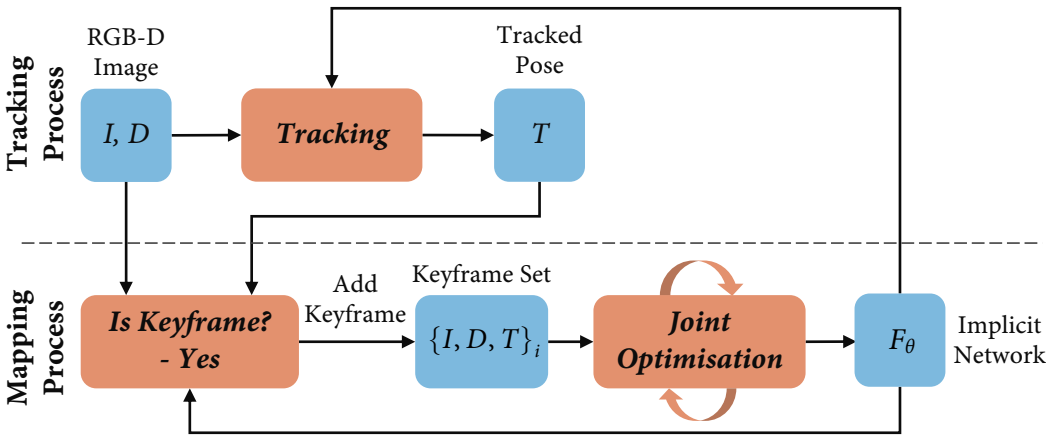



Figure 5.2: iMAP system pipeline.

(Section 5.3.5).

### 5.3.2 Implicit Scene Neural Network

Following the network architecture in NeRF [Mildenhall et al., 2020b], we use an MLP (Section 3.6.1) with 4 hidden layers of feature size 256, and two output heads that map a 3D coordinate $\mathbf{p} = (x, y, z)$ to a colour and volume density value: $F_\theta(\mathbf{p}) = (\mathbf{c}, \rho)$. Unlike NeRF, we do not take into account viewing directions as we are not interested in modelling specularities.

We apply the Gaussian positional embedding, described in Section 3.8.3, proposed in Fourier Feature Networks [Tancik et al., 2020] to lift the input 3D coordinate into $n$-dimensional space: $\sin(\mathbf{Bp})$, with $\mathbf{B}$ an $[n \times 3]$ matrix sampled from a normal distribution with standard deviation $\sigma$. This embedding serves as input to the MLP and is also concatenated to the second activation layer of the network. Taking inspiration from SIREN [Sitzmann et al., 2020], we allow optimisation of the embedding matrix $\mathbf{B}$, implemented as a single fully-connected layer with sine activation.

### 5.3.3 Depth and Colour Rendering

Our differentiable rendering engine, inspired by NeRF [Mildenhall et al., 2020b] and NodeSLAM (Chapter 4), queries the scene network to obtain depth and colour

images from a given view. See Section **??** for more details on differential volumetric rendering.

Given a camera pose $\mathbf{T}_{WC}$ and a pixel coordinate $[u, v]$, we first back-project a normalised viewing direction and transform it into world coordinates: $\mathbf{r} = \mathbf{T}_{WC}\mathbf{K}^{-1}[u, v]$, with the camera intrinsics matrix $\mathbf{K}$. We take a set of $N$ samples along the ray $\mathbf{p}_i = d_i\mathbf{r}$ with corresponding depth values $\{d_1, \cdots, d_N\}$, and query the network for a colour and volume density $(\mathbf{c}_i, \rho_i) = F_\theta(\mathbf{p}_i)$. We follow the stratified and hierarchical volume sampling strategies of NeRF.

Volume density is transformed into an occupancy probability by multiplying by the inter-sample distance $\delta_i = d_{i+1} - d_i$ and passing this through activation function $o_i = 1 - \exp(-\rho_i\delta_i)$. The ray termination probability at each sample can then be calculated as $w_i = o_i \prod_{j=1}^{i-1}(1 - o_j)$. Finally, depth and colour are rendered as the expectations:

$$\hat{D}[u, v] = \sum_{i=1}^{N} w_i d_i, \quad \hat{I}[u, v] = \sum_{i=1}^{N} w_i \mathbf{c}_i. \tag{5.1}$$

We can calculate the depth variance along the ray as:

$$\hat{D}_{var}[u, v] = \sum_{i=1}^{N} w_i (\hat{D}[u, v] - d_i)^2. \tag{5.2}$$

### 5.3.4 Joint optimisation

We jointly optimise the implicit scene network parameters $\theta$, and camera poses for a growing set of $W$ keyframes, each of which has associated colour and depth measurements along with an initial pose estimate: $\{I_i, D_i, \mathbf{T}^i_{WC}\}$. Figure 5.3 shows an schematic of the joint optimisation.

Our rendering function is differentiable with respect to these variables, so we perform iterative optimisation to minimise the geometric and photometric errors for a selected number of rendered pixels $s_i$ in each keyframe.

The photometric loss is the L1-norm between the rendered and measured colour

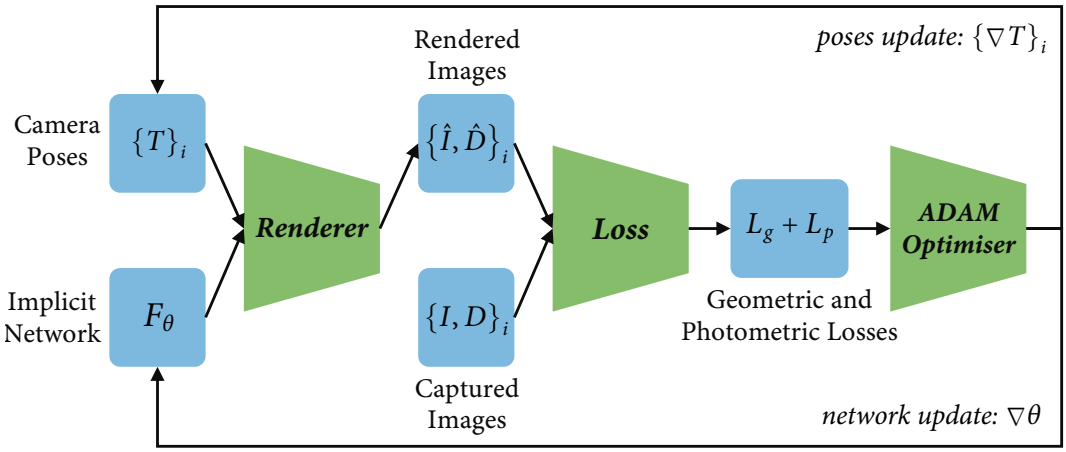



Figure 5.3: Joint optimisation. Keyframe camera poses and the implicit network are jointly optimised with ADAM through photometric and geometric losses calculated by rendering a sparse set of color and depth pixels.

values $e_i^p[u,v] = \left|I_i[u,v] - \hat{I}_i[u,v]\right|$ for $M$ pixel samples:

$$L_p = \frac{1}{M}\sum_{i=1}^{W}\sum_{(u,v)\in s_i} e_i^p[u,v]. \tag{5.3}$$

The geometric loss measures the depth difference $e_i^g[u,v] = \left|D_i[u,v] - \hat{D}_i[u,v]\right|$ and uses the depth variance as a normalisation factor, down-weighting the loss in uncertain regions such as object borders:

$$L_g = \frac{1}{M}\sum_{i=1}^{W}\sum_{(u,v)\in s_i} \frac{e_i^g[u,v]}{\sqrt{\hat{D}_{var}[u,v]}}. \tag{5.4}$$

We apply the ADAM optimiser [Kingma and Ba, 2015a] on the weighted sum of both losses, with factor $\lambda_p$ adjusting the importance given to the photometric error:

$$\min_{\theta,\{\mathbf{T}_{WC}^i\}} \left(L_g + \lambda_p L_p\right). \tag{5.5}$$

In this work we use a first order optimisation based in gradient descent (Section 3.5) in contrast to a second order optimiser as in NodeSLAM (Chapter 4) because of the MLP map representation.

**Camera Tracking** In online SLAM, close to frame-rate camera tracking is important, as optimisation of smaller displacements is more robust. We run a parallel

tracking process that continuously optimises the pose of the latest frame with respect to the fixed scene network at a much higher frame rate than joint optimisation while using the same loss and optimiser. The tracked pose initialisation is refined in the mapping process for selected keyframes.

### 5.3.5 Keyframe Selection

Jointly optimising the network parameters and camera poses using all images from a video stream is not computationally feasible. However, since there is huge redundancy in video images, we may represent a scene with a sparse set of representative keyframes, incrementally selected based on *information gain*. The first frame is always selected to initialise the network and fix the world coordinate frame. Every time a new keyframe is added, we lock a copy of our network to represent a snapshot of our 3D map at that point in time. Subsequent frames are checked against this copy and are selected if they see a significantly new region.

For this, we render a uniform set of pixel samples $s$ and calculate the proportion $P$ with a normalised depth error smaller than threshold $t_D = 0.1$, to measure the fraction of the frame already explained by our map snapshot:

$$P = \frac{1}{|s|} \sum_{(u,v)\in s} \mathbb{1}\left( \frac{\left| D[u,v] - \hat{D}[u,v] \right|}{D[u,v]} < t_D \right). \tag{5.6}$$

When this proportion falls under a threshold $P < t_P$ (we set $t_P = 0.65$), this frame is added to the keyframe set. The normalised depth error produces adaptive keyframe selection, requiring higher precision, and therefore more closely spaced keyframes, when the camera is closer to objects.

Every frame received in the mapping process is used in joint optimisation for a few iterations (between 10 and 20), so our keyframe set is always composed of the selected set along with the continuously changing latest frame.

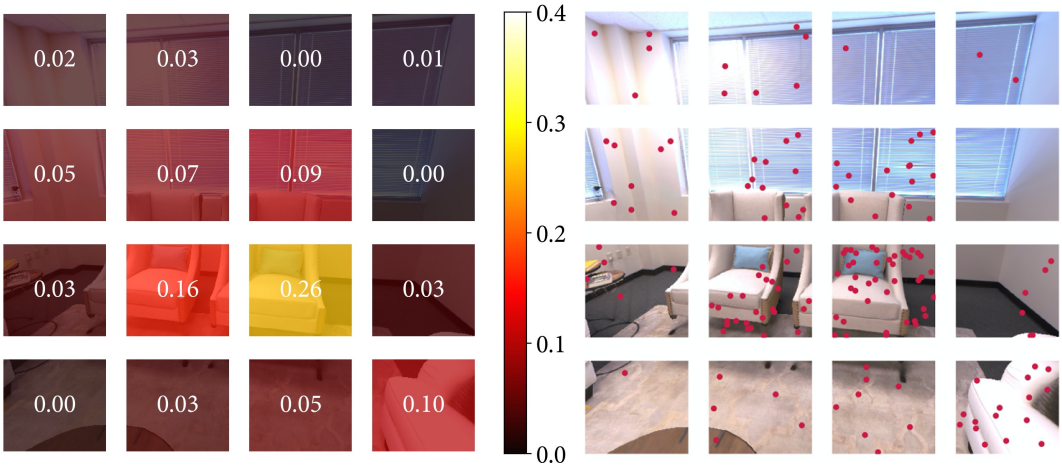



Figure 5.4: Image Active Sampling. Left: a loss distribution is calculated across an image grid using the geometric loss from a set of uniform samples. Right: active samples are further allocated proportional to the loss distribution.

### 5.3.6 Active Sampling

**Image Active Sampling** Rendering and optimising all image pixels would be expensive in computation and memory. We take advantage of image regularity to render and optimise only a very sparse set of random pixels (200 per image) at each iteration. Further, we use the render loss to guide active sampling in informative areas with higher detail or where reconstruction is not yet precise.

Each joint optimisation iteration is divided into two stages. First, we sample a set $s_i$ of pixels, uniformly distributed across each of the keyframe's depth and colour images. These pixels are used to update the network and camera poses, and to calculate the loss statistics. For this, we divide each image into an $[8 \times 8]$ grid, and calculate the average loss inside each square region $R_j$, $j = \{1, 2, \cdots, 64\}$:

$$L_i[j] = \frac{1}{|r_j|} \sum_{(u,v) \in r_j} e_i^g[u,v] + e_i^p[u,v], \tag{5.7}$$

where $r_j = s_i \cap R_j$ are pixels uniformly sampled from $R_j$. We normalise these statistics into a probability distribution:

$$f_i[j] = \frac{L_i[j]}{\sum_{m=1}^{64} L_i[m]}. \tag{5.8}$$

We use this distribution to re-sample a new set of $n_i \cdot f_i[j]$ uniform samples per region ($n_i$ is the total samples in each keyframe), allocating more samples to regions with high loss. The scene network is updated with the loss from active samples (in camera tracking only uniform sampling is used). Image active sampling is illustrated in Figure 5.4.

**Keyframe Active Sampling** In iMAP, we continuously optimise our scene map with a set of selected keyframes, serving as a memory bank to avoid network forgetting. We wish to allocate more samples to keyframes with a higher loss, because they relate to regions which are newly explored, highly detailed, or that the network started to forget. We follow a process analogous to image active sampling and allocate $n_i$ samples to each keyframe, proportional to the loss distribution across keyframes, See Figure 5.5.

**Bounded Keyframe Selection** Our keyframe set keeps growing as the camera moves to new and unexplored regions. To bound joint optimisation computation, we choose a fixed number (3 in the live system) of keyframes at each iteration, randomly sampled according to the loss distribution. We always include the last keyframe and the current live frame in joint optimisation, to compose a bounded window with $W = 5$ constantly changing frames. See Figure 5.5.

## 5.4 Experimental Results

Through comprehensive experiments we evaluate iMAP's 3D reconstruction and tracking, and conduct a detailed ablative analysis of design choices on accuracy and speed.

### 5.4.1 Experimental Setup

**Datasets** We experiment on both simulated and real sequences. For reconstruction evaluation we use the Replica dataset [Straub et al., 2019], high quality 3D reconstructions of real room-scale environments, with 5 offices and 3 apartments.

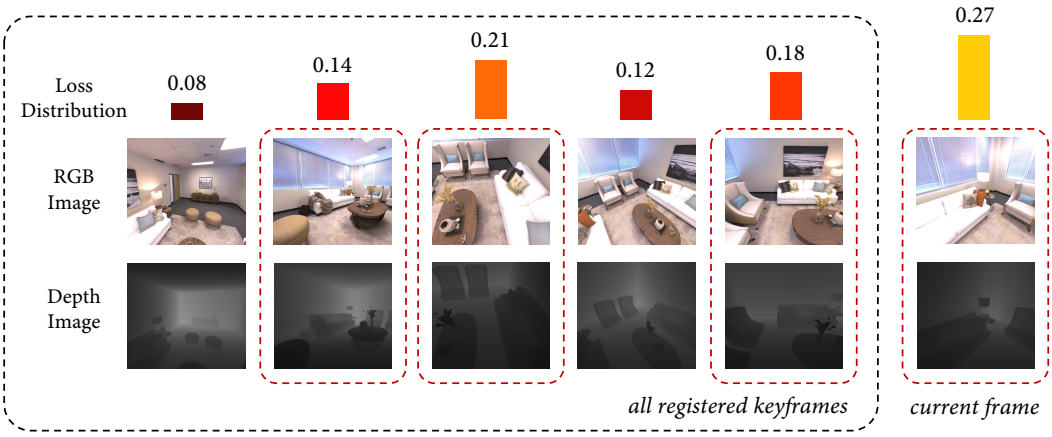



Figure 5.5: Keyframe Active Sampling. We maintain a loss distribution over the registered keyframes. The distribution is used for sampling a bounded window of keyframes (red boxes), and for allocating pixel samples in each.

Figure 5.6: Reconstruction and tracking results for Replica `room-0` along with registered keyframes.

For each Replica scene, we render a random trajectory of 2000 RGB-D frames. For raw camera recordings, we capture RGB-D videos using a hand-held Microsoft Azure Kinect on a wide variety of environments, as well as test on the TUM RGB-D dataset [Sturm et al., 2012] to evaluate camera tracking.

Figure 5.7: iMAP (left) manages to fill in unobserved regions which can be seen as holes in TSDF fusion (right).

**Implementation Details** For all experiments we set the following default parameters: keyframe registration threshold $t_P = 0.65$, photo-metric loss weighting $\lambda_p = 5$, keyframe window size $W = 5$, pixel samples $|s_i| = 200$, positional embedding size $m = 93$ and sigma $\sigma = 25$, and 32 coarse and 12 fine bins for rendering. 3D point coordinates are normalised by $\frac{1}{10}$ to be close to the $[0, 1]$ range.

In online operation from a hand-held camera, streamed images which arrive between processed frames are dropped. For the experiments presented here every captured frame is processed, running at 10 Hz. We recover mesh reconstructions if needed by querying occupancy values from the network in a uniform voxel grid and then running marching cubes. Meshing is for visualisation and evaluation purposes and does not form part of our SLAM system.

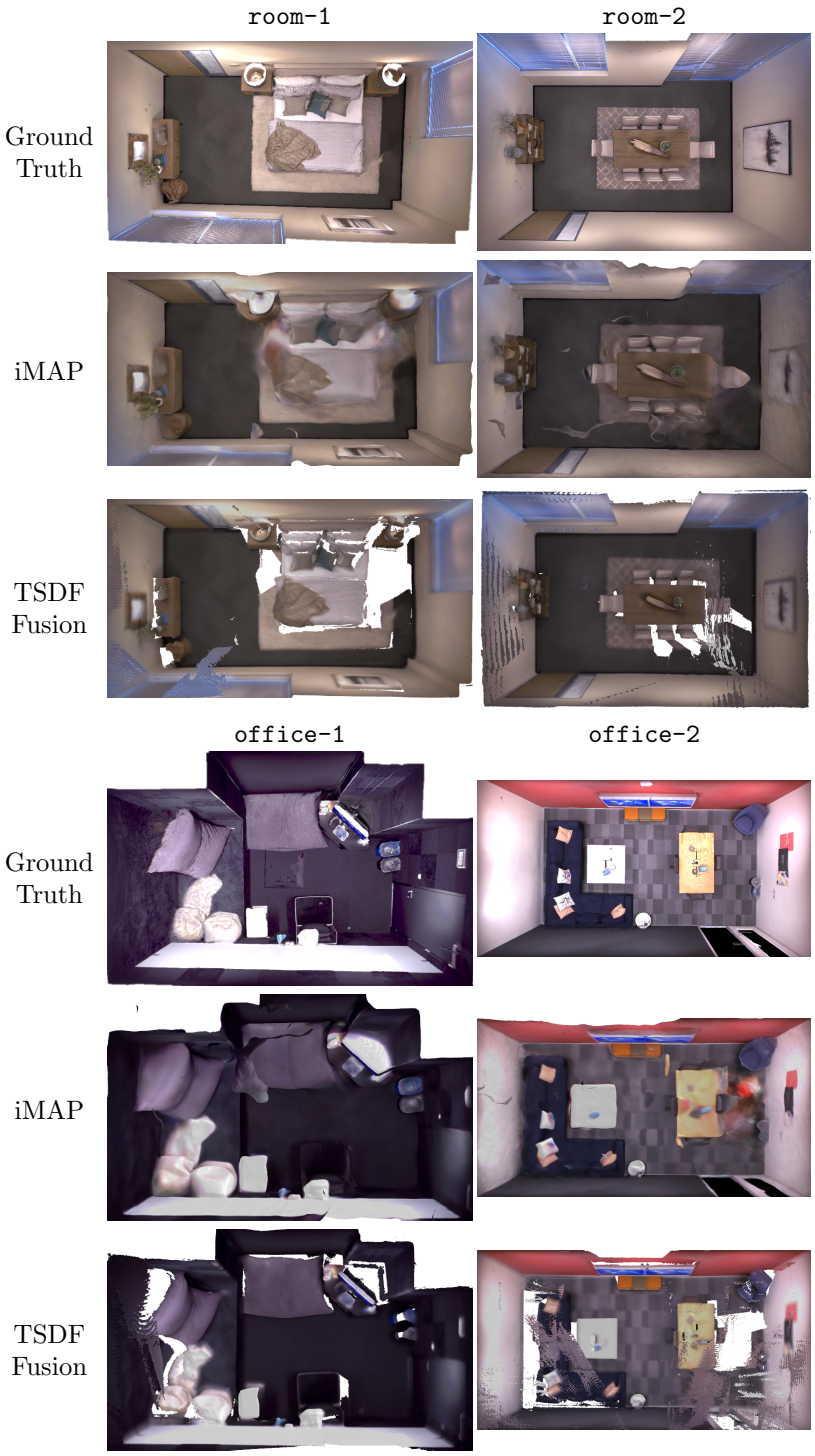



Figure 5.8: Replica reconstructions, highlighting how iMAP fills in unobserved regions which are white holes in TSDF fusion.

### 5.4.2 Scene Reconstruction Evaluation

**Metrics** We sample 200,000 points from both ground-truth and reconstructed meshes, and calculate three quantitative metrics: *Accuracy* (cm): the average distance between sampled points from the reconstructed mesh and the nearest ground-truth point; *Completion* (cm): the average distance between sampled points from the ground-truth mesh and the nearest reconstructed; and *Completion Ratio* (<5cm %): the percentage of points in the reconstructed mesh with *Completion* under 5 cm.

The ability to jointly optimise a 3D map along with camera poses gives our system the capacity to build full globally coherent scene reconstructions as seen in Figure 5.1 and 5.8, and accurate camera tracking as shown in Figure 5.6. The robustness and versatility of iMAP is demonstrated on a wide variety of real world recordings, through the reconstructions in Figures 5.11, 5.12, 5.9, and 5.10 that show its ability to work at scales from whole rooms to small objects and thin structures.

We compare scene reconstructions from iMAP with TSDF fusion [Curless and Levoy, 1996, Newcombe et al., 2011a], which is representative of fusion-based dense SLAM methods. To isolate reconstruction, we use the camera tracking produced by iMAP for TSDF fusion. The most significant advantage of our implicit representation is the ability to fill in unobserved regions as shown in Figs. 5.8 and 5.10. iMAP achieves on average a 4% higher completion ratio across all 8 Replica scenes as seen in Table 5.1, with an improvement of 11% in `office-3`.

Memory consumption for iMAP and TSDF fusion with different configuration settings is shown in Table 5.2. With default values of $256^3$ voxel resolution in TSDF fusion and 256 network width in iMAP, our system can represent scenes with a factor of 60 less memory usage while obtaining similar reconstruction accuracy as seen in Table 5.1.

When using a real camera, in addition to better completion our method outperforms TSDF fusion in places where a depth camera does not give accurate readings

| | | room0 | room1 | room2 | office0 | office1 | office2 | office3 | office4 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| iMAP | **Keyframes** | 11 | 12 | 12 | 10 | 11 | 10 | 14 | 11 | 13.37 |
| | **Acc.** [cm] | 3.58 | 3.69 | 4.68 | 5.87 | 3.71 | 4.81 | 4.27 | 4.83 | 4.43 |
| | **Comp.** [cm] | 5.06 | 4.87 | 5.51 | 6.11 | 5.26 | 5.65 | 5.45 | 6.59 | **5.56** |
| | **Comp.Ratio** [< 5cm %] | 83.91 | 83.45 | 75.53 | 77.71 | 79.64 | 77.22 | 77.34 | 77.63 | **79.06** **79.06** |
| TSDF Fusion | **Acc.** [cm] | 4.21 | 3.08 | 2.88 | 2.70 | 2.66 | 4.27 | 4.07 | 3.70 | **3.45** |
| | **Comp.** [cm] | 5.04 | 4.35 | 5.40 | 10.47 | 10.29 | 6.43 | 6.26 | 4.78 | 6.63 |
| | **Comp.Ratio** [< 5cm %] | 76.90 | 79.87 | 77.79 | 79.60 | 71.93 | 71.66 | 65.87 | 77.11 | 75.09 |

Table 5.1: Reconstruction results for 8 indoor Replica scenes. We report the highest reached completion ratio in each scene along with the corresponding accuracy and completion values at that point.


Figure 5.9: Comparative reconstruction results in various real scenes mapped with an Azure Kinect. White holes in the TDSF fusion results are plausibly filled in by iMAP, such as in black objects where a depth camera has missing data.

| | | | |
|---|---|---|---|
| **iMAP** [MB] | Width = 128 | Width = 256 | Width = 512 |
| | 0.26 | 1.04 | 4.19 |
| **TSDF Fusion** [MB] | Res. = 128 | Res. = 256 | Res. = 512 |
| | 8.38 | 67.10 | 536.87 |

Table 5.2: Memory consumption: for iMAP as a function of network size, and for TSDF fusion of voxel resolution.


Figure 5.10: Comparative reconstruction results in various real scenes mapped with an Azure Kinect. iMAP can better represent thin structures and interpolate the back of objects.

Figure 5.11: Real-time reconstruction results from iMAP in a variety of indoor settings.

Figure 5.12: Real-time reconstruction results from iMAP in a variety of outdoor settings.

| | fr1/desk (cm) | fr2/xyz (cm) | fr3/office (cm) |
|---|---|---|---|
| **iMAP** | 4.9 | 2.0 | 5.8 |
| **BAD-SLAM** | 1.7 | 1.1 | 1.73 |
| **Kintinuous** | 3.7 | 2.9 | 3.0 |
| **ORB-SLAM2** | 1.6 | 0.4 | 1.0 |

Table 5.3: ATE RMSE in cm on TUM RGB-D dataset.

as is common for black objects (Figure 5.9), and reflective or transparent surfaces (Figure 5.7). This performance can be attributed to the photometric loss for reconstruction combined with the interpolation capacity of the map network.

### 5.4.3 TUM Evaluation

We run iMAP on three sequences from TUM RGB-D. Tracking ATE RMSE is shown in Table 5.3. We compare with surfel-based BAD-SLAM [**?**], TSDF fusion Kintinuous [Whelan et al., 2012], and sparse ORB-SLAM2 [Mur-Artal and Tardós, 2017], state-of-the-art SLAM systems. In pose accuracy, iMAP does not outperform them, but is competitive with errors between 2 and 6 cm. Mesh reconstructions are shown in Figure 5.13. In Figure 5.14 we highlight how iMAP fills in holes in unobserved regions unlike BAD-SLAM.

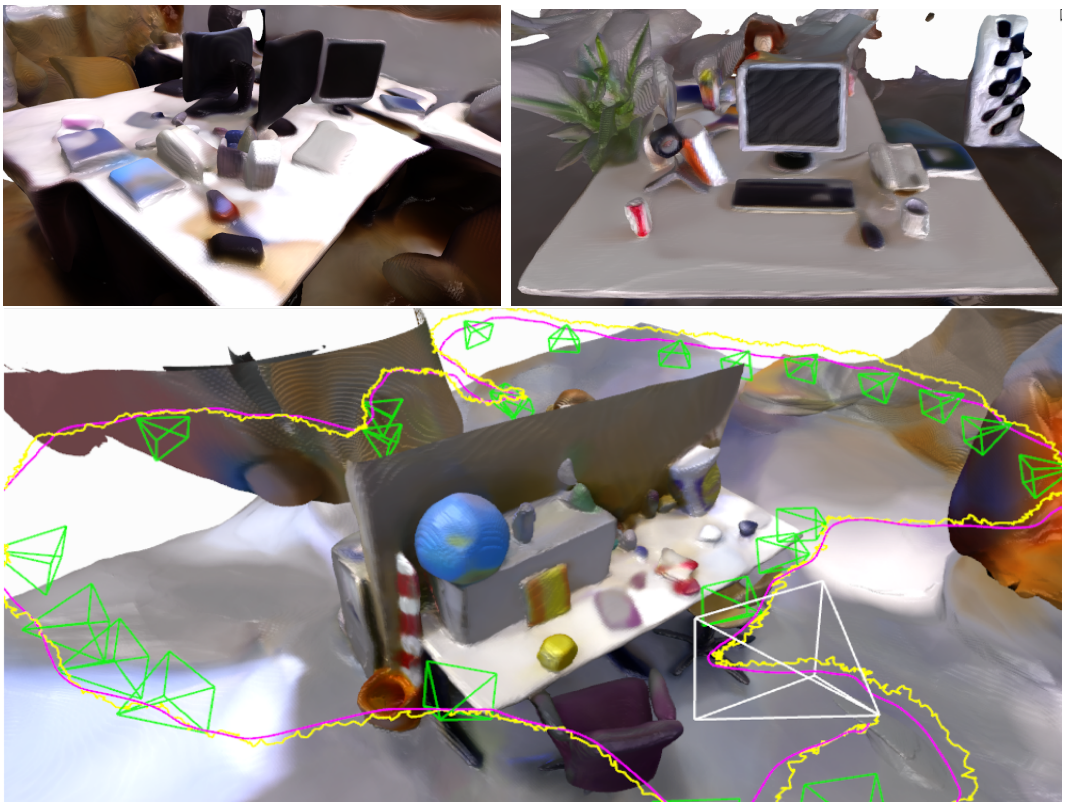


Figure 5.13: iMAP reconstruction results for TUM dataset.

Figure 5.14: Hole filling capacity of iMAP (top) against BAD-SLAM (bottom).

| | Default | Width | | Window | | Pixels | |
|---|---|---|---|---|---|---|---|
| | | 128 | 512 | 3 | 10 | 100 | 400 |
| **Tracking Time** [ms] | 101 | 80 | 173 | 84 | 144 | 74 | 160 |
| **Joint Optim. Time** [ms] | 448 | 357 | 777 | 373 | 647 | 340 | 716 |
| **Comp. Ratio** [<5cm %] | 77.22 | 75.79 | 76.91 | 75.82 | 77.35 | 77.33 | 77.49 |

Table 5.4: Timing results for tracking (6 iterations) and mapping (10 iterations), running concurrently on the same GPU. Default configuration: network width 256, window size 5, and 200 samples per keyframe. Last row: completion ratio for Replica `office-2`.

### 5.4.4 Ablative Analysis

We analyse the design choices that affect our system using the largest Replica scene: `office-2` with three different random seeds. Completion ratio results and timings are shown in Table 5.4. We found that network *width* = 256, keyframe *window* size limit of $W = 5$, and 200 *pixels* samples per frame offered the best trade-off of convergence speed and accuracy. We further show in Figure 5.15 that active sampling enables faster accuracy convergence and higher scene completion than random sampling.

These design choices enable our online implicit SLAM system to run at 10 Hz for tracking and 2 Hz for mapping. Our experiments demonstrate the power of randomised sampling in optimisation, and highlight the key finding that it is better to iterate fast with randomly changing information than to use dense and slow iterations.

Combining geometric and photometric losses enables our system to obtain full room scale reconstructions from few keyframes; 13 on average for the 8 Replica scenes in Table 5.1. Using more keyframes does little to further improve scene completion as shown in Table 5.5.

Implicit scene networks have the property of converging fast to low frequency

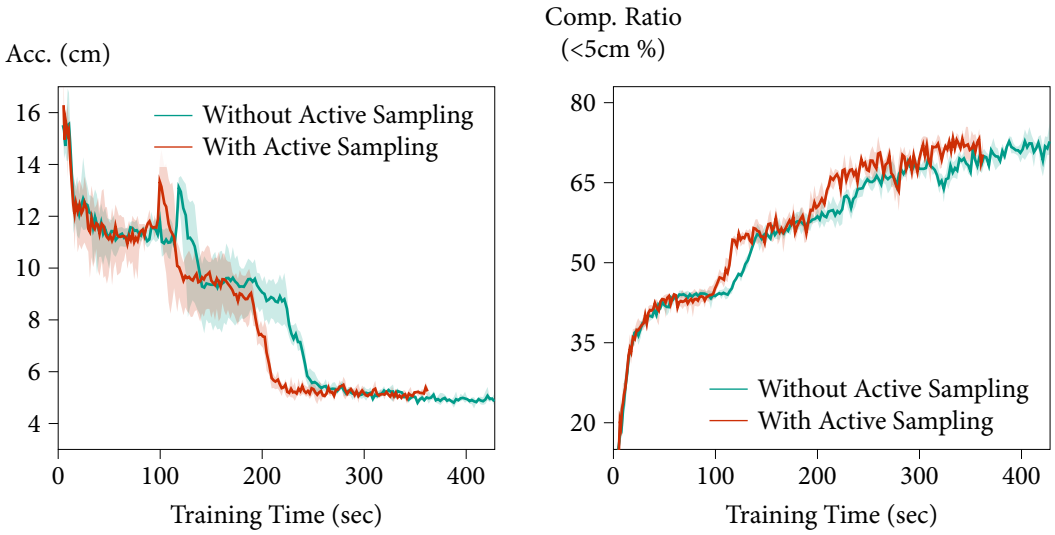



Figure 5.15: Active sampling obtains better completion with faster accuracy convergence than pure random sampling.

| | $t_P = 0.55$ | $t_P = 0.65$ | $t_P = 0.75$ | $t_P = 0.85$ |
|---|---|---|---|---|
| **# Keyframes** | 8 | 10 | 14 | 24 |
| **Comp. Ratio** [<5cm %] | 74.11 | 77.22 | 76.84 | 78.03 |

Table 5.5: Number of keyframe and completion ratio results for different selection thresholds in Replica `office-2`.

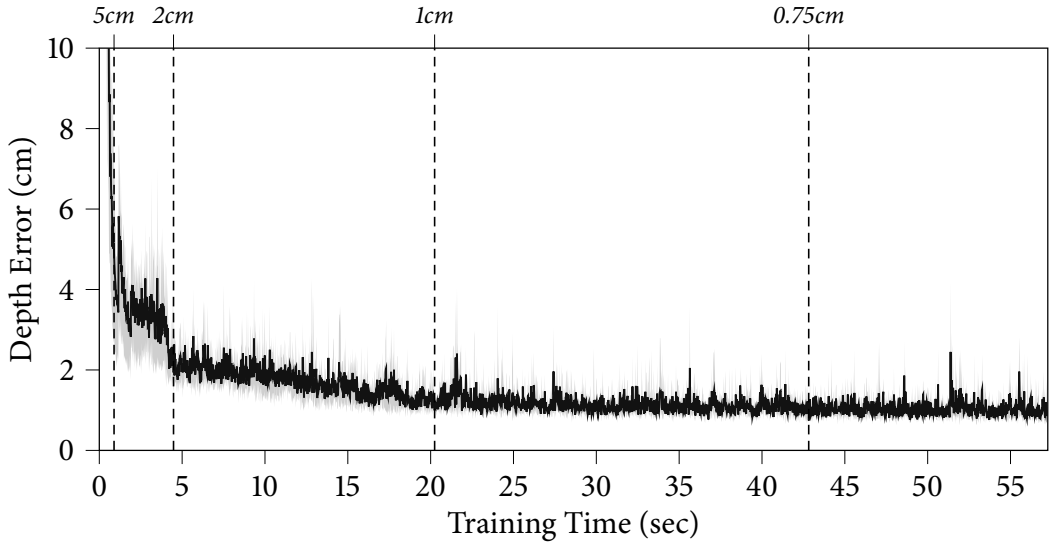



Figure 5.16: Reaching 5cm, 2cm, 1cm and 0.75cm depth error requires around 1, 4, 20, 43 seconds respectively.

Figure 5.17: Evolution of reconstruction detail.

shapes before adding higher frequency scene details. Figure 5.16 shows network training from a static camera averaged over 5 different real scenes. The depth loss falls below 5cm in under a second; under 2cm in 4 seconds; then continues to decrease slowly. When mapping a new scene our system takes seconds to get a coarse reconstruction and minutes to add in fine details. In Figure 5.17 we show how the system starts with a rough reconstruction and adds detail as the network trains and the camera moves closer to objects. This is a useful property in SLAM as it enables live tracking to work even when moving to unexplored regions.

## 5.5 Conclusions

We pose dense SLAM as real-time continual learning and show that an MLP can be trained from scratch as the only scene representation in a live system, thus enabling an RGB-D camera to construct and track against a complete and accurate volumetric model of room-scale scenes. The keys to the real-time but long-term SLAM performance of our method are: parallel tracking and mapping, loss-guided pixel sampling for rapid optimisation, and intelligent keyframe selection as replay to avoid network forgetting. We demonstrate that an MLP representation enables automatic scene compression and hole-filling, in Chapter 6 we will investigate how

the properties of efficient scene representation exhibit an inherent decomposition of elements into *objects*.

Chapter 6

# iLabel: Interactive Neural Scene Segmentation

## Contents

Work within this chapter was conducted under close collaboration with Shuaifeng Zhi, leading to the paper: Zhi, S.*, Sucar, E.*, Mouton, A., Haughton, I., Laidlow, T., Davison, A. (2022). **iLabel: Revelaing Objects in Neural Fields**. In *Proceedings of the IEEE Robotics and Automation Letters (RA-L))*. [Zhi et al., 2022]. (* denotes joint first author.)

## 6.1 Introduction

In Chapter 5 we showed that an MLP network can be trained from scratch in a single scene via automatic self-supervision to accurately and flexibly represent geometry and appearance. In this chapter we demonstrate that such a network naturally tends to discover an object-level decomposition of the scene, and that this can be revealed and aligned with user-defined semantic segmentation categories via extremely lightweight real-time annotation.

iLabel is the first interactive 3D semantic scene capture system with a unified neural field representation. It allows a user to achieve high-quality, dense scene reconstruction and multi-class semantic segmentation from scratch with only minutes of scanning and a few tens of semantic click annotations. The basis of iLabel is a real-time neural field SLAM system, augmented with a number of extra heads to serve as semantic outputs. These outputs inherit the coherence of the neural scene representation, and therefore also its decomposition properties. As the user scans a scene in real-time with a hand-held RGB-D camera, and provides very sparse semantic annotations by clicking, the network is able to generate a dense semantic segmentation of the whole scene.

Our approach requires no prior training on semantic datasets, and can therefore be applied in novel contexts, with categories defined on-the-fly by the user in an open-set manner. Standard methods for semantic scene segmentation use deep networks trained on datasets of thousands of images with dense, high-quality human

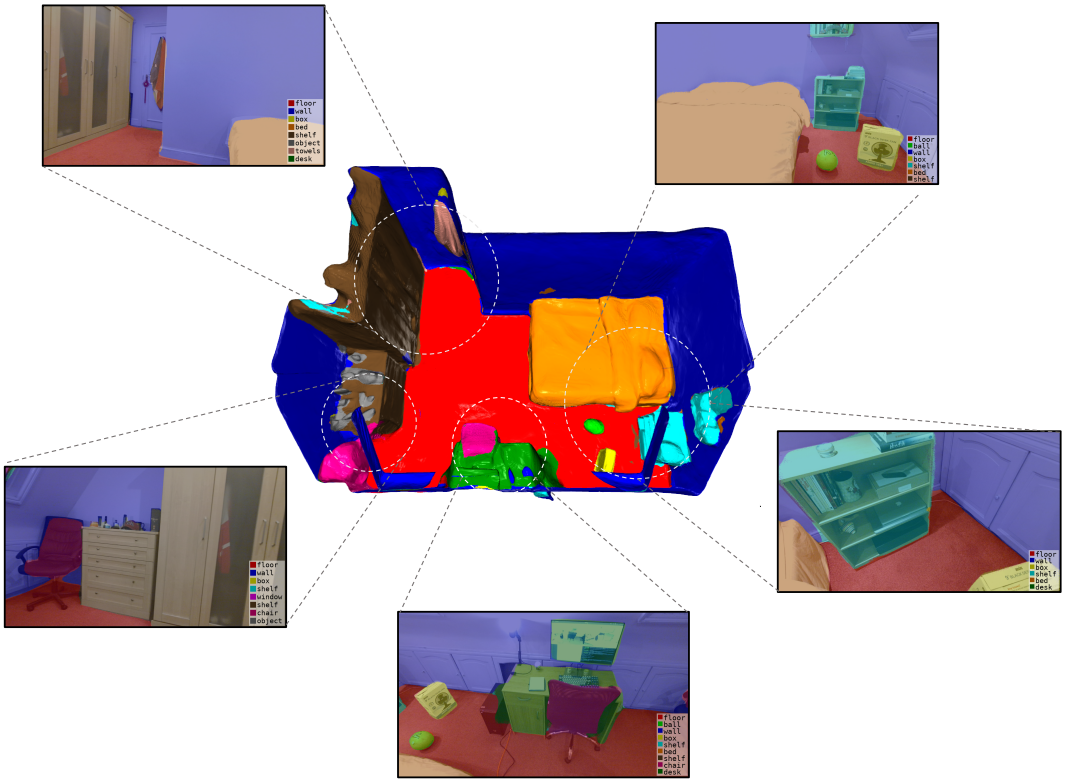


Figure 6.1: Whole-room semantic mesh labelled in real-time from only 140 interactive clicks and no prior training data.

annotations; even then they often have poor performance when the test scene is not a good match for the training set.

Because we render full predictions in real-time, the user can place annotations highly efficiently, to fix parts of the segmentation that are currently incorrect or to add new classes. This means that the quantitative labelling accuracy of iLabel scales powerfully with the number of clicks, and rapidly surpasses the accuracy of standard pre-trained semantic segmentation methods.

Alongside our core iLabel system for multi-class interactive scene segmentation, we introduce two promising variations. First, we show that hierarchical semantic labelling can be achieved by interpreting outputs as branches in a binary tree. Second, we demonstrate a 'hands free' labelling mode where an automatic uncertainty-guided framework selects a sequence of pixels for which to ask the user for label names

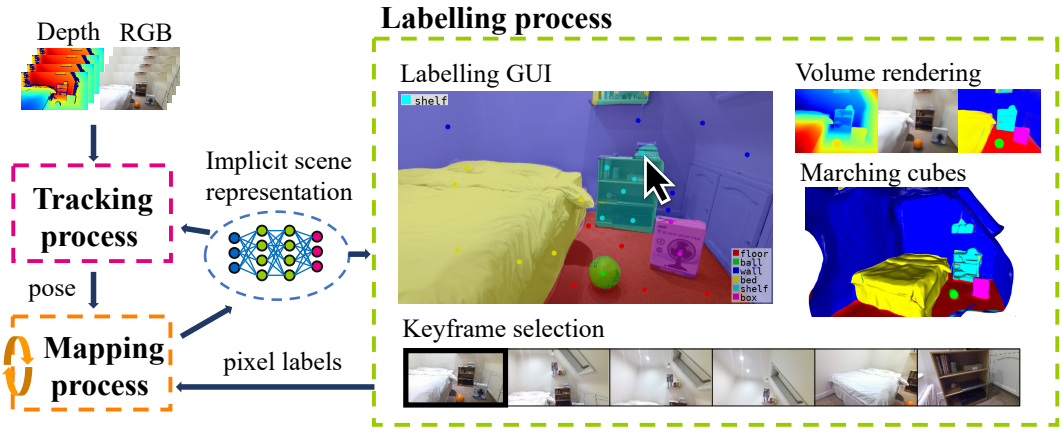



Figure 6.2: Three processes run in parallel: *(i) camera tracking*, *(ii) mapping* which optimises geometry, colour, and semantics, and *(iii) labelling* where the user provides labels through pixel selection.

without the need for clicks. We demonstrate iLabel in a wide variety of environments, from tabletop scenes to entire rooms and even outdoors. We believe iLabel to be a powerful and user-friendly tool, with much potential for interactive scene understanding with applications in augmented reality and robotics, as well as providing intuitive insights into the ability of neural fields to jointly represent correlated quantities.

We demonstrate iLabel in a wide variety of environments, from tabletop scenes to entire rooms and even outdoors. We believe iLabel to be a powerful and user-friendly tool, with much potential for interactive scene understanding with applications in augmented reality and robotics, as well as providing intuitive insights into the ability of neural fields to jointly represent correlated quantities. In Section 6.5 we demonstrate an application of iLabel for autonomous labelling of physical scene properties through robotic experimentation.

## 6.2 Related Work

Existing real-time, dense semantic mapping systems typically contain two parallel modules: 1) an RGB-D based geometric SLAM system, maintaining a dense 3D map of the scene, and 2) a semantic segmentation module that predicts dense

semantic labels of the scene [Hermans et al., 2014, Nakajima et al., 2019]. Multi-view semantic predictions are incrementally fused into the geometric model, yielding densely-labelled, coherent 3D scenes. While semantic segmentation has been performed using a variety of techniques [Nguyen et al., 2017, Krähenbühl and Koltun, 2011, Long et al., 2015, Chen et al., 2018, Xia et al., 2022], it is an inherently user-dependent and subjective problem [Martin et al., 2001]. User-in-the-loop systems are therefore crucial in enabling full flexibility when defining semantic relations between entities in a scene. In this context, the works most closely related to ours are SemanticPaint [Valentin et al., 2015] and Semantic Paintbrush [Miksik et al., 2015].

SemanticPaint [Valentin et al., 2015] is an online, user-in-the-loop system that allows the user to label a scene during capture. To this end, the user interacts with a 3D volumetric map, built from an RGB-D SLAM system, via voice and hand gestures [Nießner et al., 2013b]. A streaming random forest classifier, using hand-crafted features, learns continuously from the user gestures in 3D space. The forest predictions are used as unary terms in a conditional random field (CRF) to propagate the user annotations to unseen regions. As the CRFs are built upon the reconstructed data, there is an underlying assumption that these data are good enough to support label propagation. SemanticPaint is therefore restricted to comparably simple scenes and its efficacy in complex real-word scenarios is limited. A significant distinguishing factor between iLabel and SemanticPaint is ease-of-use. SemanticPaint has several distinct modes, requiring the user to switch between modes repeatedly and at well-timed intervals to obtain optimal results. In contrast, iLabel offers a much simpler and intuitive user experience, such that high-quality segmentations are obtained with far fewer interactions and no expert knowledge/intuition. Semantic Paintbrush [Miksik et al., 2015] extends SemanticPaint to outdoor scenes. Using a purely passive stereo setup for extended range and outdoor depth estimation, users visualise the reconstruction through a pair of optical see-through glasses and can draw directly onto it using a laser pointer to annotate objects in the scene. The system learns in an online manner from the these annotations and is thus able to

segment other regions in the 3D map.

In contrast to [Valentin et al., 2015, Miksik et al., 2015], iLabel does not rely on hand-crafted features, benefiting instead from a powerful joint internal representation of shape and appearance.

**Hierarchical Semantic Segmentation** Finding the hierarchical structure of complex scenes is a long-standing problem. Early attempts [Arbeláez et al., 2014] used image statistics to extract an ultrametric contour map (UCM), leading to further work on using convolutional neural networks (CNNs) for hierarchical image segmentation in a supervised manner [Xie and Tu, 2015, Maninis et al., 2016]. We show that iLabel can build a user-defined hierarchical scene segmentation interactively and store it within the weights of an MLP.

## 6.3 Method

iLabel represents 3D scenes using a neural field MLP which maps a 3D coordinate to colour, volume density and semantic values. We use the neural SLAM system from iMAP (Chapter 5) for real-time optimisation of the neural field [Mildenhall et al., 2020b], such as a small MLP, depth supervision, keyframes, coarse bin rendering, and sparse active pixel sampling. The MLP and camera poses of keyframes are jointly optimised through differential volume rendering while also tracking the position of a moving RGB-D camera against the neural representation.

In parallel with SLAM, a user provides annotations via clicks in the keyframes. Scene semantics are then optimised through semantic rendering of these user-selected pixels. The smoothness and compactness priors present in the MLP mean that the user-supplied labels are automatically and densely propagated throughout the scene. Thus iLabel can produce accurate, dense predictions from very sparse annotations and often even auto-segment objects and regions not labelled by the user. The ability to simultaneously reconstruct and label a scene in real-time allows for efficient labelling of new regions and for easy correction of errors in the current semantic

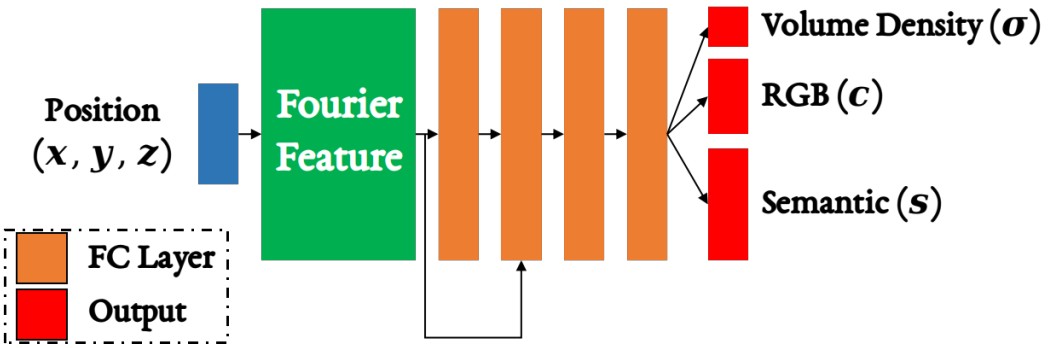



Figure 6.3: We employ a 4-layer MLP with feature size of 256.

predictions. See Figure 6.2 for an overview of iLabel.

### 6.3.1 Semantics Representation and Optimisation

At the heart of iLabel is continuous optimisation of the underlying neural field representation (Figure 6.3):

$$F_\theta(\mathbf{p}) = (\mathbf{c}, \mathbf{s}, \sigma), \tag{6.1}$$

where $F_\theta$ is a 4-layer MLP parameterised by $\theta$; $\mathbf{c}$, $\mathbf{s}$ and $\sigma$ are the radiance, semantic logits and volume density at the 3D position $\mathbf{p} = (x, y, z)$, respectively. The scene representation is optimised with respect to volumetric renderings of depth, colour and semantics, computed by compositing the queried network values along the back-projected ray of pixel $[u, v]$:

$$\hat{D}[u,v] = \sum_{i=1}^{N} w_i d_i, \quad \hat{I}[u,v] = \sum_{i=1}^{N} w_i \mathbf{c}_i, \quad \overline{S}[u,v] = \sum_{i=1}^{N} w_i \mathbf{s}_i, \tag{6.2}$$

where $N$ is the number of sampled quadrature points along the ray, $w_i = o_i \prod_{j=1}^{i-1}(1 - o_j)$ is the ray-termination probability of sample $i$ at depth $d_i$ along the ray; $o_i = 1 - \exp(-\sigma_i \delta_i)$ is the occupancy activation function; $\delta_i = d_{i+1} - d_i$ is the inter-sample distance.

As in iMAP (Chapter 5), geometry and keyframe camera poses are optimised by minimising the discrepancy between the captured and rendered RGB-D images from sparsely sampled pixels. Semantics are optimised with respect to the user-labelled pixels, with two different activations and losses, corresponding to the two semantic

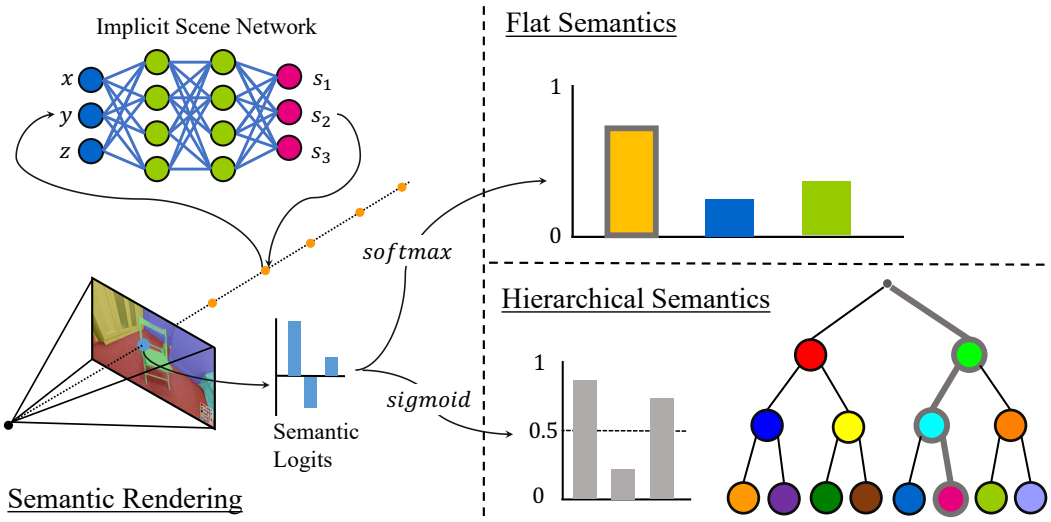



Figure 6.4: Semantic logits are rendered through ray sampling, and different activation functions are applied to the rendered logits for either flat or hierarchical semantic outputs.

modes described below. Figure 6.4 gives an overview of the semantic rendering process and the activation functions applied to the rendered logits.

**Flat Semantics** As in [Zhi et al., 2021], the network outputs $\mathbf{s}_i$ are multi-class semantic logits which are converted into image space by differential volume rendering (Eq. 6.2) followed by a *softmax* activation $\hat{S}[u,v] = softmax(\overline{S}[u,v])$. Semantics are then optimised using the image cross-entropy loss between the provided class ID and the rendered predictions.

**Hierarchical Semantics** We propose a novel hierarchical semantic representation through a binary tree, allowing for labelling and predicting semantics at different hierarchical levels. While the network output, $\mathbf{s}_i$, is still represented by an $n$-dimensional flat vector, $n$ now corresponds to the depth of the binary tree as opposed to the number of semantic classes. The semantic logits are rendered in the same manner, but the image activation and loss functions differ.

A *sigmoid* activation function is applied to the rendered logits, producing values in the range $[0,1]$. The $j^{\text{th}}$ rendered output value, $\hat{S}_j[u,v] = sigmoid(\overline{S}_j[u,v])$,

corresponds to the branching factor at tree level $j$. To obtain a hierarchical semantic prediction, each value $\hat{S}_j[u,v]$ is set to 0 or 1 by thresholding $\hat{S}_j[u,v]$ at 0.5, this means that the class output at level $j$ depends on the values of the previous levels. In the hierarchical setting, the user-supplied label corresponds to selecting a specific node in the binary tree. This label is transformed into a binary branching representation, and a binary cross-entropy loss is computed for each rendered value. A label selecting a tree node at level $L$ only conditions the loss on the output values up to and including level $L$: $\hat{S}_j[u,v], j \in \{1,...,L\}$.

With reference to the top half of Figure 6.9, the network outputs three values corresponding to the three levels in the tree. First, the user separates the scene into *foreground* and *background* classes. A background label corresponds to the vector $[0,*,*]$ where $*$ indicates that no loss is calculated for the second and third rendered values. The user then divides the background class further into *wall* and *floor*, where the *wall* label corresponds to vector $[0,1,*]$. The binary hierarchical representation allows the user to separate objects in stages. For example the user first separates a whole bookshelf from the rest of the scene, and later separates the books from the shelf without contradicting the initial labels, meaning that no labelling effort is wasted.

### 6.3.2 Semantic User Interaction Modes

Our system allows for two modes of interaction: 1) **manual interaction mode**, the usual interactive mode of iLabel, where users provide semantic labels in image space via clicks, and 2) **hands-free mode**, where the system generates automatic queries for the labels of informative pixels, driven by semantic prediction uncertainty (Figure 6.5). The latter mode eases the burden of manual annotation, and users could provide labels via text or voice.

In hands-free mode, uncertainty-based sampling actively proposes pixel positions where there there is least confidence in the semantic class for labelling, and the user only needs to supply a category name. This can be done with little computational

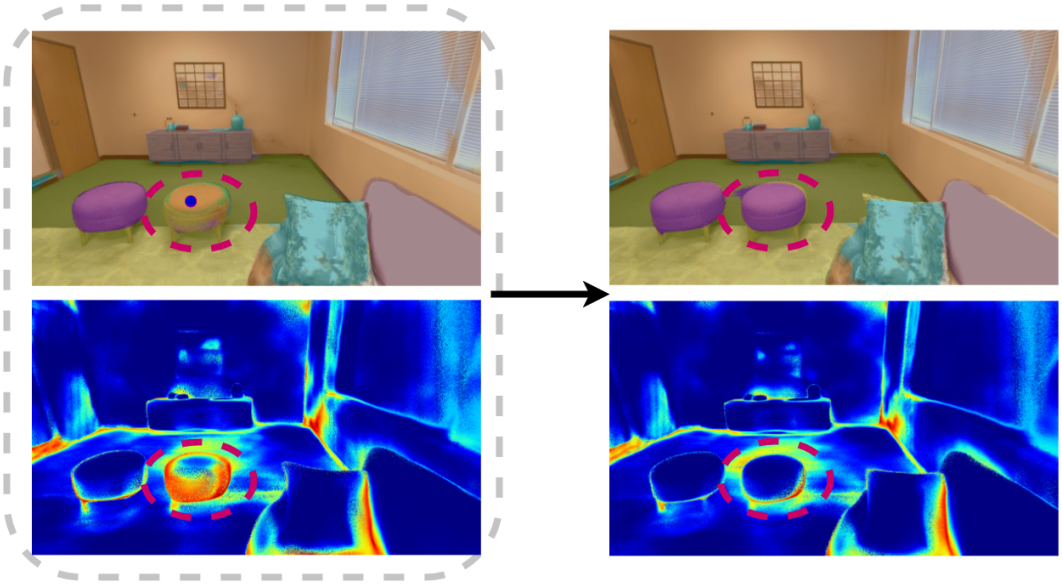


Figure 6.5: In hands-free mode with automatic query generation, semantic class uncertainty is used to actively select a pixel for which to request a label; in this case an unlabelled stool with ambiguous class prediction and high uncertainty is selected.

overhead [Settles, 2009, Ren et al., 2021]. We have explored several uncertainty measures: softmax entropy, least confidence and margin sampling [Settles, 2009]. For example, the softmax entropy is defined as $u_{entropy} = -\sum_{c=1}^{C} \hat{S}^c[u,v]\log(\hat{S}^c[u,v])$, where $C$ is the number of semantic categories.

At system run-time, semantic labels and corresponding uncertainty maps of all registered keyframes are rendered. To decide which keyframe to allocate queries to, we first compute frame-level entropy by accumulating pixel-wise entropy within frames and assign a higher probability to sampling the keyframe with higher frame-level entropy. Given a selected keyframe, we then randomly select the queried pixel coordinate from a pool of pixel positions with top-K highest entropy values. The frame-level and pixel-level uncertainty are updated every certainty mapping steps. K is set to 1% or 5% of pixel numbers to avoid repeated queries at nearby positions.

### 6.3.3 Implementation Details

iLabel operates on two GPUs (one for optimisation and the other only for rendering visualisation), running three concurrent processes: 1) tracking, 2) mapping, and 3)

labelling (see Figure 6.2).

The mapping process encompasses optimising the MLP parameters with respect to a growing set of $W$ keyframes and associated RGB-D observations: $\{(I_i, D_i, T_i)\}_{i=1}^{W}$, where $I_i$, $D_i$, $T_i$ are the colour image, depth image, and camera pose of the $i_{\text{th}}$ keyframe. As per iMAP (Chapter 5), the photometric loss $L_p$ and geometric loss $L_g$ are minimised on sparse, information-guided pixels. iLabel performs an additional optimisation on $K$ user-selected pixels ($\xi_i$) of keyframes and introduces a semantic loss $L_s$, minimising the following objective function:

$$\arg\min_{\theta} \frac{1}{K} \sum_{i=1}^{W} \sum_{(u,v)\in\xi_i} \underbrace{e_i^g[u,v]}_{L_g} + \alpha_p \underbrace{e_i^p[u,v]}_{L_p} + \alpha_s \underbrace{e_i^s[u,v]}_{L_s}, \tag{6.3}$$

where:

$$e_i^p[u,v] = \left|I_i[u,v] - \hat{I}_i[u,v]\right|, e_i^s[u,v] = -\sum_{c=1}^{C} S_i^c[u,v] \log(\hat{S}_i^c[u,v]), \tag{6.4}$$

$$e_i^g[u,v] = \frac{\left|D_i[u,v] - \hat{D}_i[u,v]\right|}{\sqrt{\hat{D}_{var}[u,v]}}, \hat{D}_{var}[u,v] = \sum_{i=1}^{N} w_i (\hat{D}[u,v] - d_i)^2, \tag{6.5}$$

and in the hierarchical setting:

$$e_i^s[u,v] = \sum_{l=1}^{L} -S_i^c[u,v] \log(\hat{S}_i^c[u,v]) - (1 - S_i^c[u,v]) \log(1 - \hat{S}_i^c[u,v]). \tag{6.6}$$

The labelling process coordinates user interactions and controls the rendering of semantic images and meshes (via marching cubes on a dense voxel grid queried from the MLP). The ADAM [Kingma and Ba, 2015b] optimiser is used with poses and map learning rates of 0.003 and 0.001. $\alpha_p$ and $\alpha_s$ are 5 and 8. Fourier features are used with sigma in the range $[25, 80]$, and input coordinated are scaled by $\frac{1}{10}$.
iLabel does not have an explicit/specific refinement process, and all user clicks are involved in the joint optimisation (Eq. 6.3). The optimisation keeps working and growing with changing sparse samples for colour and geometry reconstruction, and increasing annotated pixels for semantics, colour and depth as well.

**Timings results** Run on GeForce RTX 2080 GPU for a single iteration of tracking/10ms, mapping/45ms, and semantics/15ms, with pixel batch sizes of 200, 1000, and 100 respectively. Mapping and semantic optimisation run in parallel to tracking.

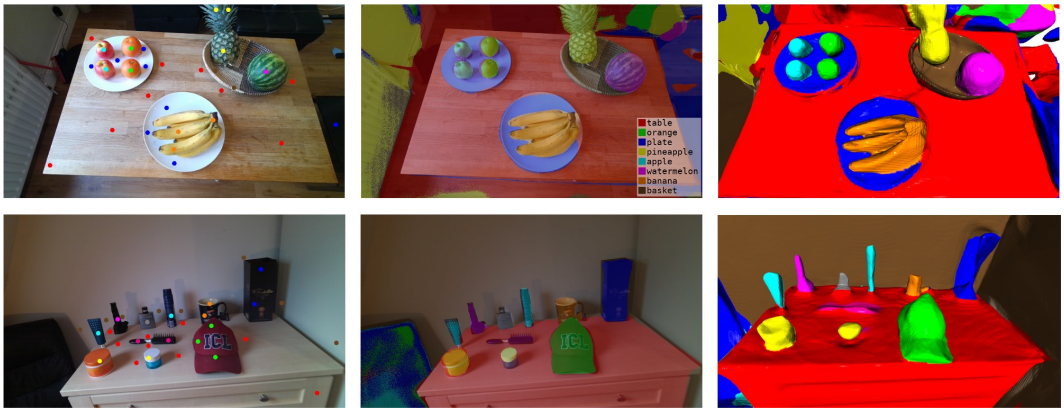



Figure 6.6: Precise segmentations can be obtained from just 1 or 2 interactive clicks per object. (Left: clicks; middle: dense labels rendered into a keyframe; right: full 3D mesh with labels.)


Figure 6.7: Segmentation results for challenging skeletal objects; left: pre-trained CNN on ScanNet (Section 6.4.2), right: iLabel.

On average tracking converges on 6 iterations and semantics propagate in 20 iterations. Rendering visualisation for labelling happens on a separate GeForce GTX 1080 GPU and takes 115ms for image size of 255x144.

## 6.4 Experiments

iLabel is an interactive tool intended for real-time use and we therefore emphasise that its strengths are best illustrated *qualitatively*. We provide extensive examples to demonstrate iLabel in a variety of interesting scenes. We show qualitative com-

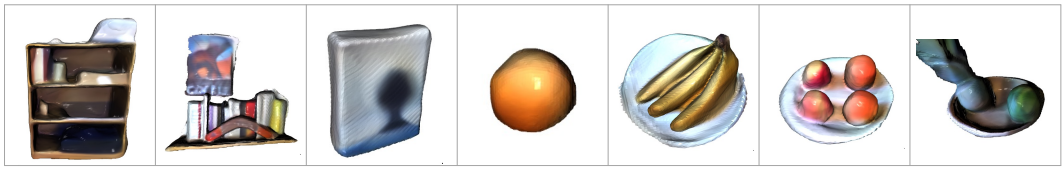


Figure 6.8: Catalog of object mesh assets separated with iLabel.

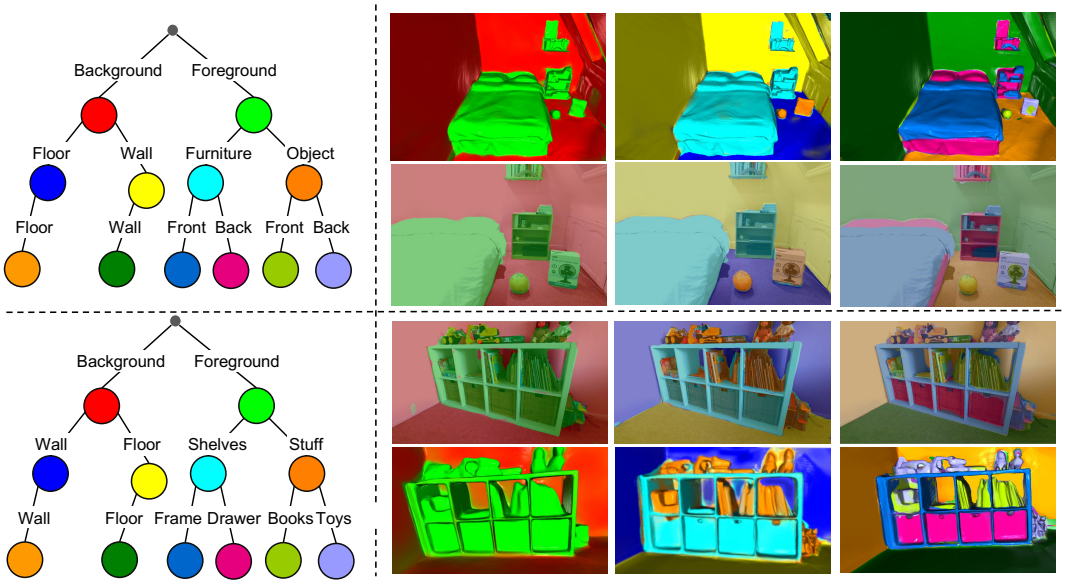



Figure 6.9: Binary tree as well as the segmentations at each level from the hierarchical mode of iLabel.

parisons with the only comparable system SemanticPaint and clearly demonstrate better segmentation quality. Additionally, we perform quantitative evaluations to show how semantic segmentation quality scales with additional user click labels, using a state-of-the-art, fully-supervised RGB-D segmentation baseline [Chen et al., 2020].

### 6.4.1 Qualitative Evaluation

As the geometry, colour and semantic heads share a single MLP backbone, user annotations are naturally propagated to untouched regions of the scene without specifying an explicit propagation mechanism (e.g. the pairwise terms of a CRF used in [Valentin et al., 2015]). This, together with a user-in-the-loop, enables efficient scene labelling with only a small number of well-placed clicks.

We have observed that the resulting embeddings are highly correlated for coherent 3D entities in the scene (e.g. objects, surfaces, etc.). Consequently, iLabel is able to segment these entities very efficiently, even with a single click. This is illustrated in Figure 6.6 and 6.10, where only a few clicks generate complete and precise segmentations for a wide range of objects and entities, ranging from small, coherent objects (e.g. fruit) to deformable and intricate entities (clothing and furniture).

The coordinate-based representation avoids quantisation and allows the network to be queried at arbitrary resolutions. This property allows reconstruction of detailed geometry and skeletal shapes that, when semantically labelled, render very precise segmentations. Figure 6.7 illustrates high-fidelity object segmentations which are challenging for a standard CNN.

iLabel can be used as an efficient tool for generating labelled scene datasets. For example, a scene of a complete room with 13 classes, can be fully segmented with high precision with only 140 user clicks (Figure 6.1). Alternatively, iLabel can be used to tag individual objects for generating object-asset catalogues (Figure 6.8) to aid robotic manipulation tasks, for example.

While iLabel is particularly powerful at segmenting coherent entities, Figure 6.11 also demonstrates its ability to propagate user-supplied labels to disjoint objects exhibiting similar properties. Each example shows label transfer between similar objects where only one has been labelled (e.g. (a) boxes on the bed, (b) food boxes and plastic cups and (c) toy dinosaurs). The table and chairs scene in Figure 6.11 (d) is especially interesting. Only four clicks are supplied: the label for the chair leg (blue) propagates to the leg of the table and the legs of the other chairs, while the table-top label (yellow) propagates to the seats of the chairs.

**Hierarchical scene segmentation** Figure 6.9 demonstrates iLabel's hierarchical mode. The colour-coded hierarchy (defined on-the-fly) is shown together with segmentations and scene reconstructions from each level. The results show the capacity of this representation to group objects at different levels, which has potential in applications where different tasks demand different groupings.

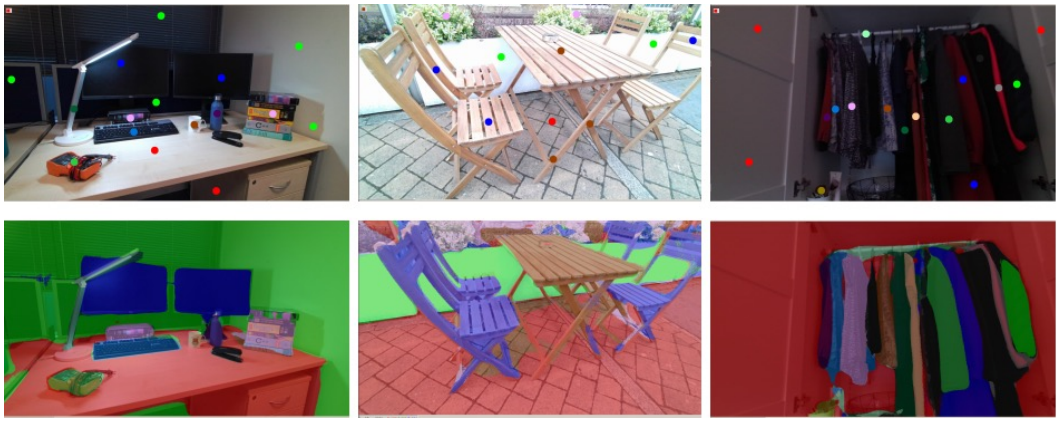


Figure 6.10: Efficient label propagation: iLabel produces high-quality segmentations of coherent 3D entities with very few user clicks, approximately 20-30 per scene.

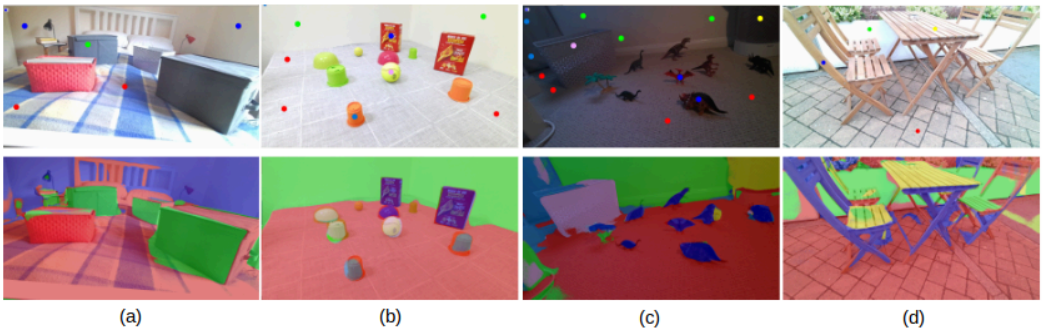
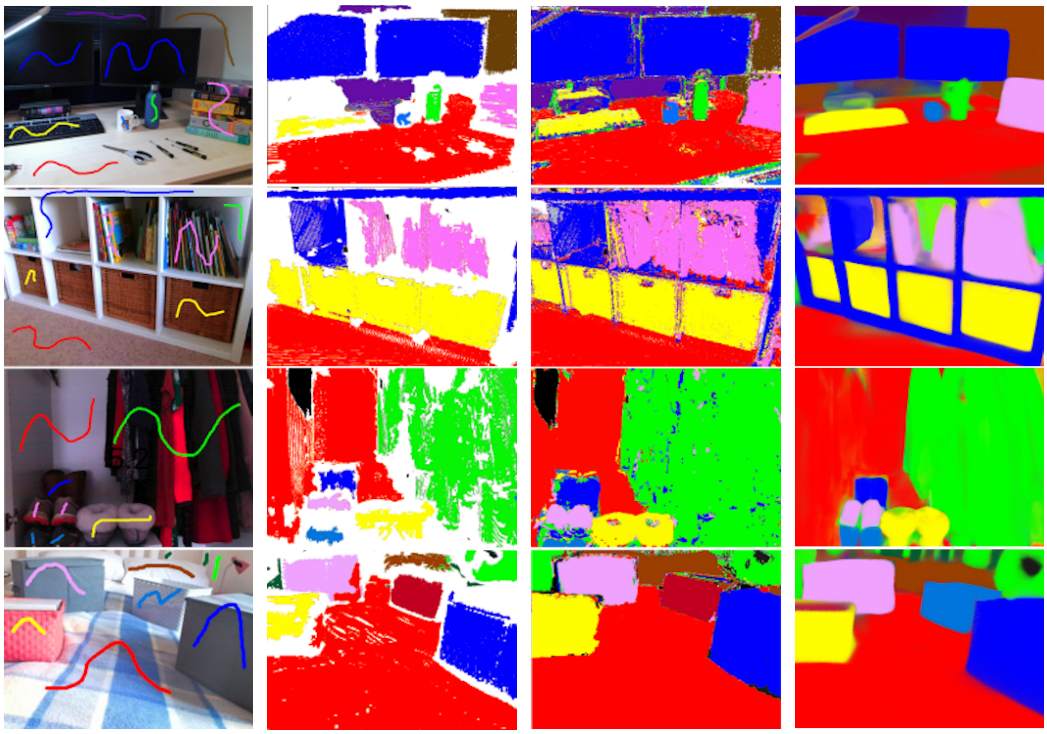



Figure 6.11: Generalisation: iLabel is able to transfer user labels to objects exhibiting similar properties. It is worth highlighting that the segmentation in (d) was achieved with only 4 clicks.

**Comparison to SemanticPaint** SemanticPaint (SPaint) [Valentin et al., 2015] is currently the only comparable online interactive scene understanding system. With several distinct modes (labelling, propagation, training, predicting, correcting, smoothing), which do not operate simultaneously, users have to switch between modes repeatedly (with careful consideration given to the duration spent in each mode) to obtain optimal results. In contrast, iLabel presents a unified interface for scene reconstruction, whereby user interaction, label propagation, learning and prediction occur simultaneously. The more intuitive and simpler interface presented by iLabel means that high-quality segmentations are obtained with far fewer interactions and no expert knowledge/intuition.



(a) Input annotations | (b) **SPaint:** Initial strokes | (c) **SPaint:** Extra strokes | (d) **iLabel:** Initial strokes

Figure 6.12: Comparison results between iLabel and SemanticPaint for user annotations in (a). (b) SPaint results for initial strokes; (c) SPaint results after corrections; (d) iLabel segmentations obtained using only the input strokes in (a).

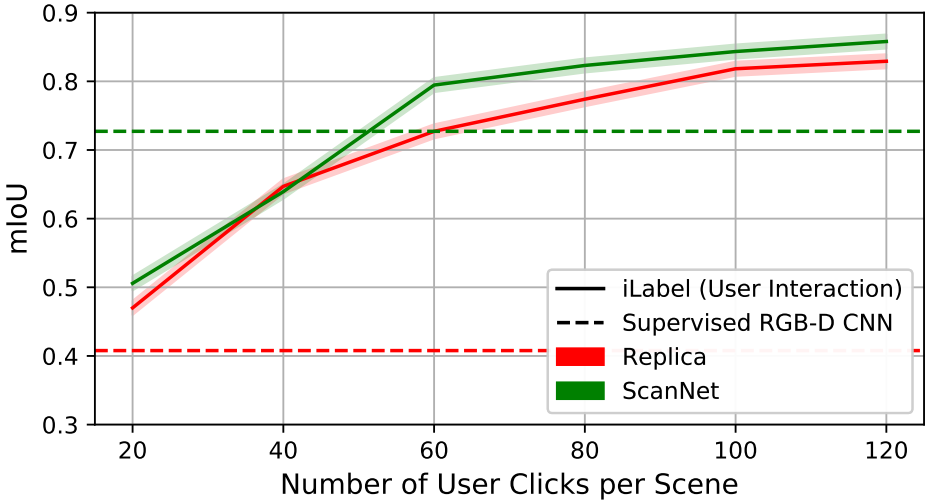
Qualitative comparisons between iLabel and SPaint is given in Figure 6.12. Scenes with varying degrees of complexity were chosen to demonstrate the superiority of iLabel even in scenes well-suited to SPaint (e.g. bottom row in Figure 6.12). For each scene in Figure 6.12, users annotated objects/regions with the strokes shown in (a). From these initial annotations only, iLabel was able to generate high-quality segmentations (Figure 6.12 (d)). In contrast, SPaint produced comparatively noisy and incomplete initial segmentations (Figure 6.12 (b)). Multiple mode switches and additional corrective strokes were required to generate the final SPaint results (Figure 6.12 (c)). We argue that the results produced by iLabel with only the initial user inputs ($< 10$ strokes), surpass those of SPaint after the additional user interactions.

### 6.4.2 Quantitative evaluation

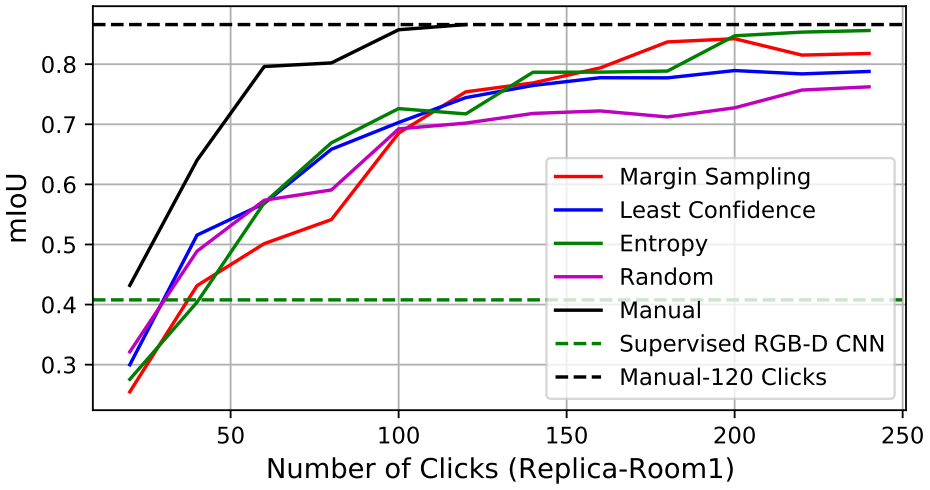
We evaluate iLabel's 2D semantic segmentation performance in both manual interaction and hands-free modes, with varying numbers of clicks per scene, on the public datasets Replica [Straub et al., 2019] and ScanNet [Dai et al., 2017a]. Both datasets are publicly available for research purposes under their licence. We report the mean Intersection Over Union (mIOU), averaged over ground truth labels remapped to NYU-13 class definitions.

**Baseline** While pre-trained segmentation models serve a different purpose than an interactive scene-specific system (to generalise to unseen scenes) we use them as a baseline to demonstrate the labelling efficiency of our system. iLabel scales rapidly with the number of clicks and rapidly surpasses the pretrained model, even when this has been trained on very similar scenes.
Performance is evaluated against SA-Gate [Chen et al., 2020] with a ResNet-101 DeepLabV3+ backbone [Chen et al., 2018], which is the current state-of-the-art in RGB-D segmentation. For Replica, we pre-train SA-Gate using the SUN-RGBD dataset [Song et al., 2015] and fine-tune on our generated Replica sequences to avoid over-fitting. We adopt a leave-one-out strategy, whereby fine-tuning is performed independently for each test scene using the remaining Replica scenes. For ScanNet, we train SA-Gate directly on the official training sets, achieving 63.98% mIOU on the validation sets of 13 classes. Approximately 11k (9860 and 475 images for our SUN-RGBD training and validation splits, 900 images for Replica fine-tuning) and 25k training images were used for baseline CNN training on each Replica and ScanNet experiment, respectively. The ResNet-101 backbone is initialised with ImageNet pre-trained weights through all the experiments. As per [Chen et al., 2020], depth maps use HHA encoding [Gupta et al., 2014], before which fast depth completion [Ku et al., 2018] is used for hole-filling in ScanNet.

**Results** Figure 6.13a shows the performance of iLabel compared against the supervised RGB-D CNN baseline (dashed horizontal line) on 5 Replica scenes and 6 ScanNet scenes from the validation set. The Replica dataset is a low data regime




(a) Manual Interaction Mode




(b) Hands-free Mode

Figure 6.13: Quantitative evaluation of 2D semantic segmentation on the Replica and ScanNet datasets. Both interaction modes are evaluated and outperform supervised baselines with a small annotation budget.

with only 7 scenes used for fine tuning, which makes generalisation specially hard. iLabel is specially suited for this settings, and surpasses the baseline with only 20 clicks per scene. In the ScanNet dataset where much more data is available, iLabel reaches similar accuracy to the baseline with around 50 clicks, and continues to improve surpassing the baseline by 20% at 120 clicks.

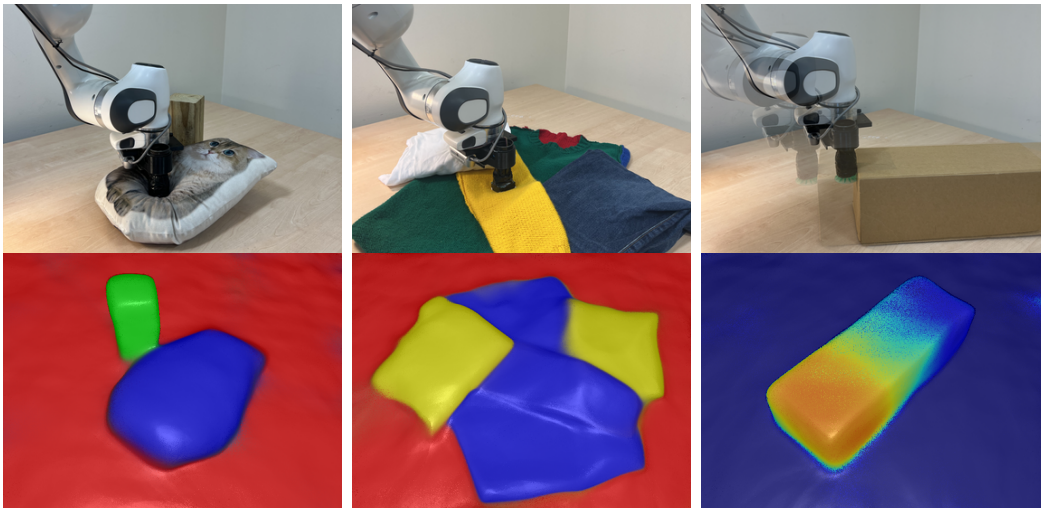
Figure 6.13b shows the effectiveness of automatic query generation guided by various uncertainty measurements, which opens the possibility for hands-free scene labelling, e.g., by voice command. As expected, this mode is less labelling efficient than manual clicks and takes around 240 clicks to reach similar performance but involves much less manual intervention. We show how random uniform pixel sampling achieves a lower performance, specially when more labels have been added, highlighting the importance of uncertainty guided pixel selection.

## 6.5 Robot Mapping of Physical Scene Properties

Work within this section describes was led by Iain Haughton builduing on top of the iLabel system, leading to the paper: Haughton, I., Sucar, E., Mouton, A., Johns, E., Davison, A. (2022). **Real-time Mapping of Physical Scene Properties with an Autonomous Robot Experimenter**. In *Proceedings of the Conference on Robot Learning (CoRL))*. [Haughton et al., 2022]

In this section we show an application of interactive labeling to autonomous robot mapping of physical scene properties. We build upon the iLabel system by exploiting an active, autonomous agent to remove the human from the loop entirely. We extend the predictive capabilities of the underlying MLP to include physical scene properties, which the robot autonomously queries from the scene. Here, we describe the components of our system that enable it to operate in a fully-autonomous manner on a physical agent to obtain rich, task-driven scene representations.

The robot builds an internal representation of its environment via a series of autonomous experiments. First, it actively selects interaction locations that are both feasible and information-rich (based on semantic entropy). Second, the selected 2D image



(a) Rigidity classification (b) Material classification (c) Force distribution

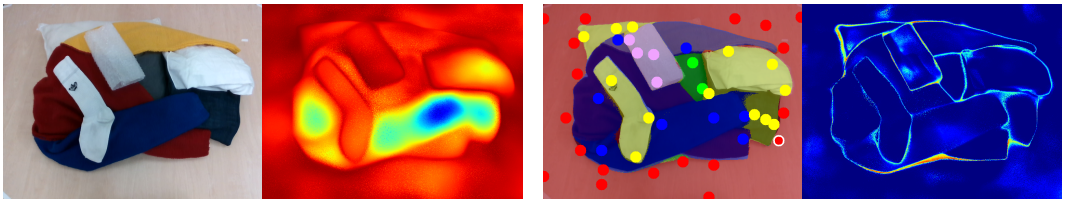
Figure 6.14: Over a few minutes our robot makes sparse, automatic physical scene interactions, such as touching to test rigidity, sampling local material type with spectroscopy or pushing to determine frictional force distribution. The interaction results are used as sparse labels to the output channels of a joint neural-field model of 3D shape and appearance, trained in real-time. Model coherence allows the measured physical properties to be efficiently and densely propagated to the whole scene, without the need for prior training data.

locations are mapped to the real-world coordinate system of the robot, and a physical interaction with the scene is planned and executed. Third, the resulting measurement is processed and/or classified (see specifics in Section 6.5.3) to obtain the ground-truth semantic label. Finally, using the labels obtained in this manner, scene semantics are optimised through semantic rendering of the robot-selected keyframe pixels.

### 6.5.1 Modes of interaction

Our framework facilitates the autonomous discovery and mapping of any measurable characteristic of a scene, provided that a suitable measurement sensor and interaction protocol can be defined. We demonstrate three particular interaction types: 1) predicting rigidity by top-down poking; 2) predicting material type using a single-pixel multiband spectrometer[1]; and 3) predicting frictional force distributions by

[1]SparkFun Triad Spectroscopy Sensor - AS7265x (Qwiic)



(a) Initial keyframe and uncertainty (b) Final keyframe and uncertainty

Figure 6.15: Example material type segmentations using a spectrometer. 46 interactions were required to separate the pile of laundry into wool (blue), cotton (yellow) and synthetic (green/pink) materials. Red/blue signifies high/low uncertainty in the uncertainty map.

lateral pushing. These modes constitute the basis of our fully automatic system described in the following sections.

### 6.5.2 Entropy-guided interactions

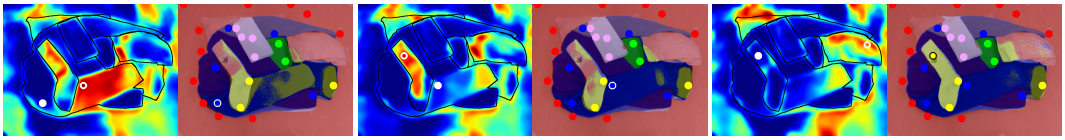
Well-placed user clicks, especially in regions where the model is performing poorly, are the most beneficial in terms of improving segmentation quality. This observation was exploited in an automatic query generation framework, whereby an uncertainty-based sampling was used to actively propose pixel positions for the user to label. Similarly, we utilise softmax entropy to guide the physical interactions that the robot makes with the scene, encouraging interactions that are optimal in terms of information gain and thereby minimising the number of interactions required to produce optimal segmentations. Softmax entropy, $u_S$, is defined as [Ila et al., 2010]:

$$u_S = -\sum_{c=1}^{C} \hat{\mathbf{S}}_c\left[u, v\right] \log\left(\hat{\mathbf{S}}_c\left[u, v\right]\right), \tag{6.7}$$

with $\hat{\mathbf{S}}_c[u, v]$ the rendered semantic distribution and $C$ the number of categories.

### 6.5.3 Semantic representation

The inputs to the semantic head of our neural-field MLP can be one of several physical properties, measured via apposite affordances and modes of interaction. Raw sensor measurements acquired by the robot need to be post-processed or converted into the target variable being predicted by the semantic head of the MLP. For binary



(a) Interaction 25 (b) Interaction 26 (c) Interaction 27

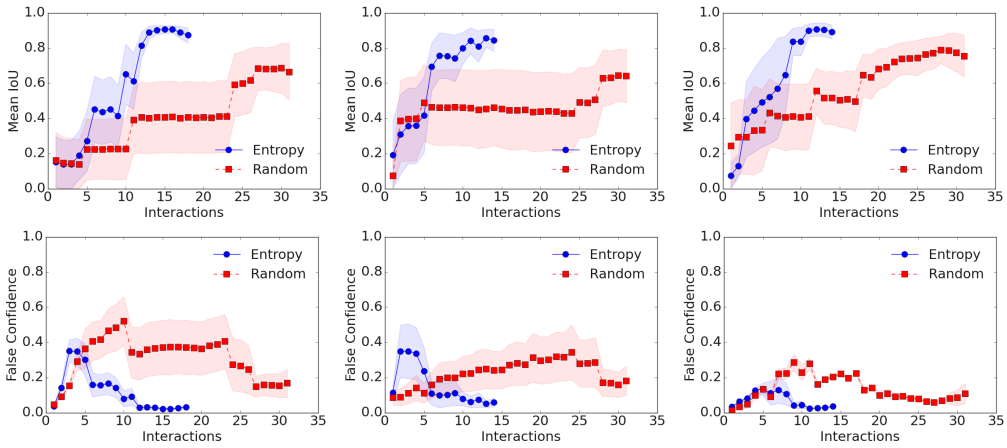
Figure 6.16: Demonstration of autonomous guidance over 3 consecutive interactions in Figure 6.15. From left to right, the interactions (unfilled markers) follow the highest uncertainty pixel. After interaction (filled marker), there is a localised reduction in uncertainty.

(a) Scene 1: material (b) Scene 2: material (c) Scene 3: rigidity

Figure 6.17: Our system can interact with and segment a variety of scenes.

prediction tasks (e.g. rigidity), this may be as simple as applying a threshold to the raw measurement. Multi-class target variables may require additional processing. For example, when predicting material type from a multidimensional spectrometer reading, we use a pretrained multiclass SVM classifier which outputs predefined material classes, which are then fed to the semantic head. In both scenarios the semantic head of the MLP predicts a categorical value and can be optimised using cross-entropy loss.

We additionally demonstrate for the first time the prediction of continuous-valued target variables in the *semantic head* of the MLP, where the ground-truths are sparse, in contrast to the dense ground-truths used in the optimisation of the colour and density heads. For example, when predicting frictional force distributions,




(a) Scene 1 (b) Scene 2 (c) Scene 3

Figure 6.18: Comparisons of mean IoU (top) and false-confidence (bottom) vs. number of interactions for entropy-based and random exploration approaches.

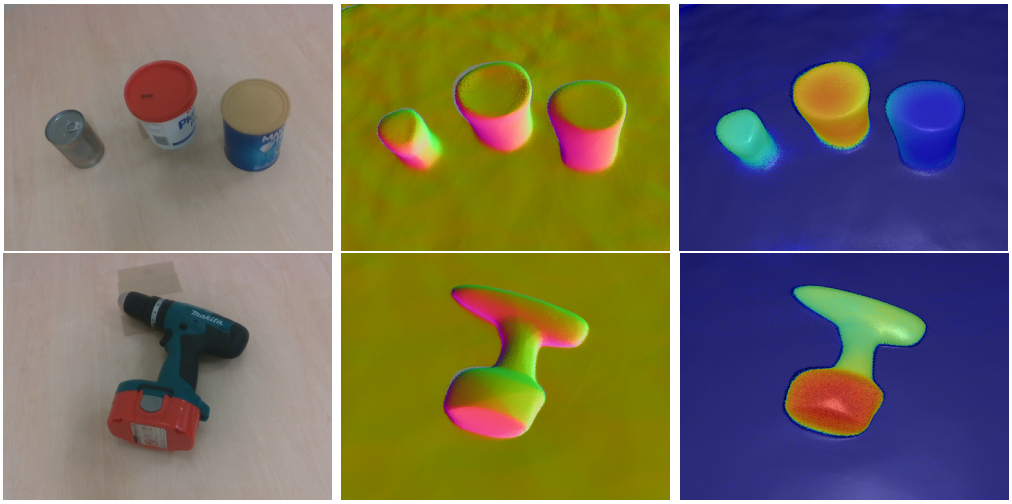
we feed the minimum (stiction) force required to move an object, directly to the semantic head and optimise using an $L1$ loss.

### 6.5.4 Single frame optimisation

During lateral pushing, interactions between the robot and the scene may introduce object displacements which violates the static-scene assumption. While this assumption allows to optimise over an expanding set of keyframes, a dynamic scene potentially invalidates historic keyframes, ultimately leading to errors in the reconstruction. We therefore clear the keyframe history and corresponding labels after each interaction in this mode. Our experimentation has suggested that neural radiance representations possess some form of temporal memory characteristic over the labelled properties, whereby network weights adapt over time and maintain consistency with the dynamic scene, provided scene changes are comparatively small.

### 6.5.5 Robotic Experiments

We demonstrate the ability of our system to perform a series of autonomous experiments, using the aforementioned interactive modes, to discover and predict a variety of physical scene properties. We demonstrate the quantitative benefits of entropy-



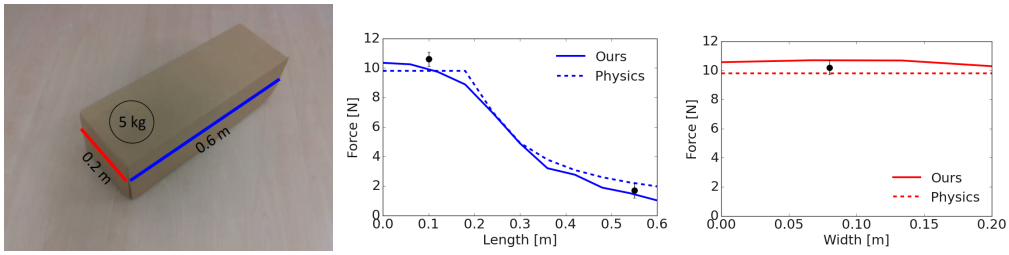
(a) Scene (b) Normals rendering (c) Force rendering

Figure 6.19: Stiction force mapping. Top row: three cylindrical objects with uniform mass of (from left to right) 0.5 kg, 1.5 kg and 0.1 kg. Guided by entropy, the robot applies a single push to each object measuring, stiction forces of 1.0 N, 3.0 N and 0.2 N. Bottom row: power drill with non-uniform mass distribution. The final rendering was produced after a sequence of 3 pushes.

guided experimentation and, in the case of rigidity and material-type classification, compare segmentation performance against two state-of-the-art, class-agnostic segmentation techniques (see Sec. 6.5.5 for details). Finally, we refer the reader to our supplementary video for additional results.

We use a Franka Emika Panda robot, anchored to a table on which a variety of objects are arranged (Figure 6.14). The robot is equipped with a Realsense D435 RGB-D sensor [Keselman et al., 2017], tracked using the forward kinematics of the arm, which is controlled using ROS [System, ]. Prior to physical experiments, the robot builds a geometric reconstruction of the scene, to allow for collision-free motion planning. For this purpose, a set of RGB-D keyframes is captured over a series of random motions in order to optimise the 3D neural field, and subsequent querying of the network produces a collision mesh and normal map. All objects of interest are placed within reach of the robot arm and any points located beyond this range, or on the plane of the table, are automatically labelled 'table'.

Rigidity and material-type prediction may be viewed as segmentation problems. While training any popular instance segmentation technique (e.g. Mask R-CNN [He




(a) Crossection (b) Length distribution (c) Width distribution

Figure 6.20: Rendered stiction force distribution, compared to an analytical approach with privileged information, along the length (blue) and width (red) of a box with non-uniform density.

et al., 2017]) on the object classes present in our scenes (e.g. material types), is likely to produce high-quality segmentations, one would need to repeat this training for each scenario. Therefore, instead of comparing against closed-set segmentation techniques, we consider two state-of-the-art class-agnostic instance segmentation approaches: 1) Mask R-CNN trained to perform class-agnostic segmentation [Gouda et al., 2022] and 2) Unseen Clustering Network (UCN) [Xie et al., 2021] with RICE refinement [Xie et al., 2022]. For each method, we perform instance segmentation on the keyframe and use the resulting instance mask to guide the robot-scene interaction. In particular, the robot takes a single sensor reading as near to the centre of each instance in the mask as is feasible and propagates the measurement to the rest of the region. The measurements are converted to categorical labels (binary rigidity or material-type) in the same manner as described in Sec. 6.5.3. We report the mean Intersection over Union (mIoU) averaged over the ground-truth labels.

Table 6.1 shows the quantitative performance comparison against the Mask R-CNN and UCN baselines for each scene. As expected, the baselines perform well for scenes 1 and 3, which contain geometrically-coherent objects and strong colour and depth cues. Scene 2, however, is considerably more challenging for the colour and/or depth-based baselines, characterised by a significant drop in performance. In contrast, our autonomous approach performs well for all three scenes, with comparable results to the baselines in Scenes 1 and 3 and significantly superior results in Scene 2.

Table 6.1: Classification performance for different types of scene, (examples in Figure 6.17).

| Segmentation | Example | Ours | Mask R-CNN | UCN + RICE |
|---|---|---|---|---|
| Material | Scene 1 | 0.91 ± 0.02 | 0.92 ± 0.02 | 0.90 ± 0.02 |
| Material | Scene 2 | 0.89 ± 0.03 | 0.56 ± 0.11 | 0.56 ± 0.10 |
| Rigidity | Scene 3 | 0.91 ± 0.04 | 0.92 ± 0.02 | 0.91 ± 0.02 |

### 6.5.6 Entropy exploration ablation study

Figure 6.15 illustrates material discovery in a complex scene containing wool (blue), cotton (yellow) and synthetic (green/pink) materials. Prior to physical measurements, there is high uncertainty (red) across the entire scene, while the final uncertainty map has high confidence (blue) throughout. We show the evolution of the uncertainty map through three consecutive interactions in Figure 6.16. The robot is guided to a high-entropy pixel (unfilled circle). On completion of the experiment, there is a clear, localised uncertainty reduction surrounding the target region (filled circle).

We observe that while the uncertainty in the localised region of measurement decreases, it often increases in more distant regions. As the model accumulates information, it continuously adapts its predictions and corresponding confidence, ultimately converging on an accurate representation. This observation motivates the use of uncertainty as an exploration metric in neural implicit representations. To substantiate the benefits of entropy-driven exploration quantitatively, we conducted an ablation study comparing performance to random exploration. We compare the evolution of mIoU and false-confidence (where the model produces high-confidence but incorrect predictions) with an increasing number of interactions for each technique. Figure 6.18 demonstrates superior convergence rates for entropy-guided interaction in all three benchmark scenes in Figure 6.17 across both metrics.

### 6.5.7 Force measurement analysis

Figure 6.19 illustrates the stiction force maps produced by our framework for objects with uniform (top row) and non-uniform (bottom row) mass and friction distributions. Note that in each scene the objects are displaced following the pushes per-

formed by the robot. The scene in the top row contains three cylindrical containers of varying mass and material. As desired, the resulting force renderings match the varying masses. This is potentially valuable information when planning for downstream manipulation tasks (e.g. distinguishing between full and empty containers). A key observation in Figure 6.19 is that the renderings for objects remain consistent despite displacement, demonstrating for the first time a memory quality in neural field representations.

In the bottom row of Figure 6.19, we demonstrate the ability of the robot to predict stiction force values reliably, even for complex geometries, with non-uniform mass and friction distributions. We substantiate this quantitatively in Figure 6.20, where the robot interacts with a non-uniform rectangular box containing a 5 kg weight at one end. We show that the output of our model, after three pushes, is comparable to that of a simple analytical physics model [Mason, 1986] with access to privileged information, including the contact surface area, mass distribution and friction coefficient.

## 6.6 Conclusions

We have shown that online, scene-specific training of a compact MLP model which encodes scene geometry, appearance and semantics allows sparse interactive labelling to produce accurate dense semantic segmentation. Despite promising results, our system's label propagation mechanism works well mainly for proximal regions and/or those sharing similar geometry or texture. A deeper understanding of this mechanism is necessary to enable better control of this process and to improve generalisation performance. In addition, how to improve its ability to hierarchically and uniquely represent rich semantics within the network is worthy of further exploration. As architectures and methods for neural field representation of scenes continue to improve, we expect these gains to be passed on to our labelling approach, and for tools like iLabel to become highly practical for applications where users are able to teach AI systems efficiently about useful scene properties.

# Conclusions and Future Work

In this thesis, we addressed the problem of jointly modeling geometric reconstruction and semantic abstraction using compressive representations. We presented different contributions encompassed by the use of neural scene representations within incremental real-time SLAM, tackling open challenges in designing scene representations for both object-level and dense SLAM systems. Specifically, we presented Neural Object Descriptors for SLAM at the level of objects, with flexibility for intra-class shape variations, as well as an MLP Neural Scene representation for compressive incremental dense semantic mapping of room-scale scenes. The key to generative inference with both of these scene representations was the development of a differentiable volumetric rendering function used for joint optimisation of the 3D representation and camera trajectory. We demonstrated our developments in practical real-time systems and robotic applications.

In Chapter 4, we addressed the problem of multi-view shape reconstruction using class-level priors. The objective of this project was to extend the benefits of template-based object mapping such as [Salas-Moreno et al., 2013] of having a compact scene representation and capacity to obtain a complete reconstruction with partial observations, but allowing for shape variation within a semantic category of objects. To achieve this, we combined 3D generative models and rendering-based shape optimisation. We learned a compact and continuous class-level latent space from aligned 3D shape models using a 3D Variational Auto-Encoder (VAE) CNN. This latent space is used for the reconstruction of new shapes from the trained classes by optimizing the compact code with respect to depth images. For image-based optimization, we presented a differentiable volumetric rendering function, and

demonstrated its increased robustness to current alternative formulations both qualitatively and quantitatively. We showed that within the scope of the trained classes (mug, bottle, can, and bowl), our method achieves complete and accurate shape reconstructions with partial and noisy depth measurements. The generative formulation of our representation allowed us to build a full self-contained SLAM system with objects as landmarks. The practical nature of our representation was highlighted in a robotic system capable of completing the challenging task of tight object packing.

We believe that NodeSLAM represents an important advancement in object-level mapping, going beyond fixed shapes by incorporating shape priors based on semantic classes. Since the paper was published, it has inspired relevant extensions, such as those that handle dynamic scenes [Xu et al., 2022] or address large-scale object mapping [Wang et al., 2021a]. However, important limitations of this work include the need for a 3D dataset to learn the latent space and the use of a single uni-modal code to represent shapes, which limits expressiveness. Ongoing research is being conducted to tackle these challenges; for example, by learning shape models with image observations through rendering supervision as in [Tulsiani et al., 2018, Henzler et al., 2021], or by modelling more complex shapes through part-based models [Mo et al., 2019], or combinations of simple geometric primitives [Landgraf et al., 2021, Genova et al., 2019].

In Chapter 5, we proposed incremental compressive representations for fully generative dense SLAM of room-scale scenes. In this project, we addressed open problems in dense SLAM, specifically the ability to perform full joint optimization of a dense 3D map and camera trajectory, as well as the ability to model spatial relationships between points at different levels of detail. To address these challenges, we proposed the use of a global and continuous 3D representation modelled with a scene-specific randomly initialized MLP Neural Field. We designed a real-time SLAM system, iMAP, for efficiently training the MLP representation from scratch. Our SLAM system runs from a stream of depth images, utilizing a keyframe-based parallel tracking and mapping design, and is optimized with differentiable volume rendering. We demonstrated the ability for efficient optimisation through random-

ised sparse rendering with active pixel selection. We showcased our system in a varied range of settings, and highlighted advantages over fusion-based SLAM, such as reduced memory requirements for representation, robustness to sparse views, and joint optimization of maps and camera poses, allowing for small drift corrections. Furthermore, we showed that the compressive and continuous MLP-based representation allows for plausible filling in of holes where depth information is missing.

iMAP presented an important advancement in demonstrating the ability of online training of neural fields, and providing insights into geometric reconstruction with an MLP. The main limitations of iMAP come at the cost of computation for rendering and optimisation by the use a global map representation, where computation increases proportional to representation capacity. Further developments on 3D representations have explored hybrid representations, a combination of local spatial structures followed by a global MLP [Müller et al., 2022, Clark, 2022], which offer a middle point in the trade-off between compute and memory storage. iMAP has inspired dense SLAM systems such [Zhu et al., 2022] that scale to bigger spaces based on a hybrid representation or [Kong et al., 2023], where a scene is decomposed into a per-object MLP. We believe there is a lot of open research in combining local and global structures with questions on flexible decomposition and the type of representation function.

In Chapter 6, we investigated the automatic abstraction properties of the scene-specific compressive representation used in iMAP. We accomplished this by constructing an interactive segmentation system called iLabel, where a user provides sparse semantic labels of the scene. These labels are then automatically propagated by the global MLP map. We demonstrated the hypothesis that an efficient representation automatically decomposes a scene into coherent regions or objects, which can be revealed with minimal user interaction. We showed that label propagation correlates with both the local geometric boundaries of objects and global appearance properties. iLabel's real-time interactive and open-set features enable efficient label placement and error correction, allowing for the acquisition of full and precise segmentations of complex scenes where pre-trained neural networks struggle. We believe that iLabel is a powerful and user-friendly tool for scene labelling that

provides insights into the decomposition properties of Neural Fields. iLabel has been extended to encode additional properties such as pre-trained 2D CNN features in [Mazur et al., 2023], which allows to go beyond local segmentation to open-set interactive semantic grouping.

## 6.7 Future Work

There remain many open challenges for achieving generally useful representations and predictive models for Spatial AI and robotic systems. There are various exciting research directions we are interested in taking this work, which we outline next.

### 6.7.1 Continual Learning

One big challenge for global representations such as the MLP used in iMAP is the need of a replay buffer to avoid catastrophic forgetting. This incurs in big computational costs for incremental training, and limits scaling to larger spaces as the replay buffer grows. We believe there is potential in exploring representations which model uncertainty in order to replace replay with proper probabilistic marginalisation. Examples of possible representation are Bayesian neural networks [Ebrahimi et al., 2020] or Gaussian processes [Williams and Fitzgibbon, 2006]. This representations could bring additional benefits such as modeling the uncertainty of the 3D reconstruction, as well as the probabilistic integration of 2D depth priors such as those predicted by [Dexheimer and Davison, 2023, Laidlow et al., 2020]. It would also be interesting to apply methods for inspecting neural networks such as [Fong and Vedaldi, 2017] to analyse the relation between network activations and scene structures such as the ones obtained from ilabel, which could provide insight into how to lock network weights or prune the network to be more efficient.

### 6.7.2 Hybrid representations

As we mentioned before hybrid representations with a combination of local and global functions are promising for improving computational efficiency. We believe exploring flexible local decomposition beyond uniform grids is an interesting direc-

tion. This could include localised basis functions such as wavelets [Rho et al., 2023] or kernel methods with inducing points as described in [Ramos and Ott, 2016]. The use of flexible local decompositions could be useful for modeling scene change in dynamic environments.

### 6.7.3 Semantic mapping

In our previous work NodeSLAM we demonstrated how semantics can provide strong priors for reconstruction in limited categories, and we would want to be more general in using semantics for aiding reconstruction. In iLabel for example structural priors could be inserted such as in regions annotated to be walls, floors or other surfaces which tend to be planar. However, we believe the key building compact and semantically meaningful representation is by exploiting repetition and self-similarity in a scene, which could be achieved by introducing convolutional structures in the representation which bring inductive biases for translation equivariance, ideally at different hierarchical levels.

### 6.7.4 General Priors

Finally, recent progress has shown the ability to learn general 2D priors for natural images through massive scale training [Ramesh et al., 2022]. We believe this will play an important role on improving 3D reconstruction specially in the setting with limited observations. We would like to explore how to incorporate this priors into an incremental SLAM setting, or how to fine tune them to directly obtain geometrical predictions such as depth images.

# Bibliography

[Alcantarilla et al., 2012] Alcantarilla, P. F., Bartoli, A., and Davison, A. J. (2012). KAZE features. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 26

[Anandan, 1989] Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision (IJCV)*, 2:283–310. 31

[Arbeláez et al., 2014] Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 138

[Armstrong et al., 1996] Armstrong, M., Zisserman, A., and Hartley, R. (1996). Self-Calibration from Image Triplets. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 27

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359. 26

[Bloesch et al., 2018] Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., and Davison, A. J. (2018). CodeSLAM – Learning a Compact, Optimisable Representation for Dense Visual SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 113

[Bolitho et al., 2009] Bolitho, M., Kazhdan, M., Burns, R., and Hoppe, H. (2009). Parallel Poisson surface reconstruction. In *Proceedings of the International Symposium on Visual Computing*. 35

[Bosch et al., 2007] Bosch, A., Zisserman, A., and Munoz, X. (2007). Image classification using random forests and ferns. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 37

[Broida et al., 1990] Broida, T. J., Chandrashekhar, S., and Chellappa, R. (1990). Recursive 3-D Motion Estimation from a Monocular Image Sequence. *IEEE Transactions on Aerospace and Electronic Systems*, 26:639–656. 28

[Brostow et al., 2008] Brostow, G., Shotton, J., Fauqueur, J., and Cipolla., R. (2008). Segmentation and Recognition using Structure from Motion Point Clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 39

[Buhmann et al., 1995] Buhmann, J., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F. E., Strikos, J., and Thrun, S. (1995). The mobile robot rhino. *Ai Magazine*, 16(2):31–31. 32

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 26

[Castellanos, 1998] Castellanos, J. A. (1998). *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. PhD thesis, Universidad de Zaragoza, Spain. 28

[Castle et al., 2007] Castle, R. O., Gawley, D. J., Klein, G., and Murray, D. W. (2007). Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 40

[Cavallari and Di Stefano, 2016] Cavallari, T. and Di Stefano, L. (2016). Volume-Based Semantic Labeling with Signed Distance Functions. *Image and Video Technology*, 1:544–556. 39

[Chabra et al., 2020] Chabra, R., Lenssen, J. E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., and Newcombe, R. (2020). Deep Local Shapes: Learning local SDF priors for detailed 3d reconstruction. *Proceedings of the European Conference on Computer Vision (ECCV)*. 113

[Chambolle and Pock, 2011] Chambolle, A. and Pock, T. (2011). A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145. 32

[Chang et al., 2015] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). ShapeNet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*. 43, 88

[Cheeseman et al., 1996] Cheeseman, P., Kanefsky, B., Kraft, R., Stutz, J., and Hanson, R. (1996). Super-Resolved Surface Reconstruction from Multiple Images. In *Maximum Entropy and Bayesian Methods*, volume 62, pages 293–308. Springer Netherlands. 34

[Chen et al., 2018] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 137, 149

[Chen et al., 2020] Chen, X., Lin, K.-Y., Wang, J., Wu, W., Qian, C., Li, H., and Zeng, G. (2020). Bi-directional cross-modality feature propagation with separation-and-aggregation gate for rgb-d semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 145, 149

[Chen and Medioni, 1992] Chen, Y. and Medioni, G. (1992). Object modeling by registration of multiple range images. *Image and Vision Computing (IVC)*, 10(3):145–155. 33

[Chibane et al., 2020] Chibane, J., Pons-Moll, G., et al. (2020). Neural unsigned distance fields for implicit function learning. *Neural Information Processing Systems (NIPS)*. 73, 113

[Chiuso et al., 2002] Chiuso, A., Favaro, P., Jin, H., and Soatto, S. (2002). Structure from Motion Causally Integrated Over Time. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(4):523–535. 28

[Civera et al., 2011] Civera, J., Gàlvez-Lòpez, D., Riazuelo, L., Tard'øs, J. D., and Montiel, J. M. M. (2011). Towards semantic slam using a monocular camera. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 40

[Clark, 2022] Clark, R. (2022). Volumetric bundle adjustment for online photorealistic scene capture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 163

[Collet et al., 2011] Collet, A., Martinez, M., and Srinivasa, S. S. (2011). The moped framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research (IJRR)*, 30(10):1284–1306. 37

[Collins, 1996] Collins, R. T. (1996). A Space-Sweep Approach to True Multi-Image Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 31

[Cox et al., 1996] Cox, I. J., Hingorani, S. L., Rao, S. B., and Maggs, B. M. (1996). A maximum likelihood stereo algorithm. In *Computer Vision and Image Understanding (CVIU)*. 31

[Cremers et al., 2001] Cremers, D., Schnorr, C., and Weickert, J. (2001). Diffusion-snakes: combining statistical shape knowledge and image information in a variational framework. In *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 137–144. IEEE. 41

[Csurka et al., 2004] Csurka, G., Dance, C., Fan, L., Williamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *ECCV04 workshop on Statistical Learning in Computer Vision*, pages 59–74. 37

[Curless and Levoy, 1996] Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH*. 31, 32, 124

[Dai et al., 2017a] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017a). ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 149

[Dai et al., 2020] Dai, A., Diller, C., and Nießner, M. (2020). SG-NN: Sparse generative neural networks for self-supervised scene completion of RGB-D scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 113

[Dai et al., 2017b] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017b). BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics (TOG)*, 36(3):24:1–24:18. 112

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 37

[Dambreville et al., 2008a] Dambreville, S., Rathi, Y., and Tannenbaum, A. (2008a). A framework for image segmentation using shape models and kernel space shape priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(8):1385–1399. 42

[Dambreville et al., 2008b] Dambreville, S., Sandhu, R., Yezzi, A., and Tannenbaum, A. (2008b). Robust 3d pose estimation and efficient 2d region-based seg-

mentation from a 3d shape prior. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 42

[Dame et al., 2013] Dame, A., Prisacariu, V. A., Ren, C. Y., and Reid, I. (2013). Dense reconstruction using 3d object shape priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 43

[Davison, 1998] Davison, A. J. (1998). *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford. 28

[Davison, 2003] Davison, A. J. (2003). Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 29

[Dellaert et al., 2017] Dellaert, F., Kaess, M., et al. (2017). Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139. 54

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 96

[Dexheimer and Davison, 2023] Dexheimer, E. and Davison, A. J. (2023). Learning a depth covariance function. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 164

[Ebrahimi et al., 2020] Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. (2020). Uncertainty-guided continual learning with bayesian neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 164

[Engelmann et al., 2016] Engelmann, F., Stückler, J., and Leibe, B. (2016). Joint object pose estimation and shape reconstruction in urban street scenes using 3d shape priors. In *Pattern Recognition: 38th German Conference, GCPR 2016, Hannover, Germany, September 12-15, 2016, Proceedings 38*, pages 219–230. Springer. 43

[Engels et al., 2006] Engels, C., Stewénius, H., and Nistér, D. (2006). Bundle Adjustment Rules. In *Proceedings of Photogrammetric Computer Vision*. 29

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. 29

[Fischler and Elschlager, 1973] Fischler, M. A. and Elschlager, R. A. (1973). The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92. 37

[Fong and Vedaldi, 2017] Fong, R. C. and Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 164

[Genova et al., 2019] Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W., and Funkhouser, T. (2019). Learning shape templates with structured implicit functions. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 162

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 38

[Gkioxari et al., 2019] Gkioxari, G., Malik, J., and Johnson, J. (2019). Mesh r-cnn. 84

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org. 60

[Gouda et al., 2022] Gouda, A., Ghanem, A., and Reining, C. (2022). Category-agnostic segmentation for robotic grasping. *arXiv preprint arXiv:2204.13613*. 157

[Grossberg, 1982] Grossberg, S. (1982). How does a brain build a cognitive code? In *Studies of mind and brain*, pages 1–52. Springer. 113

[Gupta et al., 2014] Gupta, S., Girshick, R., Arbelaez, P., and Malik, J. (2014). Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 149

[Harris and Pike, 1987] Harris, C. G. and Pike, J. M. (1987). 3D Positional Integration from Image Sequences. In *Proceedings of the Alvey Vision Conference*, pages 233–236. 28

[Harris and Stephens, 1988] Harris, C. G. and Stephens, M. (1988). A Combined Corner and Edge Detector. In *Proceedings of the Alvey Vision Conference*, pages 147–151. 26

[Hartley, 1994] Hartley, R. (1994). Lines and points in three views—a unified approach. In *Proceedings of the Image Understanding Workshop*. 27

[Hartley, 1995] Hartley, R. (1995). In Defence of the 8-Point Algorithm. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 27

[Haughton et al., 2022] Haughton, I., Sucar, E., Mouton, A., Johns, E., and Davison, A. (2022). Real-time mapping of physical scene properties with an autonomous robot experimenter. In *Conference on Robot Learning (CoRL)*. 23, 151

[He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 38, 92, 97, 156

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 38

[Henzler et al., 2021] Henzler, P., Reizenstein, J., Labatut, P., Shapovalov, R., Ritschel, T., Vedaldi, A., and Novotny, D. (2021). Unsupervised learning of 3d

object categories from videos in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 162

[Hermans et al., 2014] Hermans, A., Floros, G., and Leibe, B. (2014). Dense 3d semantic mapping of indoor scenes from rgb-d images. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 39, 137

[Hinterstoisser et al., 2012] Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. (2012). Gradient Response Maps for Real-Time Detection of Texture-Less Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 37

[Hinterstoisser et al., 2013] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2013). Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 37

[Horn, 1986] Horn, B. (1986). *Robot Vision*. MIT Press, Cambridge MA. 27

[Hosni et al., 2013] Hosni, A., Rhemann, C., Bleyer, M., Rother, C., and Gelautz, M. (2013). Fast Cost-Volume Filtering for Visual Correspondence and Beyond. *PAMI*, 35(2):504–511. 32

[Huttenlocher et al., 1993] Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J. (1993). Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(9):850–863. 37

[Ila et al., 2010] Ila, V., Porta, J., and Andrade-Cetto, J. (2010). Information-Based Compact Pose SLAM. *IEEE Transactions on Robotics (T-RO)*, 26(1):78–93. 153

[Izadi et al., 2011] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R. A., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A. J., and Fitzgibbon, A. (2011). KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of ACM Symposium on User Interface Software and Technolog (UIST)*. 84

[Kahler et al., 2015] Kahler, O., Prisacariu, V. A., Ren, C. Y., Sun, X., Torr, P. H. S., and Murray, D. W. (2015). Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 34

[Kajiya, 1986] Kajiya, J. T. (1986). The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 35

[Kalman, 1960] Kalman, R. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45. 26

[Kang, 2006] Kang, H. R. (2006). *Computational color technology*. Spie Press Bellingham. 69, 70, 72

[Keller et al., 2013] Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., and Kolb, A. (2013). Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *Proc. of Joint 3DIM/3DPVT Conference (3DV)*. 112

[Keselman et al., 2017] Keselman, L., Iselin Woodfill, J., Grunnet-Jepsen, A., and Bhowmik, A. (2017). Intel Realsense stereoscopic depth cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–10. 156

[Kim et al., 2012] Kim, Y. M., Mitra, N. J., Yan, D.-M., and Guibas, L. (2012). Acquiring 3D Indoor Environments with Variability and Repetition. In *SIGGRAPH Asia*. 40

[Kingma and Ba, 2015a] Kingma, D. P. and Ba, J. (2015a). ADAM: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 117

[Kingma and Ba, 2015b] Kingma, D. P. and Ba, J. (2015b). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 143

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 64, 88

[Kirkpatrick et al., 2017] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526. 113

[Klein and Murray, 2007] Klein, G. and Murray, D. W. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 30, 98, 111, 114

[Kolmogorov and Zabih, 2001] Kolmogorov, V. and Zabih, R. (2001). Computing Visual Correspondence with Occlusions using Graph Cuts. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 31

[Kong et al., 2023] Kong, X., Liu, S., Taher, M., and Davison, A. J. (2023). vmap: Vectorised object mapping for neural field slam. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 163

[Koppula et al., 2011] Koppula, H. S., Anand, A., Joachims, T., and Saxena, A. (2011). Semantic Labeling of 3D Point Clouds for Indoor Scenes. In *Neural Information Processing Systems (NIPS)*. 39

[Krähenbühl and Koltun, 2011] Krähenbühl, P. and Koltun, V. (2011). Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *Neural Information Processing Systems (NIPS)*. 137

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*. 38

[Ku et al., 2018] Ku, J., Harakeh, A., and Waslander, S. L. (2018). In defense of classical image processing: Fast depth completion on the cpu. In *Proceedings of the Canadian Conference on Computer and Robot Vision (CRV)*. 149

[Kumar et al., 2005] Kumar, M. P., Ton, P., and Zisserman, A. (2005). Obj cut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 38

[Kumar et al., 1989] Kumar, R. R., Tirumalai, A., and Jain, R. C. (1989). A nonlinear optimization algorithm for the estimation of structure and motion parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 27

[Kundu et al., 2014] Kundu, A., Li, Y., Daellert, F., Li, F., and Rehg, J. M. (2014). Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 39

[Kundu et al., 2018] Kundu, A., Li, Y., and Rehg, J. M. (2018). 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 84

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*. 38

[Laidlow et al., 2020] Laidlow, T., Czarnowski, J., Nicastro, A., Clark, R., and Leutenegger, S. (2020). Towards the Probabilistic Fusion of Learned Priors into Standard Pipelines for 3D Reconstruction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 164

[Landgraf et al., 2021] Landgraf, Z., Scona, R., Laidlow, T., James, S., Leutenegger, S., and Davison, A. J. (2021). Simstack: A generative shape and instance model for unordered object stacks. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 162

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 38

[Leonard and Whyte, 1991] Leonard, J. J. and Whyte, D. H. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3). 28

[Lesort et al., 2019] Lesort, T., Caselles-Dupré, H., Garcia-Ortiz, M., Stoian, A., and Filliat, D. (2019). Generative models from the perspective of continual learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE. 114

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. (2011). BRISK: Binary Robust Invariance Scalable Keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 26

[Levenberg, 1944] Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168. 29

[Li et al., 2020] Li, K., Rünz, M., Tang, M., Ma, L., Kong, C., Schmidt, T., Reid, I., Agapito, L., Straub, J., Lovegrove, S., et al. (2020). Frodo: From detections to 3d objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 101

[Liu and Cooper, 2010] Liu, S. and Cooper, D. B. (2010). Ray Markov Random Fields for image-based 3D modeling: Model and efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 35

[Liu and Cooper, 2011] Liu, S. and Cooper, D. B. (2011). A complete statistical inverse ray tracing approach to multi-view stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 35

[Liu and Cooper, 2014] Liu, S. and Cooper, D. B. (2014). Statistical Inverse Ray Tracing for Image-Based 3D Modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(10). 35

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 137

[Loper and Black, 2014] Loper, M. and Black, M. J. (2014). OpenDR: An Approximate Differentiable Renderer. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 35

[Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching Cubes: A high resolution 3D surface construction algorithm. In *Proceedings of SIGGRAPH*. 31, 68

[Lowe, 1999] Lowe, D. G. (1999). Object Recognition from Local Scale-Invariant Features. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 26, 37

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 32

[Maltoni and Lomonaco, 2019] Maltoni, D. and Lomonaco, V. (2019). Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73. 114

[Maninis et al., 2016] Maninis, K.-K., Pont-Tuset, J., Arbeláez, P., and Van Gool, L. (2016). Convolutional oriented boundaries. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 138

[Martin and Moravec, 1996] Martin, C. and Moravec, H. (1996). Robot Evidence Grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Pittsburgh, PA. 32

[Martin et al., 2001] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 137

[Mason, 1986] Mason, M. T. (1986). Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71. 159

[Matsuki et al., 2023] Matsuki, H., Sucar, E., Laidlow, T., Wada, K., and Scona, Raluca Davison, A. J. (2023). imode: Real-time incremental monocular dense mapping using neural field. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 23

[Mazur et al., 2023] Mazur, K., Sucar, E., and Davison, A. J. (2023). Feature-realistic neural fusion for real-time, open set scene understanding. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 24, 164

[McCormac et al., 2018] McCormac, J., Clark, R., Bloesch, M., Davison, A. J., and Leutenegger, S. (2018). Fusion++:volumetric object-level slam. In *Proceedings of the International Conference on 3D Vision (3DV)*. 41, 84, 86, 103

[McCormac et al., 2017] McCormac, J., Handa, A., Davison, A. J., and Leutenegger, S. (2017). SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 39

[Mescheder et al., 2019a] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019a). Occupancy Networks: Learning 3D reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 73, 113

[Mescheder et al., 2019b] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019b). Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 101

[Microsoft Corp, 2010] Microsoft Corp (2010). Microsoft Kinect. `https://www.xbox.com/en-US/xbox-one/accessories/kinect`. 32

[Miksik et al., 2015] Miksik, O., Vineet, V., Lidegaard, M., Prasaath, R., Nießner, M., Golodetz, S., Hicks, S. L., Pérez, P., Izadi, S., and Torr, P. H. (2015). The Semantic Paintbrush: Interactive 3D Mapping and Recognition in Large Outdoor Spaces. 137, 138

[Mildenhall et al., 2020a] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020a). Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 36

[Mildenhall et al., 2020b] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020b). NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 73, 113, 115, 138

[Mo et al., 2019] Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N., and Guibas, L. (2019). Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics*, 38(6):Article 242. 162

[Moravec, 1977] Moravec, H. P. (1977). Towards Automatic Visual Obstacle Avoidance. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, page 584. 26

[Moravec and Elfes, 1985] Moravec, H. P. and Elfes, A. (1985). High resolution maps from angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 32

[Morris et al., 2001] Morris, R., Smelyanskiy, V., and Cheeseman, P. (2001). Matching Images to Models — Camera Calibration for 3-D Surface Reconstruction. In *Proceedings of the International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*. 34

[Mouragnon et al., 2006] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). Real-Time Localization and 3D Reconstruction. In *Pro-*

*ceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 29

[Mourikis and Roumeliotis, 2007] Mourikis, A. I. and Roumeliotis, S. I. (2007). A Multi-State Constraint Kalman Filter for Vision-Aided Inertial Navigation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3565–3572. IEEE. 30

[Mozos et al., 2007] Mozos, Ò., Triebel, R., Jensfelt, P., Rottmann, A., and Burgard, W. (2007). Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5). 39

[Müller et al., 2022] Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15. 163

[Mur-Artal and Tardós, 2014] Mur-Artal, R. and Tardós, J. D. (2014). ORB-SLAM: Tracking and Mapping Recognizable Features. In *Workshop on Multi View Geometry in Robotics (MVIGRO) - RSS 2014*. 79

[Mur-Artal and Tardós, 2017] Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics (T-RO)*, 33(5):1255–1262. 127

[Murphy, 2023] Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press. 65

[Nakajima et al., 2019] Nakajima, Y., Kang, B., Saito, H., and Kitani, K. (2019). Incremental class discovery for semantic segmentation with rgbd sensing. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 137

[Newcombe, 2012] Newcombe, R. A. (2012). *Dense Visual SLAM*. PhD thesis, Imperial College London. 34

[Newcombe et al., 2011a] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A.

(2011a). KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 33, 112, 124

[Newcombe et al., 2011b] Newcombe, R. A., Lovegrove, S., and Davison, A. J. (2011b). DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 31, 112

[Newman, 1999] Newman, P. (1999). *On the Structure and Solution of the Simultaneous Localization and Map Building Problem*. PhD thesis, University of Sydney. 28

[Nguyen et al., 2017] Nguyen, D. T., Hua, B.-S., Yu, L.-F., and Yeung, S.-K. (2017). A Robust 3D-2D Interactive Tool for Scene Segmentation and Annotation. *IEEE Transactions on Visualization and Computer Graphics (VGC)*, 24(12):3005–3018. 137

[Niemeyer et al., 2020] Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 101, 102

[Nießner et al., 2013a] Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013a). Real-time 3D Reconstruction at Scale using Voxel Hashing. In *Proceedings of SIGGRAPH*. 34

[Nießner et al., 2013b] Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013b). Real-time 3D Reconstruction at Scale using Voxel Hashing. *ACM Transactions on Graphics (TOG)*, 32(6):1–11. 137

[Nistér, 2004] Nistér, D. (2004). An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–777. 29

[Nistér et al., 2004] Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual Odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 29

[Nüchter et al., 2003] Nüchter, A., Surmann, H., Lingemann, K., and Hertzberg, J. (2003). Semantic scene analysis of scanned 3d indoor environments. In *Proceedings of the International Workshop on Vision, Modelling and Visualization (VMV)*. 38

[Okutomi and Kanade, 1993] Okutomi, M. and Kanade, T. (1993). A Multiple-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(4):353–363. 31

[Ortiz et al., 2022] Ortiz, J., Evans, T., Sucar, E., and Davison, A. J. (2022). Incremental abstraction in distributed probabilistic slam graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 23

[Park et al., 2019] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 43, 73, 113

[Parker et al., 1998] Parker, S., Shirley, P., Livnat, Y., Hansen, C., and Sloan, P. (1998). Interactive Ray Tracing for Isosurface Rendering. In *Proceedings of Visualization*. 33

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NIPS)*. 79, 111, 112

[Peng et al., 2020] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., and Geiger, A. (2020). Convolutional occupancy networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 113

[Pollefeys et al., 1999] Pollefeys, M., Koch, R., Vergauwen, M., and Van Gool, L. (1999). Hand-held acquisition of 3D models with a video camera. In *Proceedings of the IEEE International Workshop on 3D Digital Imaging and Modeling (3DIM)*. 28, 31

[Pollefeys et al., 2008] Pollefeys, M., Nistér, D., Frahm, J. M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S. J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewénius, H., Yang, R., Welch, G., and Towles, H. (2008). Detailed Real-Time Urban 3D Reconstruction from Video. *International Journal of Computer Vision (IJCV)*, 78(2-3):143–167. 31

[Prisacariu and Reid, 2011] Prisacariu, V. A. and Reid, I. (2011). Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 42

[Prisacariu and Reid, 2012] Prisacariu, V. A. and Reid, I. D. (2012). PWP3D: Real-time segmentation and tracking of 3d objects. *International Journal of Computer Vision (IJCV)*, 98(3):335–354. 42

[Prisacariu et al., 2013] Prisacariu, V. A., Segal, A. V., and Reid, I. (2013). Simultaneous monocular 2d segmentation, 3d pose recovery and 3d reconstruction. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 42

[Quigley et al., 2009] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. 81

[Ramesh et al., 2022] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*. 165

[Ramos and Ott, 2016] Ramos, F. and Ott, L. (2016). Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *International Journal of Robotics Research (IJRR)*, 35(14):1717–1730. 36, 165

[Ren et al., 2021] Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., and Wang, X. (2021). A survey of deep active learning. 54(9):1–40. 142

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, pages 91–99. 38

[Rho et al., 2023] Rho, D., Lee, B., Nam, S., Lee, J. C., Ko, J. H., and Park, E. (2023). Masked wavelet representation for compact neural radiance fields. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 165

[Rolnick et al., 2019] Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience replay for continual learning. In *Neural Information Processing Systems (NIPS)*. 113, 114

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An Efficient Alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE. 26

[Rusinkiewicz et al., 2002] Rusinkiewicz, S., Hall-Holt, O., and Levoy, M. (2002). Real-Time 3D Model Acquisition. In *Proceedings of SIGGRAPH*. 33

[Rusu et al., 2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*. 114

[Salas-Moreno et al., 2013] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H. J., and Davison, A. J. (2013). SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 40, 41, 161

[Scharstein and Szeliski, 2001] Scharstein, D. and Szeliski, R. (2001). A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision (IJCV)*, 47:7–42. 31

[Schops et al., 2019] Schops, T., Sattler, T., and Pollefeys, M. (2019). Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 34

[Schwarz et al., 2018] Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*. 114

[Settles, 2009] Settles, B. (2009). Active learning literature survey. 142

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905. 37

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good Features to Track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 26

[Shin et al., 2017] Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In *Neural Information Processing Systems (NIPS)*. 114

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 38, 96

[Sitzmann et al., 2020] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Neural Information Processing Systems (NIPS)*. 73, 113, 115

[Smith and Cheeseman, 1986] Smith, R. C. and Cheeseman, P. (1986). On the Representation and Estimation of Spatial Uncertainty. *International Journal of Robotics Research (IJRR)*, 5(4):56–68. 27, 113

[Sohn et al., 2015] Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Neural Information Processing Systems (NIPS)*. 88

[Song et al., 2015] Song, S., Lichtenberg, S. P., and Xiao, J. (2015). SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576. 149

[Strasdat et al., 2012] Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2012). Visual SLAM: Why filter? *Image and Vision Computing (IVC)*, 30(2):65–77. 30

[Straub et al., 2019] Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., et al. (2019). The Replica Dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*. 111, 120, 149

[Stückler et al., 2015] Stückler, J., Waldvogel, B., Schulz, H., and Behnke, S. (2015). Dense real-time mapping of object-class semantics from RGB-D video. *Journal of Real-Time Image Processing JRTIP*, 10(4):599–609. 39

[Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 112, 121

[Sucar et al., 2021] Sucar, E., Liu, S., Ortiz, J., and Davison, A. J. (2021). iMAP: Implicit Mapping and Positioning in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 23

[Sucar et al., 2020] Sucar, E., Wada, K., and Davison, A. (2020). NodeSLAM: Neural object descriptors for multi-view shape reconstruction. In *Proceedings of the International Conference on 3D Vision (3DV)*. 23

[System, ] System, R. O. ROS. URL: http://www.ros.org. 156

[Szeliski and Kang, 1993] Szeliski, R. and Kang, S. B. (1993). Recovering 3D Shape and Motion from Image Streams Using Non-Linear Least Squares. Technical report, Robotics Institude. 27

[Tan et al., 2018] Tan, Q., Gao, L., Lai, Y.-K., and Xia, S. (2018). Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 88

[Tancik et al., 2020] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Neural Information Processing Systems (NIPS)*. 73, 115

[Tang et al., 2020] Tang, D., Singh, S., Chou, P. A., Hane, C., Dou, M., Fanello, S., Taylor, J., Davidson, P., Guleryuz, O. G., Zhang, Y., et al. (2020). Deep implicit volume compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 73, 113

[Thrun, 1998] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*. 32

[Thrun, 2003] Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127. 35

[Torralba et al., 2004] Torralba, A., Murphy, K. P., and Freeman, W. T. (2004). Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 37

[Tsai et al., 2003] Tsai, A., Yezzi, A., Wells, W., Tempany, C., Tucker, D., Fan, A., Grimson, W. E., and Willsky, A. (2003). A shape-based approach to the segmentation of medical imagery using level sets. *IEEE transactions on medical imaging*, 22(2):137–154. 42

[Tulsiani et al., 2018] Tulsiani, S., Efros, A. A., and Malik, J. (2018). Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 162

[Tulsiani et al., 2017] Tulsiani, S., Zhou, T., Efros, A. A., and Malik, J. (2017). Multi-View Supervision for Single-View Reconstruction via Differentiable Ray Consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 84

[Valentin et al., 2015] Valentin, J., Vineet, V., Cheng, M.-M., Kim, D., Shotton, J., Kohli, P., Nießner, M., Criminisi, A., Izadi, S., and Torr, P. (2015). Semanticpaint: Interactive 3d labeling and learning at your fingertips. *ACM Transactions on Graphics*, 34(5). 39, 137, 138, 145, 147

[Vespa et al., 2018] Vespa, E., Nikolov, N., Grimm, M., Nardi, L., Kelly, P. H., and Leutenegger, S. (2018). Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*. 34, 112

[Vidal et al., 2001] Vidal, R., Ma, Y., S., H., and Sastry, S. (2001). Optimal Motion Estimation from Multiview Normalized Epipolar Constraint. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 27

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 37

[Wada et al., 2020] Wada, K., Sucar, E., James, S., Lenton, D., and Davison, A. J. (2020). Morefusion: Multi-object reasoning for 6d pose estimation from volu-

metric fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 23

[Wang et al., 2021a] Wang, J., Rünz, M., and Agapito, L. (2021a). Dsp-slam: object oriented slam with deep shape priors. In *Proceedings of the International Conference on 3D Vision (3DV)*. 162

[Wang et al., 2018] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G. (2018). Pixel2Mesh: Generating 3D Mesh Models from Single RBG Images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67. 84

[Wang et al., 2021b] Wang, Z., Wu, S., Xie, W., Chen, M., and Prisacariu, V. A. (2021b). NeRF–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*. 113

[Weder et al., 2020] Weder, S., Schonberger, J., Pollefeys, M., and Oswald, M. R. (2020). RoutedFusion: Learning real-time depth map fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 113

[Whelan et al., 2015] Whelan, T., Leutenegger, S., Salas-Moreno, R. F., Glocker, B., and Davison, A. J. (2015). ElasticFusion: Dense SLAM Without A Pose Graph. In *Proceedings of Robotics: Science and Systems (RSS)*. 34, 84, 112

[Whelan et al., 2012] Whelan, T., McDonald, J. B., Kaess, M., Fallon, M., Johannsson, H., and Leonard, J. J. (2012). Kintinuous: Spatially Extended KinectFusion. In *Workshop on RGB-D: Advanced Reasoning with Depth Cameras, in conjunction with Robotics: Science and Systems*. 127

[Williams and Fitzgibbon, 2006] Williams, O. and Fitzgibbon, A. (2006). Gaussian process implicit surfaces. In *Gaussian Processes in Practice*. 164

[Wu et al., 2017] Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, B., and Tenenbaum, J. (2017). Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Neural Information Processing Systems (NIPS)*. 84

[Wu et al., 2016] Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Neural Information Processing Systems (NIPS)*. 43

[Wu et al., 2015] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 84, 103

[Wurm et al., 2010] Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., and Burgard, W. (2010). OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In *Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*. 34

[Xia et al., 2022] Xia, J., Li, S., Huang, J., Yang, Z., Jaimoukha, I. M., and Gündüz, D. (2022). Metalearning-based alternating minimization algorithm for nonconvex optimization. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15. 137

[Xie et al., 2022] Xie, C., Mousavian, A., Xiang, Y., and Fox, D. (2022). RICE: Refining instance masks in cluttered environments with graph neural networks. In *Conference on Robot Learning*, pages 1655–1665. PMLR. 157

[Xie et al., 2021] Xie, C., Xiang, Y., Mousavian, A., and Fox, D. (2021). Unseen object instance segmentation for robotic environments. *IEEE Transactions on Robotics*, 37(5):1343–1359. 157

[Xie and Tu, 2015] Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1395–1403. 138

[Xu et al., 2022] Xu, B., Davison, A. J., and Leutenegger, S. (2022). Learning to complete object shapes for object-level mapping in dynamic scenes. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 162

[Yang et al., 1993] Yang, Y., Yuille, A., and Lu, J. (1993). Local, global, and multi-level stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 31

[Yen-Chen et al., 2020] Yen-Chen, L., Florence, P., Barron, J. T., Rodriguez, A., Isola, P., and Lin, T.-Y. (2020). iNeRF: Inverting neural radiance fields for pose estimation. *arXiv preprint arXiv:2012.05877*. 113

[Yingze Bao et al., 2013] Yingze Bao, S., Chandraker, M., Lin, Y., and Savarese, S. (2013). Dense object reconstruction with semantic priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 42

[Zhi et al., 2021] Zhi, S., Laidlow, T., Leutenegger, S., and Davison, A. J. (2021). In-Place Scene Labelling and Understanding with Implicit Scene Representation. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 140

[Zhi et al., 2022] Zhi, S., Sucar, E., Mouton, A., Haughton, I., Laidlow, T., and Davison, A. J. (2022). ilabel: Revealing objects in n. *IEEE Robotics and Automation Letters*. 23, 134

[Zhou et al., 2018] Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*. 103

[Zhu et al., 2022] Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M. R., and Pollefeys, M. (2022). Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 163

[Zienkiewicz et al., 2016] Zienkiewicz, J., Davison, A. J., and Leutenegger, S. (2016). Real-Time Height-Map Fusion using Differentiable Rendering. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 35

[Zitnick and Kanade, 2000] Zitnick, C. L. and Kanade, T. (2000). A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 31