

Leevi Raivio

# VISUAL METRIC-SEMANTIC 3D RECONSTRUCTION

Master of Science Thesis  
Faculty of Engineering and Natural Sciences  
Examiners: Prof. Esa Rahtu  
Prof. Risto Ritala  
May 2021

## ABSTRACT

Leevi Raivio: Visual metric-semantic 3D reconstruction  
Master of Science Thesis  
Tampere University  
Degree Programme in Automation Engineering, MSc (Tech)  
May 2021

---

While people and animals understand their surroundings almost effortlessly, the problem is really hard to solve for machines. To understand their environment comprehensively, one needs to capture both spatial relations and semantic meaning from their surroundings and incorporate them into a coherent model of the environment. To apply this knowledge, one also needs to be able to relate new information they sense to the model, and update it accordingly. Although there are no complete answers to the problem, parts of it can already be solved and research on related subjects seems to only accelerate. With recent advances in relevant research areas, machines are able to generate increasingly general representations of the environment.

This master's thesis studies metric-semantic reconstruction from the perspective of visual data, using dense reconstruction of indoor environments as an example use-case. Related background information and theory are studied, and a baseline end-to-end three-dimensional metric-semantic reconstruction system is designed and evaluated. The purpose is to create a platform to base future research on and to find interesting topics to study.

Simultaneous localisation and mapping (SLAM) methods are used in this work to track the device pose and make reconstruction also possible in environments where localisation infrastructure or pre-existing maps are not available. On the other hand, panoptic segmentation is applied to incorporate rich semantic meaning into metric reconstructions. A view-based segmentation approach is chosen to render the system more robust to uncertainties related to visual data. The RTAB-Map library is applied for globally consistent three-dimensional metric SLAM, while EfficientPS is chosen as the panoptic segmentation approach. Individual components of the system are evaluated quantitatively, after which end-to-end results are generated with data captured from two indoor campus environments and analysed qualitatively.

The essential building blocks of a metric-semantic reconstruction system are specified and choices related to each are compared. Based on results, possible performance bottlenecks are identified. Improvements to existing methods are discussed, and possible future research topics are assessed as well. Although the approach is quite simple, and in some aspects can not match the most recent works on the field, it provides a strong baseline for future research.

Keywords: Metric-semantic reconstruction, scene understanding, SLAM, panoptic segmentation

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

# TIIVISTELMÄ

Leevi Raivio: Visuaalinen metrinen ja semanttinen 3D rekonstruktio  
Diplomityö  
Tampereen yliopisto  
Automaatiotekniikan DI-ohjelma  
Toukokuu 2021

---

Ihmiset ja eläimet kykenevät ymmärtämään ympäristöään lähes vaivattomasti. Koneille se on kuitenkin erittäin vaativa tehtävä. Ymmärtääkseen ympäristöään kokonaisvaltaisesti, täytyy havaitsejan ymmärtää paitsi avaruudellisia suhteita myös semanttisia merkityksiä ja sisällyttää molempia yhtenäiseen malliin. Jotta kerättyä tietoa voidaan hyödyntää, tarvitaan myös keinoja verrata sitä luotuun malliin ja vastaavasti mallin päivittämiseen tiedon avulla. Vaikka kokonaisvaltaisia vastauksia ongelmaan ei ole vielä keksitty, osia siitä pystytään jo ratkaisemaan. Relevanttien alojen tutkimus näyttäisi olevan kiihtymisvaiheessa, ja viimeaikaisten edistysaskelien ansiosta koneet kykenevät luomaan entistä yleisempiä malleja ympäristöistään.

Tämä diplomityö käsittelee metrisen ja semanttisen informaation yhdistämistä kolmiulotteisessa mallinnuksessa visuaalisen datan näkökulmasta. Esimerkkitapauksena käytetään sisätilojen tiheää mallinnusta. Työssä esitellään aiheeseen liittyvää taustatietoa ja teoriaa, ja suunnitellaan niiden pohjalta mallinnusjärjestelmä lähtökohdaksi jatkotutkimuksille. Tavoitteena on tuottaa alusta tutkimustarkoituksiin, sekä löytää uusia mielenkiintoisia tutkimusaiheita.

Jotta mallinnus olisi mahdollista ilman olemassa olevaa infrastruktuuria tai valmiita malleja, työssä seurataan kameran asentoa ja sijaintia samanaikaisen paikannuksen ja kartoituksen menetelmillä. Semanttista informaatiota sulautetaan metrisiin rekonstruktioihin panoptisen segmentoinnin menetelmillä. Järjestelmässä hyödynnetään kuvakohtaista segmentointia visuaaliseen dataan liittyvien epävarmuuksien vaikutusten minimoimiseksi. RTAB-Map -ohjelmakirjastoa käytetään globaalisti yhtenäisten mallien luomiseen, ja panoptisen segmentoinnin menetelmäksi on valittu EfficientPS. Yksittäisiä komponentteja arvioidaan kvantitatiivisesti, jonka jälkeen koko järjestelmän lopputuloksia tutkitaan kvalitatiivisesti luomalla malleja kahdesta eri kampusympäristöstä.

Työssä esitellään vastaavaan järjestelmään tarvittavat olennaisimmat komponentit, ja vertaillaan eri vaihtoehtoja. Suorituskykyyn liittyviä pullonkauloja selvitetään myös tuloksien pohjalta, jonka jälkeen tutkitaan parannuksia olemassa oleviin menetelmiin ja tarkastellaan mahdollisia uusia tutkimusaiheita. Vaikka työn lähestymistapa on melko yksinkertainen, ja suunniteltu järjestelmä ei joissain tapauksissa kykene saavuttamaan samaa tasoa alan uusimpien menetelmien kanssa, tarjoaa se silti vahvan lähtökohdan jatkotutkimuksille.

Avainsanat: Ympäristön tulkinta, samanaikainen paikannus ja kartoitus, panoptinen segmentointi

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

## PREFACE

I would like to thank my supervisor associate professor Esa Rahtu for this MSc topic and the opportunity to work on such an interesting project, as well as my former supervisor Heikki Huttunen for providing opportunities to gain the skills and knowledge applied in this thesis. I also would like to thank professor Risto Ritala for examining this thesis and providing valuable feedback. Many thanks to the former Machine Learning Group and other colleagues at the University of Tampere as well for the delightful working environment and inspiring company.

Tampereen Teekkarien Hiihto- ja Purjehdusseura and the Guild of Automation Technology provided me a second and tertiary home in times when global pandemics were not as common, as well as many memorable trips and events, of which I am very thankful. The many events and travels might be a reason to why I didn't graduate exactly in the five-year schedule determined by the Finnish government, but they certainly made the years much more enjoyable. Although my university studies are coming to an end, I hope to keep in touch with the friends I made along the way in the future as well.

I also owe huge thanks to my friends and family for their support during all these years.

Tampere, 17th May 2021

Leevi Raivio

# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Visual metric-semantic reconstruction</b>	<b>3</b>
2.1	Visual mapping in an unknown environment . . . . .	4
2.1.1	The simultaneous localisation and mapping problem . . . . .	4
2.1.2	Estimating full SLAM with graph-based nonlinear optimisation . . . . .	9
2.1.3	Visual simultaneous localisation and mapping . . . . .	11
2.2	View-based scene segmentation . . . . .	15
2.2.1	A brief introduction to convolutional neural networks . . . . .	16
2.2.2	Panoptic segmentation . . . . .	20
2.2.3	Image segmentation methods . . . . .	23
2.2.4	Extending image segmentation to three-dimensional scenes . . . . .	29
<b>3</b>	<b>Applied methods</b>	<b>32</b>
3.1	RTAB-Map: Real-Time Appearance-Based Mapping . . . . .	33
3.1.1	Odometry options . . . . .	33
3.1.2	Visual vocabulary and node generation . . . . .	34
3.1.3	Loop closure tracking with a discrete Bayesian filter . . . . .	35
3.1.4	Memory management and graph optimisation . . . . .	36
3.1.5	Reconstruction . . . . .	37
3.2	EfficientPS: Efficient Panoptic Segmentation . . . . .	38
3.2.1	Two-way FPN feature extractor . . . . .	39
3.2.2	Semantic segmentation head . . . . .	40
3.2.3	Instance segmentation head . . . . .	41
3.2.4	Panoptic fusion . . . . .	42
<b>4</b>	<b>System design and performance evaluation</b>	<b>44</b>
4.1	Hardware . . . . .	45
4.2	Design choices and general structure . . . . .	46
4.3	Odometry performance . . . . .	47
4.4	SLAM performance . . . . .	48
4.5	Segmentation performance . . . . .	50
<b>5</b>	<b>Application in practice</b>	<b>55</b>
5.1	3D segmentation of the ScanNet Dataset . . . . .	56
5.2	Indoor reconstruction of campus environments . . . . .	60

<b>6 Possible improvements based on results</b>	<b>66</b>
6.1 Hardware and localisation . . . . .	66
6.2 Training data . . . . .	67
6.3 Segmentation models . . . . .	68
<b>7 Conclusion</b>	<b>69</b>
<b>References</b>	<b>71</b>
<b>Appendix A A confusion matrix of ScanNet results</b>	<b>83</b>
<b>Appendix B MSEG dataset classes</b>	<b>84</b>
<b>Appendix C More images from practical applications</b>	<b>85</b>

## LIST OF FIGURES

2.1	Graphical model of an agent’s movement in the environment . . . . .	5
2.2	An example on localisation uncertainty in a known environment . . . . .	6
2.3	Localisation uncertainty in an unknown environment . . . . .	7
2.4	An example of a visual odometry algorithm . . . . .	13
2.5	The structure of a multi-layer perceptron . . . . .	16
2.6	Two-dimensional convolution with a finite-sized kernel . . . . .	18
2.7	A simple CNN designed to classify different vehicles . . . . .	19
2.8	Intuition behind different image segmentation approaches . . . . .	21
2.9	Topology of a Fully Convolutional Network for semantic segmentation . . .	24
2.10	An example on the significance of global context . . . . .	24
2.11	SegNet: a fully convolutional encoder-decoder network . . . . .	25
2.12	Pyramid approaches to capture multi-scale context in image segmentation	25
2.13	Parallel convolution streams of High-Resolution Network . . . . .	26
2.14	Topology of Faster R-CNN and Mask R-CNN . . . . .	27
2.15	Panoptic FPN and Panoptic-Deeplab . . . . .	28
2.16	The pinhole camera model . . . . .	29
2.17	An example of common 3D representations on the Stanford Bunny . . . . .	30
3.1	Block diagram of RTAB-Map . . . . .	33
3.2	Topology of the EfficientPS panoptic segmentation architecture . . . . .	38
3.3	The two-way FPN feature extractor of EfficientPS . . . . .	39
3.4	Semantic segmentation head of EfficientPS . . . . .	40
3.5	Instance segmentation head of EfficientPS . . . . .	42
3.6	The panoptic fusion module of EfficientPS . . . . .	43
4.1	The camera system used for indoor reconstruction . . . . .	45
4.2	Block diagram of the end-to-end system . . . . .	47
5.1	Colours associated with ScanNet evaluation classes . . . . .	56
5.2	A scene from ScanNet training set . . . . .	56
5.3	EfficientPS segmentation of the ScanNet training scene . . . . .	57
5.4	HRNet segmentation of the ScanNet training scene . . . . .	57
5.5	A scene from ScanNet test set . . . . .	58
5.6	An individual view from the ScanNet test scene . . . . .	59
5.7	2D Occupancy grids generated with the indoor reconstruction system. . .	60
5.8	An indoor scene reconstructed with Realsense D455 . . . . .	61

5.9	A comparison of depth image quality of applied cameras . . . . .	62
5.10	A lobby of of a university building reconstructed with Azure Kinect . . . . .	63
5.11	2D segmentation of two views from the first dataset . . . . .	65
5.12	2D segmentation of two views from the second dataset . . . . .	65
A.1	Confusion matrix of 3D evaluation on the ScanNet validation set . . . . .	83
B.1	Colours associated with the MSEG dataset . . . . .	84
C.1	Scene 0317_01 from ScanNet validation set . . . . .	85
C.2	Scene 0362_00 from ScanNet validation set . . . . .	86
C.3	Another indoor scene reconstructed with Realsense D455 . . . . .	87
C.4	Another indoor scene reconstructed with Azure Kinect . . . . .	88
C.5	More 2D comparisons on the ScanNet validation set . . . . .	89



## LIST OF TABLES

4.1 SLAM accuracy comparison on the EuRoC MAV dataset . . . . .	49
4.2 Panoptic segmentation performance comparison on the Cityscapes test set	51
4.3 Panoptic quality of individual classes on the ScanNet validation set . . . . .	52
4.4 Average panoptic quality on the ScanNet validation set . . . . .	52
4.5 Semantic segmentation performance on the ScanNet test set . . . . .	53

## LIST OF SYMBOLS AND ABBREVIATIONS

AR	Augmented Reality
BA	Bundle Adjustment
BoW	Bag-of-Words
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DL	Deep Learning
DoF	Degrees-of-Freedom
EKF	Extended Kalman Filter
FCN	Fully Convolutional Network
FLOP	FLoating-point OPeration
IoU	Intersection over Union
MLP	Multi-Layer Perceptron
MTL	Multi-Task Learning
NMS	Non-Maxima Suppression
PF	Particle Filter
PQ	Panoptic Quality
RANSAC	RANdom SAmples Consensus
RGB-D	Red, Green, Blue - Depth
RoI	Region of Interest
RQ	Recongnition Quality
SfM	Structure from Motion
SGD	Stochastic Gradient Descent
SLAM	Simultaneous Localisation And Mapping
SQ	Segmentation Quality
TSDF	Truncated Signed Distance Field
VIO	Visual-Inertial Odometry
VO	Visual Odometry

# 1 INTRODUCTION

An intelligent agent acquires information of its surroundings through perception and applies it to interact with the environment. A person or an animal does this instinctively. For example, a person could look for landmarks in the environment to infer their location, and an animal looks for things in its surroundings that are edible. People can search for tools useful for solving a specific task, based on the knowledge where they will most likely be found. Artificial agents – machines – capable of understanding the environment at the same level are not yet a reality, but have been envisioned extensively. A more sophisticated understanding of the environment would give them the ability to operate with a higher degree of autonomy more similar to people, to interact with their environment in more complex ways and to communicate with us more effectively, through concepts we understand more naturally.

Perception of the environment can be divided into spatial and semantic information. Spatial information itself can be further divided into topological and metric information: relative locations of parts of the environment and the things in it and the geometrical, measurable relations and shapes. Semantic information is perhaps the more abstract of the two: the meanings of things or their applications, for instance. Geometry and topology give a machine the capability to autonomously navigate an environment, while a machine capable of understanding semantics as well would be able to infer more from its environment and generate a more general understanding of its surroundings. Applications that would greatly benefit from a semantic understanding of the environment include – for example – augmented reality, autonomous driving and interactive robotics. Semantic information is quite difficult to capture explicitly, and the nature of the information itself is dependent on the perceiver. For example, people of different cultures could give different meanings to things. Colour could give necessary information of the environment, which would be completely lost if one could not perceive colour. The value of the information is also dependent on the context: speed limits are crucial information for an autonomous car, while for a pedestrian they are not as important.

This thesis explores the subject of three-dimensional metric-semantic reconstruction from the perspective of visual perception. The presented use-cases concern mostly dense reconstruction of static indoor environments and relatively closed spaces, but the methods applied in this work could in principle be extended to outdoors use-cases and other larger

open spaces as well. Visual data is especially interesting in this context because of its rich representation of the environment. The vast amounts of information in complex visual data bring about new challenges to the field, however. Even modern computers cannot utilise all of the information available in high-resolution images and videos in real time, and the details relevant to the task at hand might easily get lost among the huge amount of data. To overcome these challenges, it is therefore essential to both extract relevant information from the visual data, as well as to process it efficiently. Owing to recent advances in camera technology, computers and smart devices, the hardware to both acquire and process visual data has become quite affordable. In fact, the practical examples presented in this thesis are produced with relatively cheap consumer-grade cameras.

An end-to-end system for visual metric-semantic reconstruction is designed, evaluated and applied to real-world use-cases in this work. The practical work is supported by a study on the theoretical background and current research related to the task. The purpose of this thesis is to be groundwork for research considering metric-semantic reconstruction. With the knowledge of the current state of research, as well as practical knowledge gained from the design process, we aim to gain more insight on open research questions and possible research directions to take in the future. The key questions this work aims to answer are:

- What are the basic building blocks of a metric-semantic reconstruction system?
- How to choose the best options for practical applications?
- What are the essential performance bottlenecks?
- To what possible directions could one take this research in the future?

The rest of this work is divided into following chapters. Chapter 2 discusses the theory related to the system designed in this work, namely visual mapping in an unknown environment and view-based scene segmentation. Chapter 3, on the other hand, introduces the methods used in practical applications. In Chapter 4, the overall structure of the system, as well as key design choices related to it are described. Subsequently, the performance of the designed system is also evaluated quantitatively to justify the choices and ensure the system works as intended. End-to-end results of practical applications of the system are analysed qualitatively in Chapter 5 to assess its performance in real environments. Based on the results of earlier chapters, possible changes and improvements to the system are discussed in Chapter 6. Finally, Chapter 7 summarises key findings, answers the questions posed above and concludes the thesis.

## 2 VISUAL METRIC-SEMANTIC RECONSTRUCTION

An agent moving in an environment needs to know its location to reach a destination efficiently. The location itself is in relation to the environment, thus information about the environment is needed to know one's location in it. On the other hand, the environment could be previously unknown to the agent, who therefore needs to be able to accumulate information from it through perception. This is called the **environment mapping problem**. It has been defined as one of the key issues in navigation and mobile robotics. [1] To navigate, the agent needs a spatial model – a map – of the environment and a way to determine its location in it. Being able to localise and navigate without pre-existing information gives it more freedom to explore: a person is not tied only to areas which they know beforehand, and similarly, a robot capable of exploring on its own could be used in a wider range of applications. Going forward, mapping is defined as generating a reference for localisation, while reconstruction applies maps to produce the desired end result in the context of this thesis: a detailed reconstruction of the environment.

In addition to **spatial information**, **semantic information** of the environment is also useful in knowing one's location. Humans use semantic information instinctively: a house of certain shape and colour might tell you what street you are in, and familiar furniture might suggest you are in an indoor environment you have been to before. In [2], the connection between human cognition and robot navigation was studied, and the idea of cognitive mapping for robots was discussed across different fields of research. The ability of robots to understand their surroundings through both spatial and semantic information would result in more robust localisation, as well as their ability to interact with the environment better. Machines could also communicate with people about the environment in a more efficient and familiar way to us, via semantic meaning in addition to spatial information.

This chapter is an introduction on how to acquire metric and semantic information through visual perception. Because the topic is quite broad, discussion has to be kept quite general. However, a large number of references are given to allow the interested reader to study individual subjects more thoroughly. Section 2.1 studies the problem of generating a spatial map of a previously unknown environment, especially from the perspective of visual perception, while Section 2.2 introduces ways to capture dense semantic information from images and fuse it with metric data to form a metric-semantic reconstruction.

## 2.1 Visual mapping in an unknown environment

The need to both generate a map and localise oneself in it simultaneously – called the **simultaneous localisation and mapping problem** (SLAM) – renders exploring a previously unknown environment a really challenging task. The mathematical framework behind SLAM is very well established, but there are still a lot of unanswered questions regarding practical implementations. [3, 4, 5, 6] Complex multi-dimensional data from sensors like cameras and laser scanners – while enabling more detailed representations of the environment – also provide new challenges to solve. [7] This section aims to introduce the most essential concepts related to mapping in an unknown environment in general, as well as specific knowledge required to apply visual data for the task.

Considering visual data, SLAM is quite close to the Structure from Motion (SfM) problem. [8, 9] However, while SfM is largely considered an offline task, SLAM approaches also pursue incremental reconstruction and real-time localisation. To allow for a wider range of possible applications, and to limit the scope of the thesis, this work focuses on visual SLAM approaches.

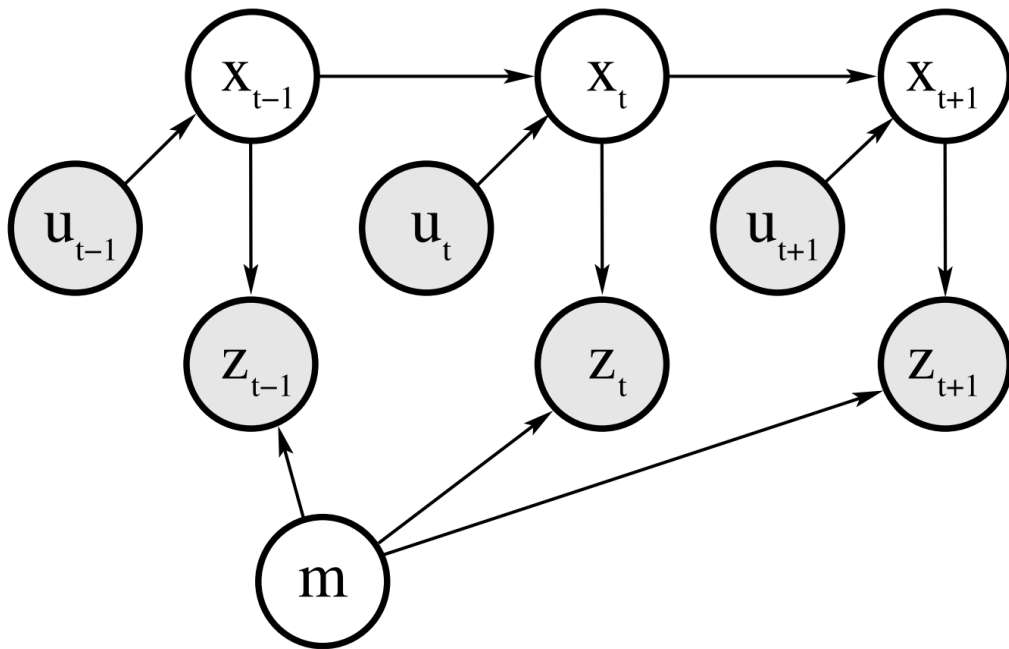
The rest of this section is structured as follows. The mathematical premises behind simultaneous localisation and mapping are first introduced in 2.1.1, after which the most prominent way to apply SLAM using visual data – graph-based nonlinear optimisation – is introduced briefly in 2.1.2. Finally, these concepts are considered from the viewpoint of visual data in 2.1.3.

### 2.1.1 The simultaneous localisation and mapping problem

From the perspective of an agent moving in an environment previously unknown to it, simultaneous localisation and mapping can be defined as follows. The agent's starting position is known, but no pre-existing map nor localisation infrastructure is available to determine its position going forward. Therefore, a map needs to be acquired in order to have a reference of the agent's position in the environment. [10, p. 982] Since the mapping process itself needs the location of the agent, it needs to concurrently build the map of the environment while localising itself in it at the same time. Usually, the agent cannot directly sense its location, but has to estimate it in relation to its **actions** and **perception** of the environment. The agent forms a map of its surroundings based on the information perceived by it and expands it by exploring the environment. Later – when revisiting previous locations – it can localise itself with the map.

The agent's movement in an unknown environment can be described as a sequence of poses, which form a **path**. Moreover, this can be modelled as a discrete time-series of form [3]

$$X_T = \{x_0, x_1, x_2, \dots, x_T\}, \quad (2.1)$$



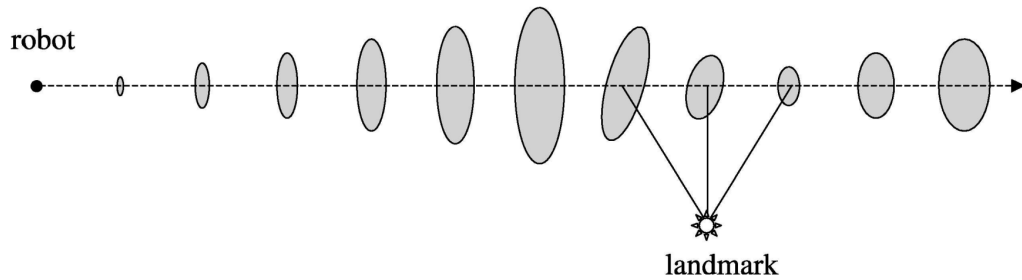
**Figure 2.1.** Graphical model of an agent's movement in the environment. [3] Odometry  $u$  and measurements  $z$  are directly observable by the agent, who tries to infer their state  $x$  and the map  $m$ . Arrows represent causal relationships between the agent's states and other nodes.

where  $x_t$  defines the agent's location and pose – or **state** – at time  $t$ ,  $T$  being some terminal time. The initial state  $x_0$  is assumed known, while further states cannot be measured directly.

**Odometry** can be used to estimate where the agent is relative to an earlier state. [3] An estimate of the path can then be acquired by concatenating the relative positions over time. The integration of a path solely from an agent's previous locations and actions is also known as **dead reckoning**. [11] In a robotics case, predictions might be calculated from wheel odometers, motor controls or inertial measurements, for example. In visual SLAM, movement could be predicted from perceived camera movements. Similarly, humans use their eyes, vestibular system and muscle feedback to estimate where they are going. Odometry readings can be modelled as a discrete time-series [3]

$$U_T = \{u_0, u_1, u_2, \dots, u_T\}, \quad (2.2)$$

where the reading at time  $t$  is  $u_t$ . The path can be incrementally predicted from odometry with a **motion model**, which describes the relationship between odometry readings and movement. [12, p. 91-118] If no uncertainties were involved, the predicted path would be equal to the actual path of the agent. In the real world, however, some phenomena are always left unexplained by the model. For example, measurement noise can only be approximated and the outcomes of actions might differ from what was anticipated. Since the path is predicted incrementally, errors are integrated into the process as well, and so



**Figure 2.2.** An example on localisation uncertainty in a known environment. [3] The ellipses illustrate probability distribution of a robot's state at each timestep. As it moves in the environment, state uncertainty grows because of odometry drift, until it senses a known landmark and thus becomes more sure of its position.

the **uncertainty** of the **state estimate** grows with each step. A person walking blindfolded becomes more uncertain of their position the further they walk, given no other feedback is available. This phenomenon is called **odometry drift**. [12, p. 107-112][13, p. 27-46]

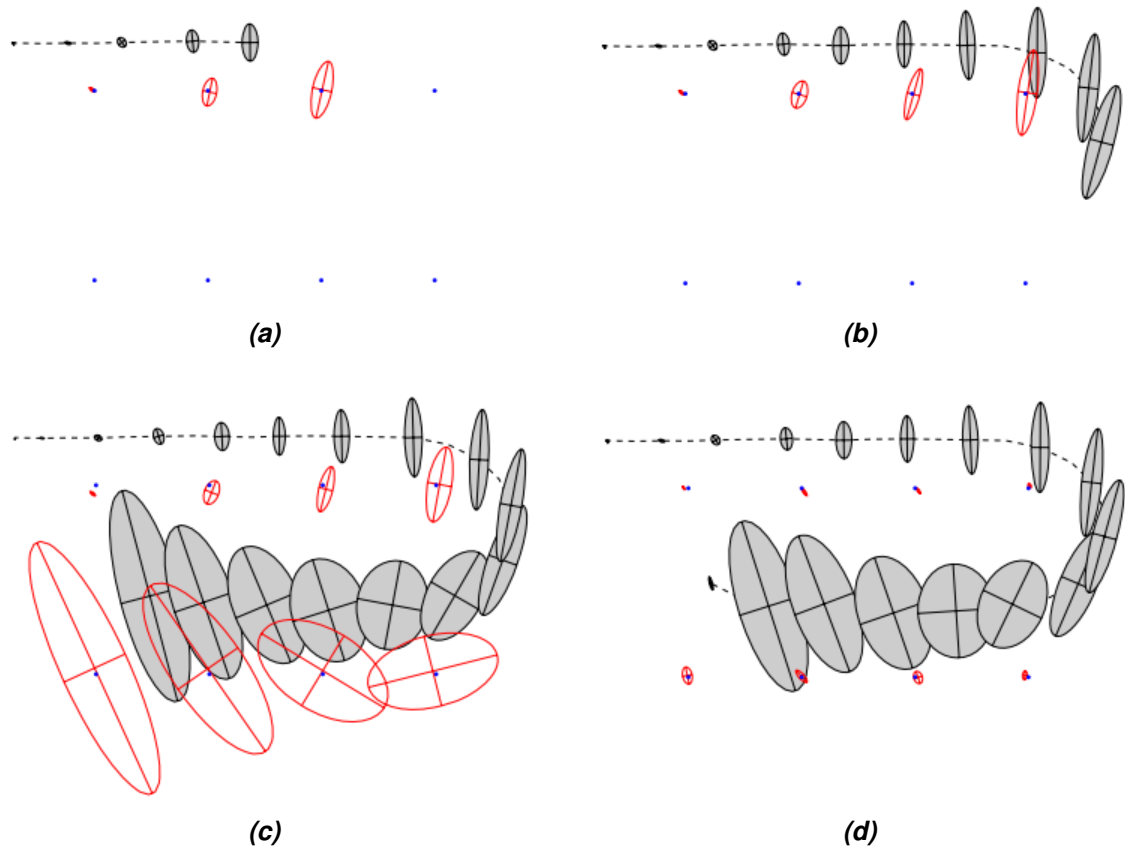
**Measurements** provide the agent information about the environment relative to its current state. They can also be represented sequentially, as a discrete time-series

$$Z_T = \{z_0, z_1, z_2, \dots, z_T\}, \quad (2.3)$$

where the  $z_t$  is the set of measurements at time  $t$ . [3] The set could be empty – if there are no measurements at a certain time – or it might have an arbitrary number of measurements depending on the perceptive capabilities of the agent and the complexity of the environment. The relationship between measurements and the agent's state is described with a **measurement model**. [12, p. 121-154] Figure 2.1 visualises the relationships between the agent's state, odometry, measurements and the map.

Measurements are used to decrease the uncertainty of the agent's state estimate. [12, p. 157-170][13, p. 27-46] While the agent's estimate grows more uncertain with odometry predictions, it becomes more sure of its location with measurements corresponding to a location in the map. While walking in the woods, a person might look for landmarks and locate them on a map to be more certain of their own location. The development of uncertainty in a known environment is visualised in Figure 2.2. However – contrary to localisation in a known environment – the uncertainties related to feature locations grow with state uncertainty in an environment previously unknown to the agent until evidence of their relative locations can be retrieved. This is because the agent only knows their location in relation to itself. Figure 2.3 depicts uncertainty in a previously unknown environment. An agent's uncertainty of its location in relation to its starting point grows due to odometry drift, as well as its uncertainty of the perceived landmarks' locations. When a known landmark is perceived again, the agent gains knowledge of the amount of drift by comparing the landmarks' location in the map to its perceived location. This new





**Figure 2.3.** Visualisation of localisation uncertainty in an unknown environment. [3] In (a)-(c) an agent moves in the environment, predicting its and the landmarks' position with odometry. Uncertainty of the agent' location as well as the landmarks grow, until in (d) the agent senses a previous landmark, and thus becomes more sure of its current location and the map.

information can then be applied to improve the current state estimate and the map.

If the dimensionality of raw measurements grows too high to manage in a reasonable time with limited computational resources, they can be reduced to a lower-dimensional feature space with **feature extraction** techniques. [12, p. 147-155] A lot of information will be lost, but, on the other hand, the data becomes easier to process. This is usually the case with visual applications, where *e.g.* corners, edges or local feature descriptors [14, 15] are extracted from raw pixel data. However, feature extraction is not a straightforward task, since it requires one to preserve as much valuable information as possible while discarding enough data to reduce dimensions effectively.

A key question related to measurements and feature extraction is the **data association problem**: does a measurement or perceived feature represent some feature already found in the map, or a completely new one? [12, p. 152][10, p. 599] If the correspondence between detected features is known, the task becomes simply to find the most similar match for the feature from the map, and accept or reject it based in how similar it is. With unknown correspondence, the problem becomes harder: the relative locations of

features and the agent need to be used to determine the feature’s identities. If a feature is perceived close to a known feature in the map, it could be the same one. If it is sensed in the same location multiple times, the uncertainty of its location decreases. Both approaches are prone to generate wrong associations, which in turn appear as errors in the map. Errors in the association make the problem quite hard to solve. [3]

In the example of Figure 2.3, the agent only updates the current state estimate, foregoing the correction of its full path. This is the goal of **online SLAM** methods. [3] Online SLAM requires less computation than maintaining the full history, and as such is useful if the use-case only requires the current state to be known accurately, for example when navigating from one point to another. However, if a detailed reconstruction of the environment is to be generated, a **full SLAM** solution is more useful. By maintaining the full path estimate and saving all measurements, the generated map – and in extension, the reconstruction – is more accurate and consistent. In early research, full SLAM was largely considered an offline problem, but lately – with ever-increasing computational resources and clever memory management – many approaches are able to solve it in real time. [16, 17, 18] Since this thesis is focused on reconstruction, the full SLAM problem is addressed in more detail. The posterior probability distribution of full SLAM is defined as [12, p. 246]

$$p(X_T, m|U_t, Z_t). \quad (2.4)$$

It contains all the necessary information about the path  $X_T$  and the map  $m$ . In theory, the posterior distribution can be solved explicitly, since odometry  $U_T$  and measurements  $Z_T$  are observable, but it is usually infeasible in practice even in online SLAM problems where only the last state  $x_t$  has to be solved. The dimensionality of the problem grows fast with time, and so the distribution has to be approximated in any practical application.

In many environments – especially in larger scale – it could be possible to reach the same location from multiple directions. Determining whether a previously visited location is reached is essential for both topologically and metrically accurate mapping. Such an event is called a loop closure, and the task to be solved is called the **loop closing problem**. [3] There is always some drift present in SLAM due to the measurements always being relative to the estimated location, and so without loop closure the same location reached from different directions could appear in different locations in the map. Therefore, loop closure is necessary to retain topological consistency in the maps generated by a SLAM algorithm. [6] Nowadays, a SLAM algorithm is only considered complete if it contains a loop closing mechanism. [3, 6, 19] A visual example of a loop closure can be seen in the example of Figure 2.3: an agent detects a previously sensed location and corrects the map accordingly. Loop closure is also closely related to **place recognition** techniques and the **kidnapped robot problem**, which address the problem of an agent being placed in an unknown location in a possibly already familiar environment. This

could happen, for example, when mapping is performed in multiple sessions or the agent needs to recover from getting lost. They would then need to determine if they indeed are in a previously known location to be able to utilise previous maps without outside help.

A key challenge in loop closure detection is how to handle **false positives**: erroneous loop closures that the detection algorithm determines correct. Because false positives cannot be directly detected, some form of belief distribution is usually generated to track the probabilities of loop closure candidates. [20] An estimate of the probability of the loop closure being correct is maintained based on previous detections and the state estimate and a threshold for the probability of a detection being false is chosen based on the application. The choice is a trade-off between **precision** – the ability to filter out false positives – and **recall** – the ability to find more true positives. False positives generate topological errors in the map, but on the other hand – if the algorithm is too restrictive – a failure to detect a correct loop closure could result in errors as well.

### 2.1.2 Estimating full SLAM with graph-based nonlinear optimisation

There are three fundamental approaches to SLAM, on which most other approaches are based: extended Kalman filters, particle filters and graph-based nonlinear optimisation methods. Graph-based methods are the most popular solution to the full SLAM problem, thus being most interesting in the scope of this thesis. The others are first introduced briefly to understand the fundamental differences, after which graph optimisation methods are discussed in more detail.

The earliest SLAM methods were based on **Extended Kalman Filters** (EKF), where the state and map are estimated with a multivariate Gaussian distribution [3, 21, 22]

$$p(x_t, m | Z_t, U_t) \sim \mathcal{N}(\mu_t, \Sigma_t). \quad (2.5)$$

The estimated state, as well as map feature estimates, are contained in the state vector  $\mu_t \in \mathcal{R}^n$ , while the covariance matrix  $\Sigma_t \in \mathcal{R}^{n \times n}$  contains their relative uncertainties. The number of state variables  $n$  – *i.e.* number of states and map features – increases when new features are discovered. Because the covariance matrix of a Kalman Filter grows quadratically with additional states – and computational cost as well as memory consumption along with it – EKF methods can only maintain relatively small maps. Modern EKF SLAM methods circumvent the issue by dividing a larger map into smaller local portions [23, 24], which in several aspects resemble graph-based methods. [3]

Nowadays, **particle filter** (PF) methods have surpassed EKF in popularity for online approaches. Because each of the particles is estimated with their own independent, uncorrelated distributions, the dimensionality of the system only grows linearly and larger maps can be formed. [3] Additionally, particle filters are capable of estimating multi-modal prob-

ability distributions, which allows the algorithm to track multiple possible robot locations at once. For example when there could be multiple similar locations in the map – and more evidence would be needed to confidently determine the agent’s real location – a PF maintains separate distributions for each location and only later favours the one with more evidence. Even though they are more scalable, PF methods are still mostly limited to on-line SLAM problems and – since their computational complexity grows with maps – the scale of the map is limited. [3] The key discovery to applying particle filters efficiently to the SLAM problem was the FastSLAM algorithm [25, 26], a Rao-Blackwellised PF which is the basis of most PF works on the field ever since.

**Graph-SLAM** methods draw their inspiration from graphical representations such as the one in Figure 2.1. Each state and map feature is represented by a node in a graph, while correspondences between them are described with edges between the nodes. Edges between each consecutive state are the transformations necessary to reach one state from the other according to odometry and the motion model, and features are connected to states where they were perceived with transformations according to the measurement model. Given the perceived features can be distinguished from one another, data association becomes as simple as connecting each feature node to each of the state nodes where it was sensed. Similarly, if a loop closure is detected, an edge is formed between the corresponding states. Even though the graph can grow really large with time, it is sparse: one node is only connected to few others. The number of constraints in it grows linearly with time and the number of nodes.

Solving the SLAM problem becomes a task of finding the path  $X_T$  and map  $m$  corresponding to the state of minimal energy. [3] That is, by finding the path and map corresponding to minimal errors according to the posterior distribution (2.4). The path  $X_T^*$  and map  $m^*$  corresponding to minimal energy can therefore be solved by finding the mode of the log-posterior distribution:

$$X_T^*, m^* = \operatorname{argmax}_{X_T, m} [\log(p(X_T, m|U_t, Z_t))]. \quad (2.6)$$

If noise is assumed Gaussian, the log-likelihood can be maximised by minimising the quadratic function [3, 27]

$$C + \sum_t (\Delta x^T R_t^{-1} \Delta x) + \sum_t (\Delta z^T Q_t^{-1} \Delta z), \quad (2.7)$$

where  $C$  is a constant and

$$\begin{aligned} \Delta x &= x_t - g(x_{t-1}, u_t), \\ \Delta z &= z_t - h(x_t, m). \end{aligned} \quad (2.8)$$

In other words, the task is to find the path and map which correspond to least errors, given the estimates of motion model  $g$  and measurement model  $h$ . The graph is sparse,

therefore the function can be solved with efficient quadratic optimisation methods. [3] The matrices  $R_t$  and  $Q_t$  contain covariances between different state errors  $\Delta x$  and measurement errors and  $\Delta z$ . If errors are assumed uncorrelated, the covariance matrices become diagonal and the problem is simplified further. Visual SLAM systems often apply a special case of nonlinear optimisation, called **bundle adjustment** (BA). [28] BA methods aim to jointly optimise both the estimated path and the 3D structure of the map by treating the correspondence between a camera pose and map features as bundles of light rays connecting them. The cost function in this case isn't necessarily quadratic, but many efficient methods for solving it have been invented. [29, 30, 31]

Graph-based methods scale really well to large environments [3, 16], although with larger graphs, global optimisation becomes impossible to perform in real time. However, many modern approaches – *e.g.* [16] and [32] – only optimise smaller local portions in the neighbourhood of the current node, which allows them to maintain a constant computation time and function in real time even with large maps. Even without global optimisation, the map is topologically correct, but if a metrically more accurate map is desired, the graph can be optimised globally offline later. In [17], on the other hand, global bundle adjustment is performed on the background separate from the rest of the SLAM algorithm, when loop closures are detected. This makes global optimisation possible to compute without affecting the performance of the rest of the algorithm but can impose severe delays in loop closure updates if the map grows too large.

### 2.1.3 Visual simultaneous localisation and mapping

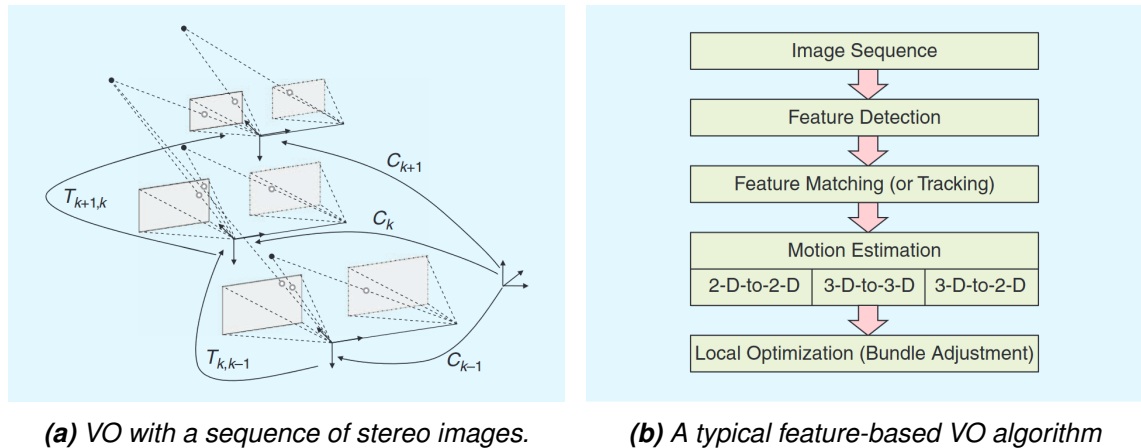
**Visual simultaneous localisation and mapping** (V-SLAM) is a relatively new research field compared to other SLAM research. Although visual data has been used in motion estimation since the 1980's [33], it has only recently been applied to SLAM in general due to high computational requirements. [7] Images contain a lot of useful information, but the algorithms used for extraction of said information demand a lot of computational resources, especially if the computation needs to be performed in real time. With advances in computational power and computer vision algorithms, real-time implementations of V-SLAM have been emerging since the early 2000's [34]. Cameras and range sensors are really affordable compared to laser scanners, therefore visual data provides a cheaper alternative to LiDAR, the trade-off being the higher computational requirements.

A digital image is a two-dimensional projection of the three-dimensional environment. However, for information in the image to be useful for localisation, it needs to be projected back to a three-dimensional representation. The 3D points corresponding to image pixels are also required for dense reconstruction of the environment. The third dimension missing from regular images is **depth**; the perpendicular distance to the respective 3D point corresponding to a pixel in the image plane. Stereo cameras – similar to human eyes –

acquire depth by finding the pixels corresponding to the same point in the environment in two parallel images and triangulating the distance. On the other hand, range sensors such as time-of-flight cameras can also be used to acquire depth directly. Camera systems that capture both colour and depth are often referred to as RGB-D (red, green, blue and depth) cameras. Sequences of monocular images can also be used similarly to stereo cameras to infer depth if their relative transformations are known accurately enough. However, depth estimation with monocular images is more prone to errors, and therefore not preferable in cases where depth can be acquired in other ways. On the other hand – because stereo and RGB-D imaging can only capture depth at quite a close range – monocular approaches are interesting in cases where distances are longer, for instance outdoors. [19, 35]

V-SLAM methods are separated to **filter-based** and **keyframe-based** approaches. Filter-based methods process every frame captured by the camera – *e.g.* with EKF or PF – while keyframe-based methods aim to only process the frames where significant changes have occurred. [36] Keyframe approaches are usually implemented with graph-based methods, in which case motion estimation can still be performed frame by frame to minimise drift and avoid errors caused by large viewpoint deviations while only keyframes are saved in the map graph. [16, 35] Keyframes can be selected with various criteria, for example based on changes in pose [37] or in visual appearance [35]. Keyframe-based approaches scale significantly better compared to filter-based methods – and therefore are the most common choice for full SLAM. [36]

**Visual odometry** (VO) is the process of estimating an agent's egomotion from consecutive images. [38] It can be thought of as the motion estimation part of the full SLAM algorithm. [19] The distinction between VO and V-SLAM is often drawn to loop closure: a full SLAM algorithm has to maintain a globally consistent map – which requires loop closure – while VO only considers the local consistency of the path. On the other hand, many VO works also apply keyframe selection and local bundle adjustment [39, 40, 41], thus they could also be considered as visual online SLAM in this regard. [6] Nevertheless, when applying VO as part of a full SLAM algorithm, keyframe selection and bundle adjustment are easier to associate as part of the actual SLAM algorithm, while motion estimation would then be separate from the rest of the system. [16] This separation allows the SLAM algorithm to be more flexible with the choice of odometry approach. Many works also utilise inertial measurement units (IMUs) as supplemental sensors to cameras. They provide a cheap remedy to situations where odometry based on purely visual data would fail, for example when there are occlusions or textureless surfaces. Inertial measurements can also be used to acquire scale in monocular V-SLAM. [42, 43] The sensor fusion of IMUs and cameras – commonly referred to as visual-inertial odometry (VIO) – can also improve the overall motion estimate. [44, 45] Labbe and Michaud even state that "Unless a long-range lidar is used, having odometry input from proprioceptive



**Figure 2.4.** An example of a visual odometry algorithm. [19] Local, relative, poses  $T$  are computed from feature matches, which are then concatenated and transformed to a global coordinate frame as absolute poses  $C$ .

sensors (e.g., IMU, wheelencoders) is mandatory for robust autonomous navigation." [16]

There are two distinct approaches to motion estimation. **Direct methods** [46, 47] aim to solve camera motion and the three-dimensional structure of the environment simultaneously by utilising every pixel in each image. On the contrary, **feature-based methods** [16, 35] only concentrate on visual features, areas in the image where more reliable correspondences are can possibly be acquired. Direct methods inherently result in a dense representation of the map, where each pixel corresponds to a point in space, while feature-based methods generate a sparse representation. However – since camera poses are known – feature-based methods can also generate a dense reconstruction separately if it is required by the application. Features like SIFT [14], SURF [15] and ORB [48] are designed to be robust against photometric and geometric changes, *i.e.* changes in lighting and viewpoint, while direct methods can only tolerate smaller changes in viewpoint and need to model a photometric map between the images. Feature-based approaches also advocate more efficient computation of camera motion and are more suited to bundle adjustment, as errors between feature matches are uncorrelated unlike in most motion models used with direct methods. [49] Direct methods are beneficial in cases where an accurate three-dimensional reconstruction of the environment is required, if relative motion between images is small, brightness changes are low and the scale of the map remains relatively small. [49, 50] On the other hand, feature-based methods should be applied if larger changes in viewpoints and lighting can occur, which is usually the case in practical SLAM applications. [35, 49]

An example of visual feature-based odometry with a stereo camera is presented in Figure 2.4. The same steps are usually found in similar approaches, but the methods with which they are implemented can vary. [19] The stereo camera is not a strict requirement: depth could be acquired by other means as well without affecting the overall algorithm

much. First, features are detected in each stereo image pair and triangulated to get their representative location in three-dimensional space. The features are then matched – or tracked – between viewpoints, after which the relative motion between views can be computed. Finally, bundle adjustment is performed to reduce drift. The steps up to motion estimation could be performed for each frame acquired from the camera, but to be able to perform local bundle adjustment, keyframes need to be detected. In the case of direct methods, instead of feature detection, matching and motion estimation, the task would be to directly estimate motion based on changes in appearance and geometry.

Loop closure can be applied in three fundamental ways: **image-to-image**, **image-to-map** or **map-to-map**. Image-to-image matching is performed by directly finding the closest match to the current image from earlier keyframes, while the latter two maintain a sparse map of visual features. Features in the two latter cases are matched either from the current image or from a local feature map consisting of the most recent features. Image-to-image methods are most suited to large-scale environments, since they scale really well and provide good robustness and recall even with a large search space. [51, 52] In this case, a keyframe-based approach is necessary since without it the search space grows too fast to be feasible for practical applications. [7] **Bag-of-Words (BoW)** [53] techniques are most commonly applied due to their high efficiency. [16, 35] BoW methods generate quantised words for each image based on visual features, from which a vocabulary is formed. Each image is represented by a signature: a collection of words. The task of finding a possible loop closure then becomes simply to find the closest match based on said signature, and determine if the signatures are similar enough. Once a match is found, a transformation between images can be computed with the features to complete the loop. A modern implementation of BoW loop closure is presented in Section 3.1.3.

Data association in V-SLAM can be considered from multiple standpoints. In feature-based approaches, the features between images need to be associated with their counterparts in other images. This is generally achieved by comparing descriptors based on the features' appearance. Wrong associations between feature descriptors are bound to happen, which will greatly reduce estimation accuracy if not handled correctly. [54] The random sample consensus (RANSAC) [55] is proven to be an effective method for removing these outliers. Additionally, features could be tracked between consecutive images, if the framerate is high enough. This will limit the search space for matching, since only a small portion of the image needs to be searched instead of comparing each descriptor in the first image with every descriptor of the second one. In data-association terms, descriptor matching has known correspondence, while tracking has unknown correspondence. Consequently, image-to-image matching in the context of loop closure also has known correspondence, while in the case of view-to-map or map-to-map matching the correspondence is most often unknown, since matches are usually achieved by minimising a distance function between groups of features. [56]

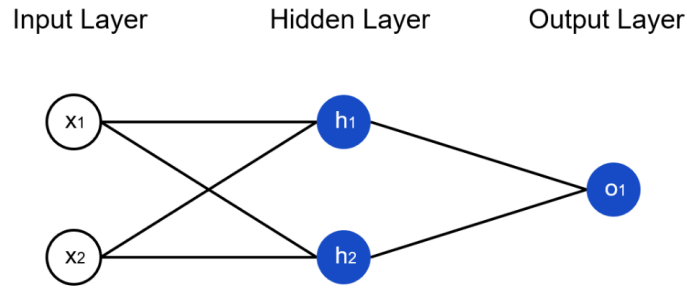


## 2.2 View-based scene segmentation

Scene segmentation methods can be divided into two categories: view-based and map-based. View-based approaches combine image segmentation methods and multi-view three-dimensional reconstruction. The goal of view-based methods is to segment each camera view separately and fuse the individual views to a three-dimensional representation, while map-based approaches try to segment the already generated 3D reconstruction – or in the case of large maps, a portion of it – at once. In theory, map-based approaches are more efficient, since each point in the scene has to only be processed once, while view-based methods will inevitably have some overlap between views. Additionally, map-based approaches will have more data available at once, from a larger area of the environment, which will provide more context if the model is able to capture it. For example, an object might be hard to classify from only one viewpoint, since it could be only partially visible or there might be no distinctive features on the side it is viewed from, while an accurate 3D representation of the object has no such limitations.

However, map-based approaches are more sensitive to noise in camera pose estimates. [57] In most visual SLAM use-cases, pose errors are large enough to make the view-based approaches the most appropriate choice. Sequential processing of views could also be performed online – given enough computational resources are available – and thus the gained semantic information could potentially be used to aid SLAM as well. [58, 59, 60] Because data is mainly acquired with RGB-D cameras and poses are estimated with V-SLAM, this thesis will focus more on image segmentation methods and view-based scene segmentation. Map-based approaches, utilising point cloud segmentation methods like [61] or [62], would be the obvious choice if data would be captured *e.g.* with a LiDAR, although they could also be a viable choice in the future for RGB-D cameras as well if visual mapping is found accurate enough.

Image segmentation has been researched for quite some time, in many different application fields. Early approaches were based on handcrafted features and methods like thresholding [63], clustering [64] and probabilistic graphs [65], but recently – with the ever-increasing amount of computational resources and data – research has focused more around **deep learning** (DL). [66] Most modern image segmentation architectures are based on **Convolutional Neural Networks** (CNNs) [67, 68]. Because image segmentation with CNNs is a relatively new research field, research on the subject advances quickly and applied methods deviate a lot from one another. Therefore, this section focuses more on general concepts and the current state of research. A brief introduction to three-dimensional reconstruction of scenes from individual views is also presented. Since this thesis is more focused on what happens before reconstruction, the introduction is quite general and related mathematical discussion is kept at the minimum for the sake of brevity. Nevertheless, these concepts help to understand the implementation and



**Figure 2.5.** The structure of a multi-layer perceptron. [69]

practical applications of methods addressed in this work.

The rest of this section is structured as follows. The theoretical concepts necessary to understand CNNs in the scope of this thesis are first introduced briefly in Section 2.2.1, after which the segmentation scheme chosen for the thesis – panoptic segmentation – is introduced and justified in 2.2.2. Third, methods for image segmentation in general as well as specifically for panoptic segmentation are overviewed in 2.2.3. Finally, methods for extending image segmentation to three-dimensional scenes are discussed in 2.2.4.

## 2.2.1 A brief introduction to convolutional neural networks

The most basic form of a deep neural network – the **multi-layer perceptron** (MLP) – is presented in Figure 2.5. There are an arbitrary number of input and output units in their respective layers depending on the application, and a number of hidden layers consisting of hidden units. The number of hidden layers determines the networks **depth**, while the number of units in each hidden layer determine their respective **width**. Each unit – a perceptron – in the network is a function that takes a number of scalar inputs and outputs a single scalar. Most common form of such a function is [70, p. 192]

$$f(x) = g(W^T x + b), \quad (2.9)$$

where  $x$  is a vector containing all inputs. The vector of weights  $W$  and bias  $b$  of the linear inner function are trainable parameters. The nonlinear **activation function**  $g$  is applied to introduce nonlinearity to the model. **Rectified linear units** (ReLUs) [70, p. 192]

$$ReLU(z) = \max\{0, z\} \quad (2.10)$$

and their variants are a common choice for activation function in the hidden layers, since they make the network easy and consistent to train.

The **softmax** activation [70, p. 195]

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_{j=0}^{n-1} \exp(z_j)} \quad (2.11)$$

represents a probability distribution of a discrete variable across all its possible values, which is why it is often applied in output layers. They can be used *e.g.* to represent the model's confidence on a certain input representing any of the output classes in a classification task. MLPs are regularly employed as classifiers in larger deep learning architectures, in which case they are often called **fully connected layers**.

Convolutional neural networks apply the convolution operation in place of the matrix multiplication performed in MLPs in at least some of their layers. In continuous space, convolution of a function  $x(t)$  with a weighting function  $w(t)$  is defined as [70, p. 327]

$$s(t) = (x * w)(t) = \int x(a)w(t - a)da. \quad (2.12)$$

However, CNNs generally operate in discrete space. The convolution over a discrete function is defined as [70, p. 328]

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t - a), \quad (2.13)$$

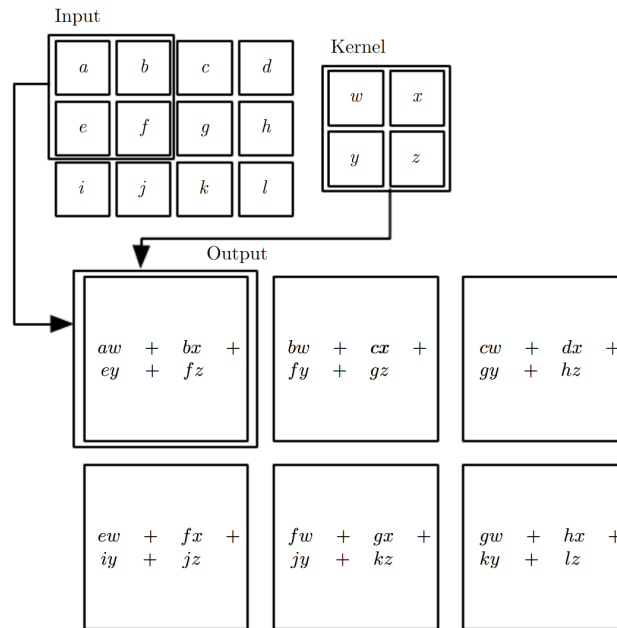
where  $x$  and  $w$  have discrete values at regular intervals of  $t$ . The weight  $w$  – also known as the **kernel** – is often assumed to be zero outside a finite set of values, which means that the infinite sum can be implemented as a sum over finite number of array elements. To perform convolution over the full input, the operation is then simply repeated for every step  $t$  and the outputs of each individual operation are concatenated to form the output, often called the **feature map** in CNN literature.

In many applications of CNNs, the input is a multi-dimensional array: a **tensor**. To capture the relationships between neighbouring elements in all dimensions, the convolution is performed over multiple axes simultaneously. For example, a greyscale image is a two-dimensional array. To extract features from the image, a convolution with a two-dimensional kernel  $K$  with fixed width  $w$  and height  $h$  for a pixel in image coordinates  $(i, j)$  in image  $I$  can be defined as [70, p. 328]

$$S(i, j) = (K * I)(i, j) = \sum_{m=0}^{w-1} \sum_{n=0}^{h-1} I(i - m, j - n)K(m, n) \quad (2.14)$$

Figure 2.6 presents an example of two-dimensional convolution over a small array. Notice that because the kernel is larger than a single array element, the resulting feature map is smaller than the input. This behaviour could be avoided by padding the outside of the input with enough zeroes, which can make the tensor dimensions easier to grasp in deeper networks where multiple convolutions follow each other.

Each convolutional layer of a CNN learns the weights for a multitude of kernels, with which the convolution is performed over all input dimensions. The individual feature maps

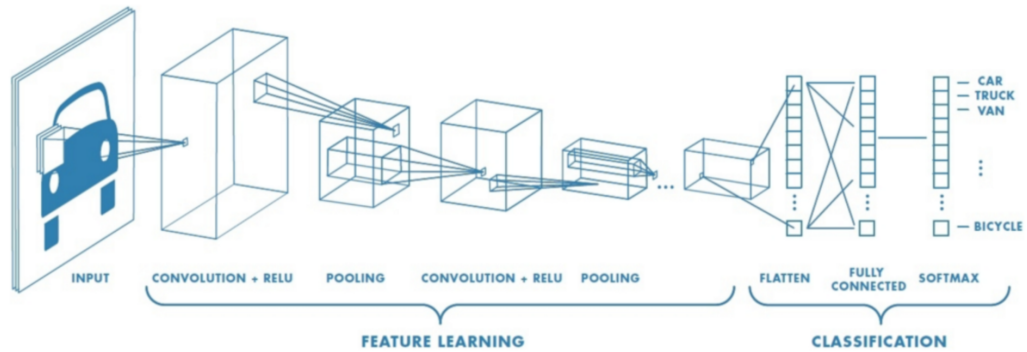


**Figure 2.6.** An example of two-dimensional convolution with a finite-sized kernel. [70, p. 330] Each element of the output array is a weighted sum of a window in the input array.

are then stacked atop each other, therefore the number of output channels is equal to the number of kernels in the layer. For example, a colour image has three channels representing the RGB values of each pixel. If we pass the image through a convolutional layer with *e.g.* 32 kernels, the output will have similar width and height as the image – assumed it is zero-padded to not reduce dimensions with the convolution – and 32 channels. Similarly, if the feature maps are passed through another layer with 64 kernels, the result will have 64 channels. Similar to MLPs, the activation functions are simply performed for each individual element of the feature map separately.

CNNs are especially useful when extracting information from structured, grid-like data like images. Instead of operating on individual input value, the convolution kernel captures **features** from a neighbourhood around each input location. This means convolutional layers can utilise context efficiently, *e.g.* to associate cars with roads or boats with water. They also offer sparser interactions than fully connected layers: there are fewer connections between units and therefore less trainable parameters, which make the networks easier to train and reduce computational requirements. [70, p. 329-335]

The size of a region in the input image corresponding to output features – *i.e.* the **receptive field** – grows larger towards the deeper layers of the network, which allows the network to extract features with varying degrees of detail and abstraction. The first layers of the network can extract smaller details and relationships like texture or edges, while the deeper layers can learn more abstract concepts like larger features and relationships between them. For example, in a CNN tasked with identifying cats and dogs from images, the early layers could try to separate different variants of fur, while the deeper



**Figure 2.7.** Visualisation of a simple CNN designed to classify different vehicles. [71]

layers might look for features like ears, tails or the general shape of the animal. While the receptive field does grow with each successive convolution, it is often useful to accelerate the growth with **pooling layers**. They are used to compress the feature maps to a smaller representation, in which case a finite-size kernel will capture a larger area of the input than without pooling. The most common pooling approaches are to simply take an average or a maximum value over small neighbourhoods of feature map elements and merge them to a new feature map. [70, p. 335-339] Pooling also renders the network more robust against small translations of the input, which means – for example – that an object’s predicted label is not as dependent on its location in the image.

Figure 2.7 presents a simple CNN of a classic form tasked with classifying vehicles from images. A section consisting of convolutional layers and pooling operations is first applied to extract features from the image. Then, the features are flattened to a vector and passed through a number of fully connected layers to capture the essential information related to the classification task, *e.g.* the existence of headlights or handlebars. Finally, the last fully connected layer has a number of outputs equal to possible object classes and a softmax activation function to acquire a confidence score respective to each class. The predicted class of the object is then simply selected to be the one with the highest confidence.

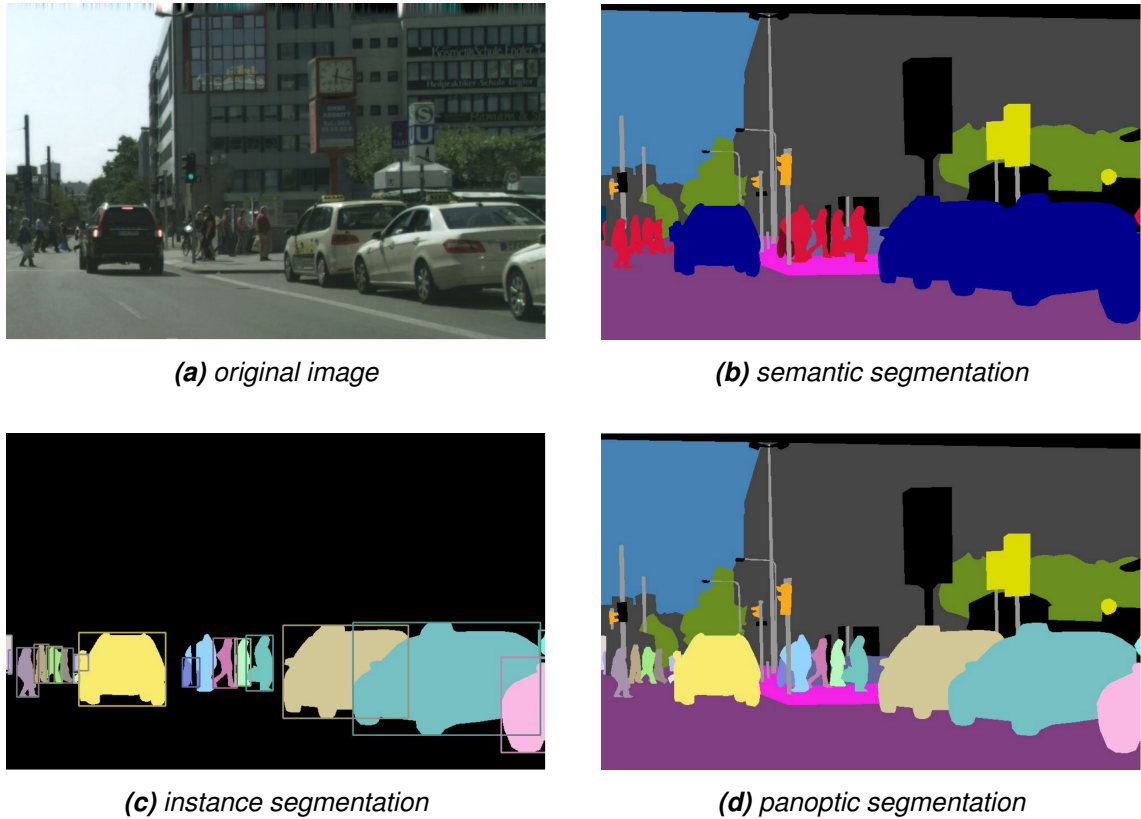
The CNNs introduced in this thesis are trained with **supervised learning** methods, most prominent of which in deep learning are the **stochastic gradient descent** (SGD) algorithm and its derivatives. [70, p. 149-150] The basic principle is to pass a **training sample** as an input through the network and compare the resulting output to a known **ground truth**. The outputs are compared to the ground truth with a **loss function**: a performance measure designed for the task at hand. The gradient of the loss function is then computed in relation to each weight in the network, after which all the weights of the network are then adjusted by a small step towards the opposite direction of their respective gradients in a process called **back-propagation**. The goal is for the network to eventually converge to a minimum with respect to the loss function, minimising the error between training outputs and the ground truth.

## 2.2.2 Panoptic segmentation

**Panoptic segmentation**, proposed in [72], combines the tasks of **semantic segmentation** and **instance segmentation**. The goal of semantic segmentation is to associate every pixel of an image to a semantic class, while instance segmentation methods aim to segment each individual object in the image, as well as determine its semantic class. Semantic segmentation lacks information on individual instances: with the semantic labels, one can discern where different types of objects are in the image, but cannot necessarily separate them from one another. For example, one could tell that there are cars in the image, but could not track each car individually. Instance segmentation takes care of this problem, but the semantic knowledge of unquantifiable, amorphous areas – like sky, road or pavement – in the image is lost. Therefore, by combining the two segmentation approaches, a more complete representation of the scene can be formed. While both tasks can be solved separately, the combined task helps to resolve inconsistencies between them. Estimating both in parallel with a unified model would also save resources, and could even improve generalisation in comparison to learning them individually. [73] Lately, research on semantic segmentation and instance segmentation has largely been isolated, while research on panoptic segmentation strives to bring the two tasks closer together again. [72] The intuition behind the three different segmentation formats is visualised in Figure 2.8.

In addition to introducing the new task, Kirillov *et. al.* also propose a simple and general output format, as well as a unified panoptic evaluation metric, coined **Panoptic Quality** (PQ). [72] Pre-existing performance metrics are not suitable for the panoptic segmentation task. In previous research on similar tasks, the performance on semantic- and instance segmentation has been evaluated separately using independent metrics for both. In the article, it is noted that this could make comparing different algorithms more difficult, while a unified metric would make algorithm development and comparison easier, thus further encouraging the study of the unified task.

The task format is proposed as follows. Given a set of  $n$  semantic class labels  $\mathcal{L} = [0, 1, \dots, n - 1]$ , each pixel  $i$  of an image is to be assigned a label  $l_i \in \mathcal{L}$  and an instance id  $z_i$ . Each pixel with the same  $z_i$  belong to the same object instance. Pixels not belonging to any of the  $n$  classes can be assigned to an additional void label. The set  $\mathcal{L}$  is divided to two subsets:  $\mathcal{L}^{St}$  and  $\mathcal{L}^{Th}$ , representing **'stuff'** and **'things'**. A class can only belong to one of the subsets. Stuff is defined as classes that can not be quantified, while things are classes that can be quantified. The distinction might vary between applications: buildings – when viewed from the street – can be thought of as stuff, while in the case of aerial images they could be assigned to a thing class. If no thing classes are specified, the task is identical to semantic segmentation. On the other hand, contrary to instance segmentation, overlaps between object segments are not permitted,



**Figure 2.8.** Intuition behind different image segmentation approaches. [72] In semantic segmentation, each pixel is labelled with a semantic class, while in instance segmentation each individual instance is segmented and labelled separately. Panoptic segmentation generalises them to a unified task.

and confidence scores for each prediction are not required. The exclusion of confidence scores makes the task symmetric between humans and machines, therefore the consistency of both can be evaluated similarly. However, it is noted in the article that confidence score could still be included in the algorithms, and might be beneficial in providing more information to downstream systems. [72]

Three desirable qualities are set for the PQ metric. First, stuff and things should be of uniform importance, and all aspects of the task should be evaluated. Second, the metric should be easy to interpret, and third, it should be easy to define and implement to avoid confusion and improve transparency. A **prediction** is compared against a **ground truth** comparison. Image segments are matched between the ground truth and the prediction. The segments can match only if their intersection over union (IoU) is strictly greater than 0.5. This – in addition to the requirement of no overlaps in the segmentation – results in a unique match: only one predicted segment can match each segment in the ground truth.

The metric is computed independently for each class and averaged over the set of classes, which makes it insensitive to **class imbalance**: some classes might be represented more than others. Segment matches for each class belong to one of three categories: **true**

**positives** ( $TP$ ), **false positives** ( $FP$ ) or **false negatives** ( $FN$ ). The metric is computed as

$$PQ = \frac{\sum_{(p,q) \in TP} IoU(p, q)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}. \quad (2.15)$$

A segment's area therefore does not affect the evaluation. The PQ metric can also be presented as

$$PQ = \frac{\sum_{(p,q) \in TP} IoU(p, q)}{|TP|} \times \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}, \quad (2.16)$$

which divides the metric to two parts: the left-hand side of the multiplication is called **segmentation quality** (SQ), which is simply the mean IoU of correct matches, while the right-hand side – called **recognition quality** (RQ) by the authors – is equal to the widely used  $F_1$  metric [74]. However, Yang *et. al.* [75] criticise the PQ metric by arguing that since the area of segments do not affect the PQ score, it might overemphasise smaller objects. They also propose the **Parsing Covering** (PC) metric as an alternative for cases like autonomous driving, where objects closer to the camera – thus appearing larger in the image – are generally more important than ones further away.

Kirillov *et. al.* [72] also provide suggestions on how to handle void labels and group annotations. Since it is hard to differentiate between out-of-class pixels and ambiguous or unknown areas of the image, void pixels are excluded from the evaluation: they should not affect the IoU computation and unmatched void areas are not counted as false positives. In some datasets, adjacent objects are annotated as groups rather than separate instances. They are not used in matching and the unmatched predicted segments where group labels of a single class have over 0.5 IoU with the segment are omitted from false positives.

Panoptic segmentation is a form of Multi-Task Learning (MTL) [73]. There has been research on unified semantic- and instance segmentation before the term panoptic segmentation was introduced, but since the publication of [72], it has gained a lot more popularity and has already been adopted as a common term describing the task. However, some MTL approaches on dense image prediction also include depth estimation [76], which could also be beneficial to segmentation tasks [77], but has not been combined to a similar unified task as panoptic segmentation. As this thesis considers cases where depth information is already available, depth estimation is not necessarily required, although depth completion [78] could be beneficial for accurate 3D reconstruction. Additionally, depth could provide geometrical context to *e.g.* aid in situations where segments are hard to interpret from colour images alone [79, 80] or to separate the image to areas representing different scales, which has been proven to simplify the segmentation of scenes with large variance in depth. [81]

Since the proposition of panoptic segmentation in 2018, there has been a lot of research

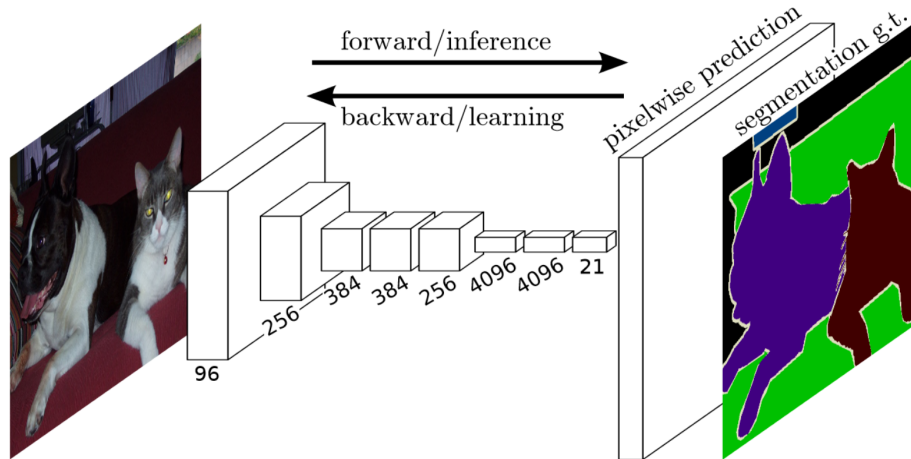


in the area. Many datasets have been updated with panoptic annotations, most notable of which are the COCO Panoptic Segmentation Challenge [82] and Cityscapes Panoptic Benchmark [83]. Since June 2018, there have been 27 unique submissions in the COCO challenge, while the Cityscapes benchmark – launched in June 2019 – has accumulated 20 entries in little over a year. For comparison, the instance segmentation task of COCO Detection Challenge 2020 [84] – launched simultaneously with the panoptic challenge – currently has 41 entries, while Cityscapes instance- and semantic segmentation tasks have 64 and 249 entries since 2016, both having more than 20 entries since the launch of the panoptic benchmark. The individual segmentation tasks seem to still be more popular than panoptic segmentation – perhaps due to familiarity – but panoptic segmentation seems to have gained a lot of popularity in just a few years and most likely will do so in the future as well.

### 2.2.3 Image segmentation methods

One of the first deep-learning approaches to semantic image segmentation was the **Fully Convolutional Network** (FCN) proposed by Long *et. al.* [85]. A simple illustration of FCN is presented in Figure 2.9. The model – based on earlier CNN architectures such as AlexNet [86], VGG16 [87] and GoogLeNet [88] – consists solely of convolutional layers, enabling segmentation of inputs of arbitrary sizes. The FCN model also combines feature maps from earlier layers of the network containing more detailed features with the deeper final layers consisting of coarser features with more semantic meaning. This contributed to a better fusion of appearance and semantic information. The model has influenced many other works on the field since: most feature extractors – also known as backbones – of segmentation models are fully convolutional networks. [66] Although FCN fuses features of multiple scales together, it still struggles to capture large-scale context in an image. This is clarified by an example in [89]: in Figure 2.10, a boat is misclassified as a car, although a boat is much more likely to be found in a river.

Another fundamental technology behind most image segmentation models is the convolutional encoder-decoder architecture. [66] An **encoder** is a model that compresses an input to a feature representation capturing the input’s semantic information, while a **decoder** is used to predict the output from said features. Noh *et. al.* applied a VGG16 model without the final classifier layer to encode input images and a symmetric decoder to predict a segmentation mask. [90] Deconvolution – also known as transpose convolution or up-convolution – and unpooling layers are utilised in a decoder network to upsample the features in a mirroring fashion to downsampling in the encoder. Badrinarayanan *et. al.* [91] built on this idea and introduced SegNet, a fully convolutional network with a similar structure. They also proposed a novel way of upsampling the lower resolution features by reusing indices of the pooling layers of the encoder in non-linear upsampling. This



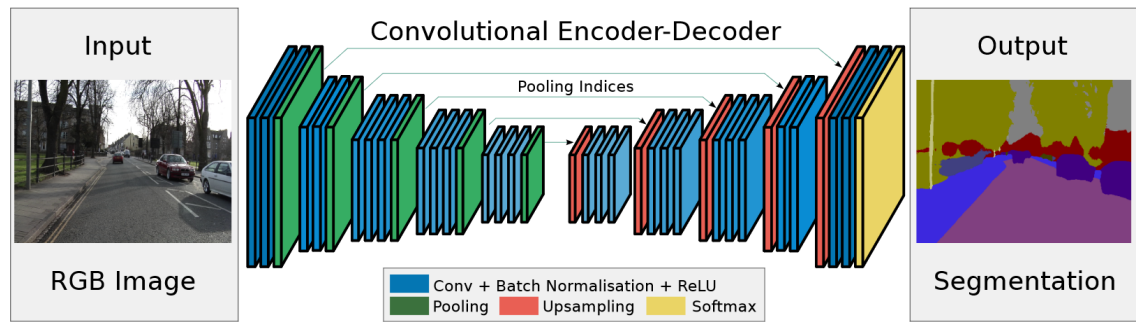
**Figure 2.9.** Topology of a Fully Convolutional Network for semantic segmentation. [85]



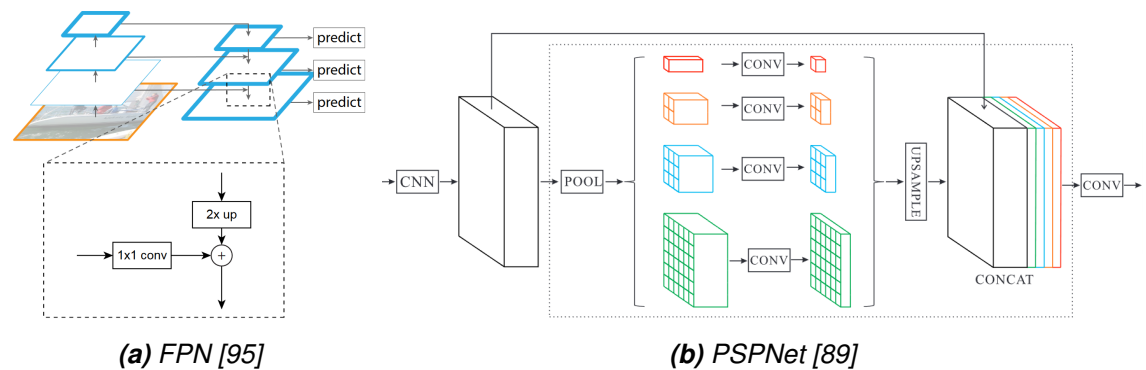
**Figure 2.10.** An example in [89]. Left: original image, middle: ground-truth, right: FCN prediction. A boat (labeled as light green) is misclassified as a car (blue) due to missing the global context: a boat is much more likely to be found in a river.

way, upsampling does not need to be learned. The upsampled sparse features are then passed through convolutional layers to produce dense feature maps. Because of the fully convolutional structure and novel upsampling, SegNet had significantly fewer parameters than other competing architectures at the time. The model is depicted in Figure 2.11. Encoder-decoder models have also been popular in medical image segmentation. There have been multiple models developed initially for this field – like U-Net [92] and V-Net [93] – that have later made their way to other computer vision research as well.

Since the receptive field of FCN's and other early encoder-decoders' output is at a single scale, the models need to make a trade-off between the ability to capture detail and context. Details are easier to capture with a smaller receptive field, but larger-scale context is lost and vice versa. Image pyramids – utilised in computer vision long before deep learning had gained popularity – are a way to avoid loss of detail and retain context in multiple scales. [94] More recently, they have also been an inspiration for many deep learning methods addressing the issue. Feature Pyramid Network (FPN) proposed by Lin *et. al.* [95] was first introduced as a backbone for object detection, but has since gained a lot of popularity in instance segmentation as well. It fuses features from the intermediate layers of the decoder to preserve information from multiple scales. FPN outputs multiple feature maps representing different scales, which can each be used independently for prediction. At roughly the same time, Zhao *et. al.* [89] introduced the Pyramid Scene Parsing



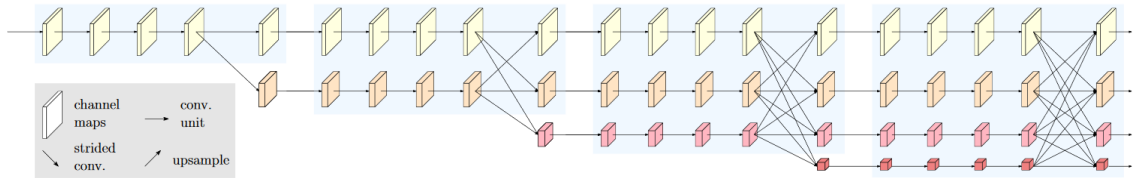
**Figure 2.11.** SegNet: a fully convolutional encoder-decoder network. [91]



**Figure 2.12.** Visualisation of pyramid approaches to capture multi-scale context in image segmentation. Feature Pyramid Network acquires multi-scale features from different phases of the backbone, while Pyramid Scene Parsing Network pools the backbone's final representation in multiple scales.

Network (PSPNet) for semantic segmentation. It acquires features of multiple scales by pooling sub-regions of different sizes from a backbone's output and passing them through additional convolutional layers. The resulting features are upsampled and concatenated along with the original backbone output, after which the combined feature map is used to predict the semantic segmentation. Both pyramid approaches are visualised in Figure 2.12. The High-Resolution Network (HRNet) [96] has a slightly different approach to the issue. It maintains features of multiple scales with parallel convolution streams of different resolutions that exchange information between them repeatedly. The backbone of HRNet is depicted in Figure 2.13. Like FPN, it outputs multiple feature maps, which are used to predict the semantic segmentation in a way similar to PSPNet.

The DeepLab model family [97, 98, 99, 100] is another popular group of segmentation architectures, more focused on semantic segmentation. They utilise dilated – also known as atrous – convolution to reduce the number of required parameters for upsampling in relation to deconvolution. Atrous Spatial Pyramid Pooling (ASPP), a method of capturing features in multiple scales more efficient than earlier approaches, was introduced with DeepLabv2. [98] However ASPP – while substantially increasing the model's performance – also requires a large number of parameters. More recently proposed methods based on ASPP – Dense Prediction Cells (DPC) [101] and Efficient Atrous Spatial Pool-



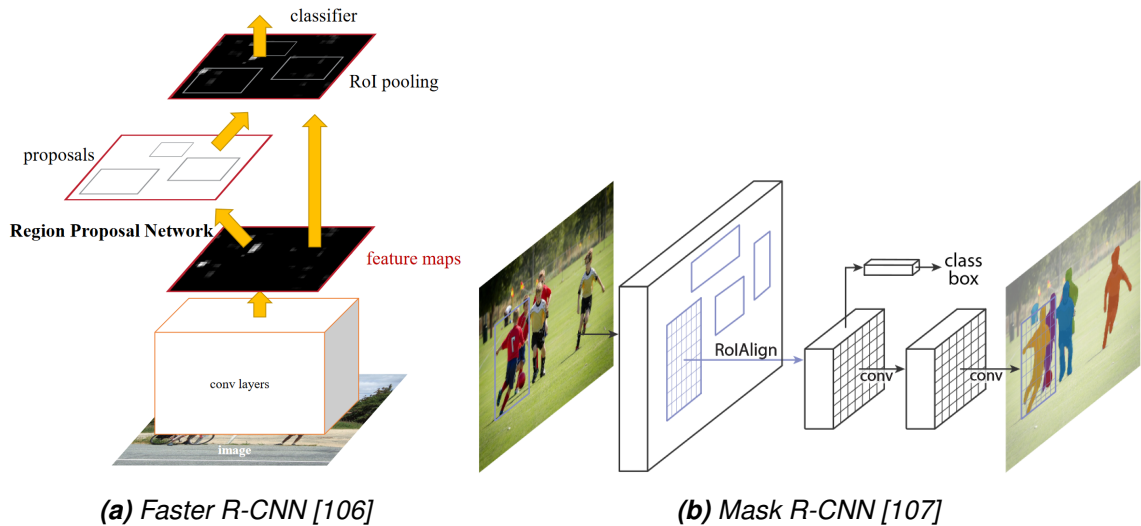
**Figure 2.13.** Parallel convolution streams of High-Resolution Network. [96]

ing (eASPP) [102] – increase performance even further, while simultaneously reducing the number of parameters needed to one tenth of the original. DeepLab models also utilise a fully connected Conditional Random Field (CRF) [103] – a form of probabilistic graphical model – on top of the convolutional network, which proved to improve the segmentation at objects’ boundaries. CRF:s and other similar probabilistic graphical models are known to capture large-scale context in the images as well. [66]

Regional Convolutional Neural Network (R-CNN) [104] and its extensions Fast R-CNN [105] and Faster R-CNN [106] have been really popular in object detection tasks. Mask R-CNN [107] further extends Faster R-CNN to instance segmentation by adding an instance segmentation head parallel to bounding box regression and object classification. The key paradigm that allowed R-CNN to surpass other object detection methods at the time was a selective search method for Regions of Interest (RoI) in the input image. The method for generating the rectangular proposals in R-CNN is quite slow, and thus RoI pooling was introduced with Fast R-CNN. While RoI pooling made end-to-end training of the model possible and significantly sped up training and inference speed, region proposals still had to be acquired separately and fed to the network as an input. As a remedy, the Region Proposal Network (RPN) – a network that generates the proposals for RoI Pooling from the backbone’s output – was introduced with Faster R-CNN. Information flow in Faster R-CNN is visualised in Figure 2.14a.

RoI pooling quantises features from the backbone network, which results in small misalignments in the proposals. This is easily corrected with the following bounding box regression but leads to noticeable errors in instance segmentation. Therefore, a new iteration of the method – RoI Align – that does not quantise but rather interpolates the features was introduced with Mask R-CNN. The structure of Mask R-CNN is depicted in Figure 2.14b. The backbone network is typically an FPN with a ResNet [108] encoder. Region proposals are generated by RPN from the features generated by the backbone, which are then fed to the RoI align module. The RoI:s are then fed to three output branches: the object classifier, the bounding box regressor and the instance mask head. Many works on instance segmentation, like Path Aggregation Network (PANet) [109], MaskLab [110] and TensorMask [111], have since been based on Mask R-CNN.

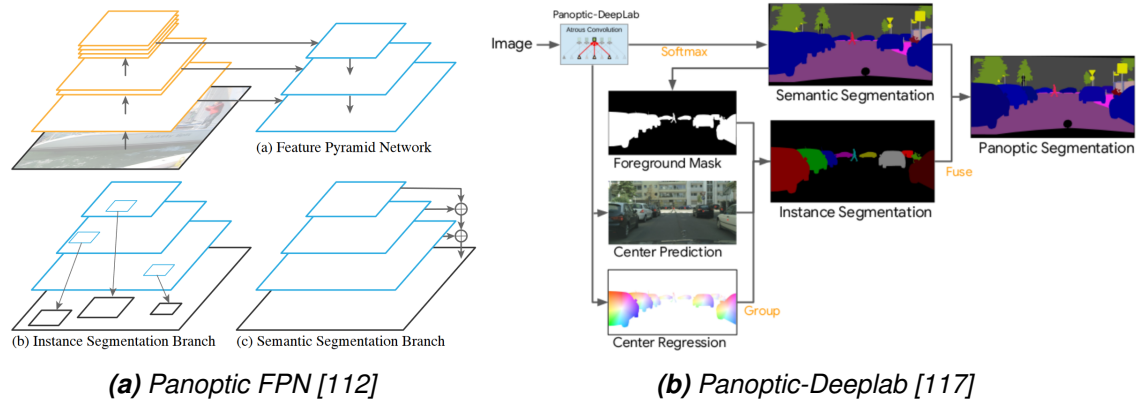
When Kirillov *et al.* introduced panoptic segmentation, they applied a combination of PSPNet and Mask R-CNN as a baseline for the task. [72] Semantic and instance seg-



**Figure 2.14.** Topology of Faster R-CNN and its extension to instance segmentation, Mask R-CNN.

mentations are predicted with the separate models, and then fused together by pasting instance segmentations atop semantic masks in a naive manner. However, they note in the article that they expect future approaches to address the task with more efficient end-to-end models capable of solving both segmentation tasks simultaneously, as well as more sophisticated fusion mechanisms. They also simultaneously published Panoptic Feature Pyramid Network [112], visualised in Figure 2.15a, which added a lightweight semantic segmentation head parallel to the output of Mask-RCNN. Similar to Panoptic FPN, most panoptic segmentation models, like UPSNet [113] and Seamless [114], are proposal based – also known as top-down – architectures. [115] TASCNet [116] is another interesting top-down model based on the FPN architecture that learns both the fusion mechanism and segmentation simultaneously.

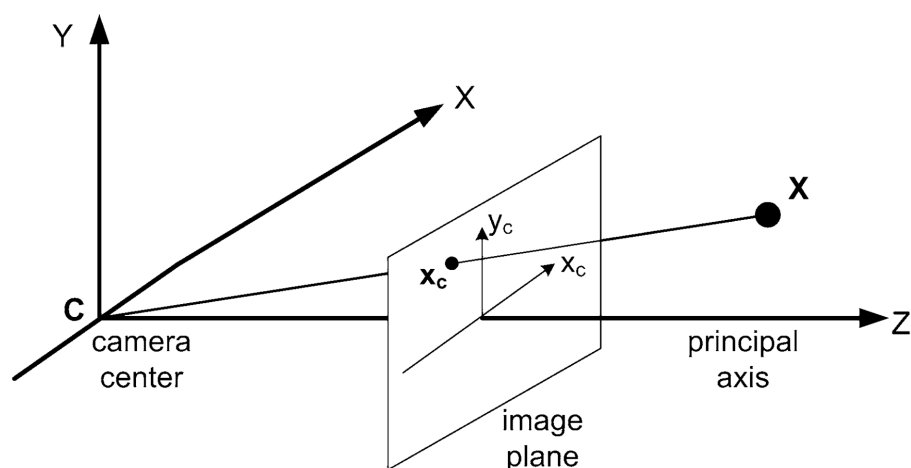
While proposal-based architectures are most common in panoptic segmentation, proposal-free – or bottom-up – models have been studied as well. The computational cost of Mask-RCNN and models based on it increase with the number of generated proposals, so by foregoing the proposals altogether the model should be more efficient. On the other hand, naively fusing separate segmentation outputs favours instances over semantic segmentation. The first of these approaches was DeeperLab [75]. It combines a semantic segmentation model inspired by DeepLabV3+ [100] with a class agnostic keypoint-based instance segmentation model. Stuff labels are determined directly from the semantic segmentation predictions, while thing labels are determined by a majority vote from the semantic segmentation results in the areas described by instance segmentation. Since both stuff and thing labels originate from the same segmentation map and predicted instances have no overlap, there is no bias towards either. Cheng *et. al.* built on this idea with Panoptic-Deeplab [117]. They added separate ASPP decoders for semantic segmentation and the class agnostic instance segmentation after a shared encoder, the outputs of



**Figure 2.15.** A proposal-based panoptic segmentation model, *Panoptic FPN*, and a proposal-free model, *Panoptic-Deeplab*. *Panoptic FPN* segments object instances from region proposals generated by the RoI align module, while *Panoptic-Deeplab* uses class-agnostic segmentations to capture instance labels from the semantic mask.

which are fused similarly to the Deeplab approach. Information flow in the model is visualised in Figure 2.15b. While the two methods do not need proposals to determine instances, the semantic and instance segmentations are largely learned separately, increasing computational overhead. Additionally, the inference of instance segmentations from the detected keypoints requires quite a lot of computation as well. To address these issues, Gao *et. al* proposed the Single-Shot Instance Segmentation With Affinity Pyramid (SSAP) [118] model, which jointly learns semantic segmentation and a pixel-pair affinity pyramid describing the probability of neighbouring pixels belonging to the same object instance. Instances are then inferred from the affinity pyramid with a computationally more efficient cascaded graph partition module. For panoptic fusion, a similar voting scheme is applied as in the previous bottom-up approaches.

One of the newest works in the field, EfficientPS [115], has extended the Panoptic FPN by introducing a more efficient two-way FPN backbone, a novel semantic segmentation head and a panoptic fusion module that integrates output logits of both segmentation heads, outperforming all previous approaches. A top-down approach was chosen due to Mask-RCNN:s robustness against large scale variation of instances. The two-way FPN backbone utilises parallel top-down and bottom-up feature pyramids atop an EfficientNet [119] encoder, which provides better segmentation with fewer parameters than earlier approaches. The novel semantic segmentation head utilises Dense Prediction Cells [101] to capture long-range context and Large Scale Feature Extractor (LSFE) modules to capture features of larger scale efficiently. The misalignments between DPC:s and LSFE:s are corrected with Mismatch Correction (MC) modules. Extensive ablation studies in the article show that the proposed modules are indeed more efficient than their other commonly used counterparts. Since EfficientPS has been applied in this thesis, a more detailed description is provided in Section 3.2.

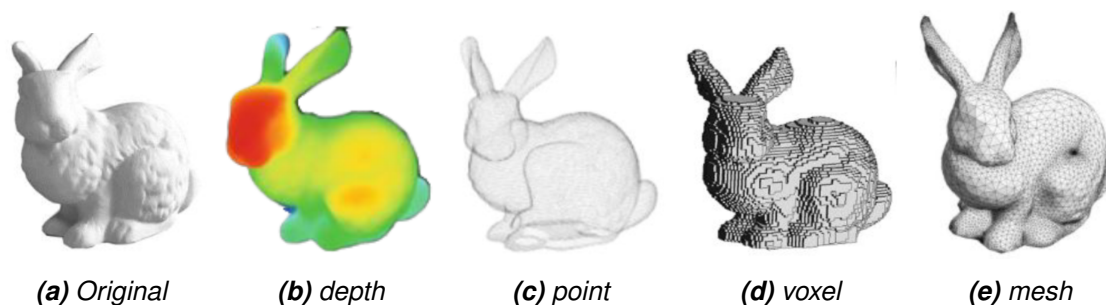


**Figure 2.16.** A visualisation of the pinhole camera model. [120, p. 42]

## 2.2.4 Extending image segmentation to three-dimensional scenes

Images are two-dimensional projections of the three-dimensional environment. The image only contains information on objects' locations in the projection on a 2D plane, which alone is not enough to project them back to a 3D representation of the environment. For example – in the case of Figure 2.16 – if we imagine a light ray passing from a point in space through the camera centre, the point is projected to the image in the intersection of the image plane and the light ray. This is the **perspective projection** of the 3D point. Correspondingly, we can **back-project** a point in the image as an infinite ray extending to the environment. The location of the point along the ray cannot be determined from the two-dimensional image alone, but if the point's distance from the camera centre or the image plane is known – acquired *e.g.* with a stereo camera or a range sensor – its 3D location relative to the camera centre can be computed with simple geometry.

Figure 2.16 presents a simple way to describe a camera: the pinhole camera model. An absolute – *i.e.* correct in both dimensions and scale – three-dimensional reconstruction from images requires – corresponding to each image – both the **intrinsic** and **extrinsic** parameters of the camera, as well as an aligned depth map or range image. [120, p. 72] Intrinsic parameters are the camera's internal parameters related to the task: the focal length, *i.e.* the distance from the camera centre to the image plane along the principal axis; the principal point, where the principal axis and the image plane intersect; pixel width and height; and image skew. Most cameras also have **lens distortion**, which is not taken into account by the pinhole model. The distortion therefore needs to be corrected for accurate reconstruction with the model. [120, p. 47] The intrinsic parameters and distortion can be automatically estimated with a **calibration** procedure, by capturing pictures of a calibration target with known dimensions from multiple angles and comparing them to the true shapes. Extrinsic parameters are the translation and rotation of the camera in world coordinates, acquired *e.g.* with a visual SLAM algorithm.



**Figure 2.17.** An example of common 3D representations on the Stanford Bunny. [121]

A **point cloud** is a collection of 3D points, which in the case of RGB-D imaging are back-projections of pixels in a digital image. If the Six-Degrees-of-Freedom (6-DoF) pose of the camera is known, these points can also be transformed to the map's coordinate system, thus the clouds respective to each image can be fused together to represent the whole environment. A point cloud in itself is not suitable for many applications outside simple visualisation, but it can be refined further to generate more useful and efficient representations [120, p. 139-182]. Since there are always noise and distortion that are not explained by the camera model, the point cloud will not be an exact reconstruction. Moreover, overlapping views result in redundant points, and errors in the 6-DoF poses might result in objects warping and appearing in multiple locations. Therefore, the real environment can only be approximated with the data.

Generally, representations generated from raw point clouds are divided to **surfaces** and **solids**. [120, p. 139-182] Most common way to represent surfaces is a **polygonal mesh**, which consists of vertices connected by edges. The edges form triangular faces. Each face is two-dimensional, thus a 2D image could be projected to the surface as a **texture**. For example, the images from which the point cloud was generated could be projected to the surface to colour the reconstruction according to the real environment. The other way to add colour to the mesh is to associate each vertex with a colour and interpolate the texture of the face according to the vertex colours. Solids, on the other hand, are a volumetric representation of the environment. The most common solid representation is a **voxel grid**. Voxels – "volumetric pixels" – are the 3D analogy of pixels. An area of 3D space can be divided into a binary cubical grid, where ones and zeroes represent occupied and unoccupied space. Alternatively, each voxel could be represented by a scalar *e.g.* to represent different densities. Each occupied voxel could also be associated with a colour. Examples of different 3D representations are presented in Figure 2.17.

Polygonal meshes are the most common 3D representation in computer graphics. They are efficient to store compared to point clouds and voxel grids, and they are efficient to display on a computer screen. Mesh surfaces are also visually pleasing. With high enough resolution the surface appears smooth, and textures can be applied to achieve a realistic colouration. In addition to visualisation, meshes are often used, *e.g.*, for navigation



and obstacle avoidance by mobile robots [122], or interaction with the scene, for example with augmented reality (AR) applications [123]. Solid representations, however, are better suited to some applications. They can be used to efficiently approximate the volumes of objects, or in efficient three-dimensional obstacle avoidance [124, 125]. Voxels can be visualised either directly – like an MRI scan, for example – or by first transforming them into a surface representation. Solids are less efficient to store than surfaces though – since unoccupied space is represented as well – therefore smaller portions of the environment can be addressed at once at a similar resolution compared to them. [120, p. 151]

Semantic segmentation masks are fairly straightforward to apply to the 3D representation: since each pixel of each input image has been labelled, each point has its own label as well. These could then be applied to the vertices of the generated mesh or voxels by *e.g.* a majority vote over a number of nearby points. The labels can then be visualised by associating each vertex or voxel with a colour. However, in the case of instance segmentation, the task is not as simple. Because instance ids are only valid in the context of a single image, additional methods need to be applied to track them over multiple views. Video segmentation methods like [126, 127] could be applied to track the labels in 2D, provided the views are captured in a sequence with a fast enough framerate. Another interesting approach was presented in [128], where the video segmentation model was taught to track the labels by itself. However, 2D tracking methods cannot necessarily re-identify instances if they appear in the video again later, therefore some sort of three-dimensional tracking method would be required regardless. For example, in [123] the labels of an image are compared to a 2D projection of labels in the 3D representation corresponding to the same view. Intersection over union is used to determine whether an instance corresponds to one already in the partial reconstruction. The confidence scores of each 2D instance segmentation corresponding to a 3D instance are integrated in a Bayesian manner to determine the instance class. A ray-casting algorithm similar to [129] is used to incrementally update a truncated signed distance field (TSDF) [130] volume, from which a polygonal mesh is generated. Once the reconstruction is complete, a fully connected CRF is employed to regularise the segmentation.

### 3 APPLIED METHODS

The essential methods for metric-semantic reconstruction applied in this work are described in this chapter. A SLAM method is utilised to render the system independent of any external infrastructure or existing maps. On the other hand, since paths are acquired with visual SLAM, a view-based segmentation method is the safest choice because of its robustness to pose errors. Moreover, panoptic segmentation has been chosen as the segmentation scheme since as it allows for the widest range of applications: both semantic and instance segmentation can be captured from it separately, or fused together as the full panoptic representation.

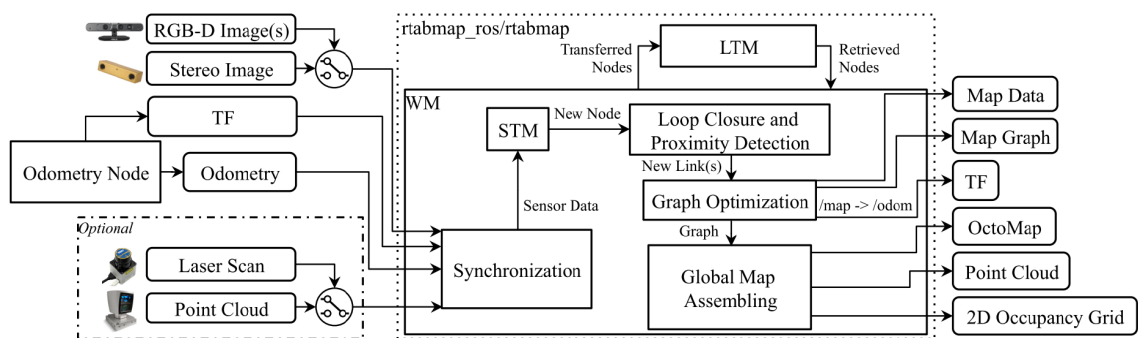
As determined earlier, a graph-based SLAM approach with image-to-image loop closure is the most reasonable choice when scalability and global consistency are required. The **Real-time Appearance-based Mapping** (RTAB-Map) library [16] has been chosen as the graph-based V-SLAM approach in this work due to its efficiency, performance and large selection of options. The library is introduced in detail in Section 3.1.

**EfficientPS** [115] is chosen as the panoptic segmentation method for this work because of its exceptional performance on public benchmarks and relatively low computational requirements. The authors of the original article have published a re-implementation of the source code used to acquire the reported results, which has been re-purposed in this work for view-based segmentation of indoor scenes. Although some modifications had to be made to the original code for it to function in this work's purposes, the underlying neural network still remains the same. The EfficientPS panoptic segmentation model is described in Section 3.2.

While the most essential methods are described here, their integration into the full reconstruction system is introduced in the next chapter, where their performance will also be evaluated. Apart from assessments based on the background provided earlier, their suitability for the task is evaluated and justified in the chapters following this one.

### 3.1 RTAB-Map: Real-Time Appearance-Based Mapping

RTAB-Map first began as a loop closure algorithm in 2013 [131], and has since then grown to be one of most complete open-source graph-based visual SLAM approaches to date. [16] It has been developed to support various different use-cases and hardware configurations, especially real-time computation and scalability in mind. It offers both robust and accurate real-time localisation, as well as offline tools for generating detailed maps from the SLAM graph. The maps can then also be used for localisation afterwards with the same framework. Because of the open-source implementation – both in ROS and as a standalone C++ library – and extensive options and tools, RTAB-Map is the ideal choice for early-stage research and prototyping. There are some publicly available V-SLAM implementations that seem to be more accurate based on results reported on benchmark datasets, but none provide the flexibility that RTAB-Map has.



**Figure 3.1.** Block diagram of RTAB-Map [16].

The core algorithm has five fundamental parts: odometry, memory management, loop closure, graph optimisation and map assembly. Each of these provides multiple options to cater for different setups and purposes. Block diagram of the overall system is depicted in Figure 3.1. Messages from different sensors and the odometry node are first synchronised and passed to **Short Term Memory** (STM), which generates nodes for the map graph. Then, loop closures are detected. Possible new edges are added to the map graph, and the graph is optimised. From the graph, a dense global map can be assembled, either online or offline. Only nearby nodes are contained in **Working Memory** (WM). If localisation becomes too slow, *i.e.* when map update time or occupied memory exceeds a given threshold, less important nodes are transferred to **Long Term Memory** (LTM), and are no longer available to the nodes inside WM. When a loop closure is detected, nodes near the recognised location are transferred back to WM from LTM.

#### 3.1.1 Odometry options

Basic visual and LiDAR odometry approaches are provided with the library in case no other options are available, but since the odometry node is independent of the rest of the

algorithm, any kind of external odometry can be applied as well. In [16], the authors of RTAB-Map find that while the provided odometry is already quite good, some other VO approaches can be more accurate. Notably, the VO implementation of ORB-SLAM2 [17] provides the best accuracy in their studies.

The methods provided with the library are standard approaches inspired by [19]. A keyframe-based approach is applied with local bundle adjustment. Visual Odometry can estimate motion either in relation to last keyframe, **Frame-to-Frame** (F2F), or in relation to a local map of features, **Frame-to-Map** (F2M). The size of the local feature map is limited to allow constant-time computation. Similarly, LiDAR odometry can also be performed both in **Scan-to-Scan** (S2S) and **Scan-to-Map** (S2M) manner. Detected visual features can be any of the ones supported by OpenCV [132], *e.g.* SIFT [14], SURF [15] or ORB [48]. The features' positions in the next frame are predicted with a motion model to limit search windows when matching them with the new frame. Motion estimation is performed with the Perspective-n-Point (PnP) RANSAC implementation in OpenCV. [132]

### 3.1.2 Visual vocabulary and node generation

Short-Term Memory is responsible for generating the nodes of the map graph. Nodes are generated on fixed time intervals but can be fused later if they are found similar enough. Following information is included in each node: [133]

- **ID**: a unique index for the node.
- **Weight**: an indicator of the importance of the node. Used to determine the order in which nodes are saved to LTM.
- **Signature**: a unique signature used to determine similarity between images.
- **Sensor data**: odometry pose, RGB image, depth image and laser scan.

An image's signature is generated with the Bag-of-Words (BoW) approach. Each time an image is acquired by STM, it extracts visual features from it. These can be any type of feature supported by OpenCV [132], *e.g.* SIFT [14], SURF [15] or ORB [48]. They are then quantised to visual words and appended to a vocabulary to limit the size of the search space. The features are compared with the Nearest neighbour Distance Ratio (NNDR), a commonly used distance metric in feature matching. A feature is considered to match a visual word if their distance is less than the distance to the second-nearest word multiplied by a given parameter  $T_{NNDR}$ , *i.e.* if its NNDR is low enough. If a feature does not match any visual word, it is appended to the vocabulary as a new one. Otherwise, it is associated with one already found in the vocabulary. The signature is then simply the collection of words associated with the image. Because of a large search space and high-dimensional words, the vocabulary is a randomised forest of k-dimensional trees, implemented in FLANN [134].

The similarity between signatures of consecutive nodes STM is defined as [131]

$$\mathcal{S}(z_t, z_{t-1}) = \begin{cases} N_{pair}/N_{z_t}, & \text{if } N_{z_t} \geq N_{z_{t-1}}, \\ N_{pair}/N_{z_{t-1}}, & \text{if } N_{z_t} < N_{z_{t-1}}, \end{cases} \quad (3.1)$$

where  $N_{pair}$  is the number of matching words between the pair,  $z_t$  the new node and  $z_{t-1}$  the previous one, while  $N_{z_t}$  and  $N_{z_{t-1}}$  are the number of words in their respective signatures. If  $\mathcal{S}(z_t, z_{t-1})$  is higher than a given threshold, then  $z_t$  is merged to  $z_{t-1}$ , otherwise the weight of  $z_t$  is initialised to zero and a bidirectional neighbour link is formed between the two. If the two nodes are merged, the signature of  $z_{t-1}$  is retained, while the newly added words in  $z_t$  are removed from the vocabulary. Neighbour- and loop closure links to  $z_{t-1}$  are redirected to the merged node and  $z_{t-1}$  is deleted from STM. The weight of the merged node is set as the sum of the two plus one, thus each node gains more weight the longer they remain as the newest one.

### 3.1.3 Loop closure tracking with a discrete Bayesian filter

Loop closure hypotheses are tracked by a discrete **Bayesian filter**. It estimates the probability of the current node matching a previously visited location stored in WM. The nodes in STM are not considered in loop-closure detection, since they are very similar to the current node, thus the hypotheses would be biased towards them. Instead, the number of nodes in STM is limited by a threshold determined by the estimated velocity of the camera and the rate at which nodes are updated. When the threshold is reached, oldest nodes are moved to WM. If  $S_t$  represents the states of all loop closure hypotheses at time  $t$ ,  $p(S_t = i | Z_t)$  is the probability of  $z_t$  and  $z_i$  representing the same location given the set of nodes  $Z_t = \{z_0, \dots, z_t\}$  in WM and  $p(S_t = -1)$  is the probability of  $z_t$  representing a new location, then

$$p(S_t | Z_t) = \eta p(z_t | S_t) \sum_{i=-1}^n p(S_t | S_{t-1} = i) p(S_{t-1} = i | Z_{t-1}) \quad (3.2)$$

is the **full posterior probability** of loop closure in the current node with respect to all other nodes in WM. The probability distribution is normalised to  $[0, 1]$  by  $\eta$ . The **observation model**  $p(z_t | S_t)$  is evaluated element-wise with the likelihood function

$$p(z_t | S_t = j) = \mathcal{L}(S_t = j | z_t) = \begin{cases} \frac{s_j - \sigma}{\mu}, & \text{if } s_j \geq \mu + \sigma \\ 1, & \text{otherwise} \end{cases} \quad (3.3)$$

where  $s_j$  is the similarity (3.1) between  $z_t$  and  $z_j$ , while  $\mu$  is the mean of all non-null scores and  $\sigma$  their standard deviation.

The **transition model** is evaluated by [131, 135]

$$p(S_t|S_{t-1}) = \begin{cases} 0.9, & \text{if } S_t = S_{t-1} = -1 \\ 0.1/N_{WM}, & \text{if } S_t = i \wedge S_{t-1} = -1 \\ 0.1, & \text{if } S_t = -1 \wedge S_{t-1} = j \\ \mathcal{N}_i(j, \sigma), & \text{if } S_t = i \wedge S_{t-1} = j \end{cases} \quad (3.4)$$

The number of nodes in WM is denoted as  $N_{WM}$ . If a loop closure was found in a neighbour node – *i.e.* cases when  $S_{t-1} = j$  applies – there is a high chance of finding a loop closure nearby in the next one as well since the nodes are close to each other. Therefore, the probability of a new loop closure after a previous one is given by  $\mathcal{N}_i(j, \sigma)$ . It is the value at  $i$  in a discretised normal distribution with mean  $j$  and standard deviation  $\sigma = 1.6$ . Values of the distribution are set to null outside the range  $[j - 16, j + 16]$ , and normalised to sum 0.9, so that the full distribution of the case  $S_{t-1} = j$  would sum to one. On the other hand,  $S_{t-1} = -1$  means there was no loop closure in the previous node.

When the posterior probability  $p(S_t|z_t)$  has been updated, the hypothesis in  $S_t$  with the highest probability is chosen. If the probability of a new location  $p(S_t = -1|Z_t)$  is lower than a given threshold, the loop closure is accepted and a bidirectional loop closure link is formed between the corresponding nodes. However, the old node  $z_i$  is not deleted: rather, its weight is copied to the new node  $z_t$  and then set to zero. This keeps the two different signatures of the same location, which helps further loop closure tracking, especially in dynamic environments. [131] If a loop closure is achieved, the transformation between the two views is computed with the PnP RANSAC algorithm in OpenCV. [132]

### 3.1.4 Memory management and graph optimisation

Following loop closure detection, the neighbours of the loop closure hypothesis with the highest probability not already in WM are retrieved from LTM, regardless of whether a loop closure was accepted or not. Simultaneously, the visual vocabulary is updated with the words in signatures of the retrieved nodes with the same process that new nodes' descriptors are quantised. Because the operation is quite time-consuming, a maximum of two nodes are retrieved each iteration. If more than two neighbours could be retrieved, the ones closer in time (direct neighbours) are prioritised over the ones closer in space (linked through loop closure) since they are more trustworthy. After staying still for a while, all neighbouring nodes would be retrieved.

When the time or memory constraints are reached, some of the nodes in WM need to be transferred back to LTM. The order in which they are moved to LTM depends first on their weight and – in the case of equal weights – older nodes are transferred first. For the sake of robust loop closure evaluation, the hypothesis with the highest probability and its

neighbours are not allowed to be moved to LTM. However, the neighbouring nodes are limited by a time range threshold to avoid accidentally accumulating many nodes to the same small area, which would render detection of larger loops impossible.

Whenever a loop closure is detected, or nodes are transferred between WM and LTM, the local map graph in WM is optimised. The RTAB-Map library includes three popular graph optimisation approaches: TORO [31], g2o [30] and GTSAM [136]. The authors of RTAB-Map state that TORO converges slower than the other two, but is more robust and less sensitive to odometry noise. On the other hand, GTSAM and g2o are reported to be more accurate in single session mapping, particularly with poses with six degrees of freedom. GTSAM is deemed slightly more robust than g2o and therefore chosen as the default approach. [16] Loop closure might sometimes result in invalid detections, which would add errors to the map when optimising the graph. If a link's transformation after optimisation has changed more than a certain threshold, all links added by the most recent node are rejected.

### 3.1.5 Reconstruction

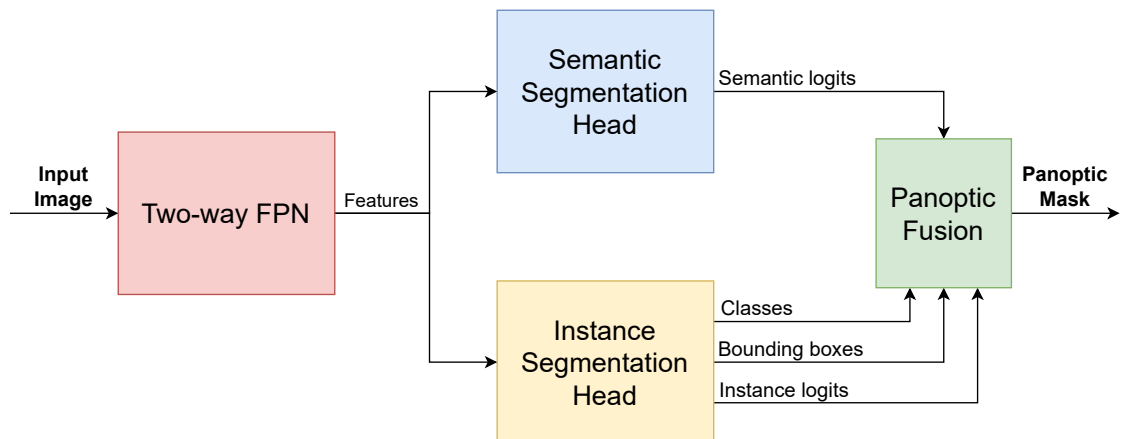
Along with node generation, STM is also tasked with generating a local occupancy grid. The occupancy grid can be either in 2D (pixel) or in 3D (voxel), depending on parameters and hardware configuration. While 2D laser scans can naturally only be converted to 2D grids, depth images and 3D scans can be converted to 3D grids or projected to 2D if three-dimensional information is not necessary. Depth images can be from an RGB-D camera, or they can be computed from stereo images with the tools in the library. Using the poses from the map graph, the local occupancy grids can be fused into a global grid.

The global occupancy grid can also be generated online, but it has to be re-assembled every time a loop closure happens. Moreover, since global graph optimisation can only be performed offline because of intense computational requirements, the online grid is less accurate than one reconstructed offline. The global grid is not needed by the SLAM algorithm and local grids are sufficient for the purpose of obstacle avoidance, thus in most use-cases it is only used for visualisation purposes. The map is topologically consistent without global optimisation and thus suitable for navigation as well.

The 3D occupancy grid is a volumetric reconstruction of the environment, which could be further processed into a surface representation. However, since the camera poses corresponding to each keyframe are available, a more detailed point cloud reconstruction can also be generated from the images or scans associated with the keyframe. The library provides offline tools for further cleaning and optimising the point clouds, as well as for generating detailed polygonal mesh surfaces out of them. The tools are convenient for visualisation and debugging purposes, but higher quality 3D meshes could be achieved with more recent tools specialised in that specific task.

### 3.2 EfficientPS: Efficient Panoptic Segmentation

EfficientPS [115] is a recent panoptic segmentation architecture that reaches top of the line accuracies in many 2D datasets more efficiently than earlier approaches with similar results. The overall incentive of the architecture is efficiency: a better trade-off between accuracy and the number of trainable parameters. It is an end-to-end top-down approach with a shared feature extractor and separate segmentation heads, and thus shares many similarities with Panoptic FPN [112]. However, the feature extractor and segmentation heads have been reworked to gain much higher accuracies with minimal increases in network size. A novel panoptic fusion method is also introduced. EfficientPS outperformed all the earlier approaches in the Cityscapes [137] panoptic segmentation benchmark and reached second places in Cityscapes semantic- and instance segmentation tasks as well. Additionally, it is also one of the most accurate methods on Mapillary Vistas [138] and Indian Driving Dataset (IDD) [139], as well as on the panoptic segmentation dataset composed from KITTI [140] by its authors. A comprehensive ablation study is performed in the article to verify the performance gains of individual parts.



**Figure 3.2.** Topology of the EfficientPS panoptic segmentation architecture. Input images are first fed through a shared feature extractor, after which semantic- and instance segmentation tasks are solved in separate heads. The segmentation heads' outputs are finally fused together in the panoptic fusion module.

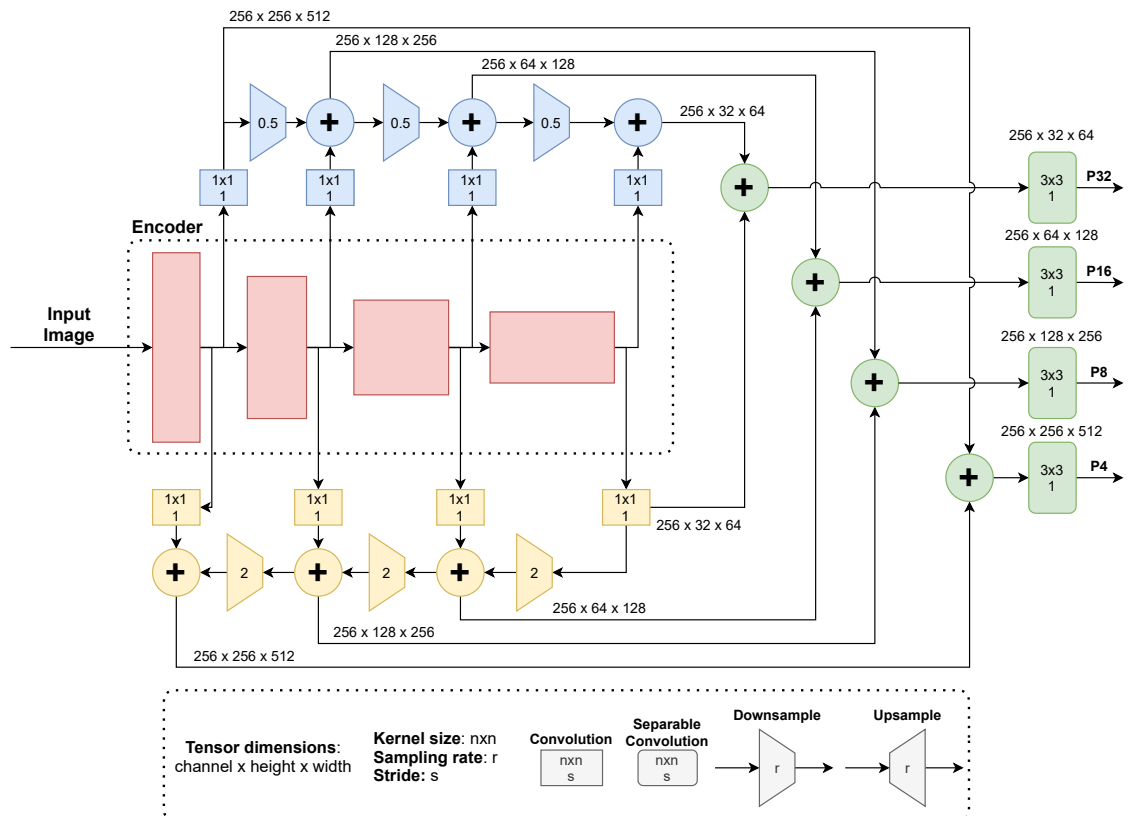
Figure 3.2 presents the overall structure of the EfficientPS architecture. First, input images are passed through a shared feature extractor composed of an encoder and a novel **two-way Feature Pyramid Network**, after which the extracted features are passed on to two different segmentation heads. The semantic segmentation head solves the semantic segmentation task, while the instance segmentation head similarly solves instance segmentations. These two outputs are then finally combined to a panoptic segmentation mask in the panoptic fusion module. Standard convolution layers are substituted with separable convolutions [141] throughout the model, which significantly reduces the number of trainable parameters with minimal effect on accuracy. Instead of commonly used Rectified Linear Unit activation function (ReLU) [142], Leaky ReLU – which is proven to



outperform the original ReLU [143] – is applied. Additionally, regular batch normalisation layers are also replaced by synchronised Inplace Activated Batch Normalisation (iABN) [144] for more efficient training with multiple GPUs.

### 3.2.1 Two-way FPN feature extractor

The two-way FPN feature extractor is visualised in Figure 3.3. The encoder isn't restricted to a single model, but the authors of the architecture utilise a modified version of the encoder part of EfficientNet-B5 [119]. Because of EfficientNets' compound scaling approach, they provide a better trade-off between the number of parameters and accuracy than earlier encoder architectures. The two-way FPN builds upon original Feature Pyramid Networks [95]: features of multiple scales are aggregated by capturing outputs – which are downsampled by different factors – from different phases of the encoder. The channel dimensions in each of the encoder outputs are first increased or decreased to the same size with  $1 \times 1$  convolution layers, after which the features are transformed to a uniform shape and added together. In the top-down branch, the encoders' outputs are upsampled, interpolated and passed on further to  $3 \times 3$  convolution layers. Similarly, the



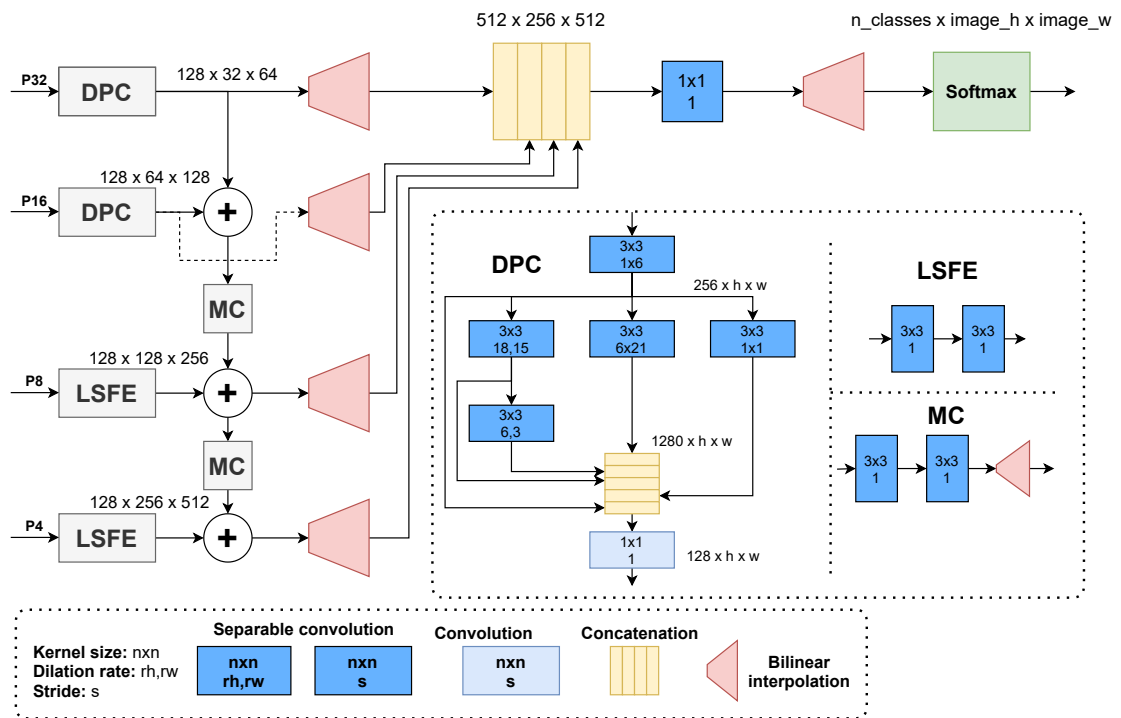
**Figure 3.3.** The two-way FPN feature extractor Image is first passed through an encoder (red), from which four outputs representing different scales of features are captured. The outputs are passed through top-down (yellow) and bottom-up (blue) feature pyramids and combined to to four separate feature maps (green).

outputs are downsampled to the same shape with average pooling and passed through  $3 \times 3$  convolution layers in the bottom-up branch. Finally, the branch outputs are summed together and passed through final  $3 \times 3$  convolution layers to generate four output feature maps of different scales.

An ablation study by the authors shows that the modified EfficientNet-B5 encoder outperforms other common encoder choices, including ResNeXt-101 [145] which has almost three times the number of trainable parameters. Similarly, the 2-way FPN was tested and found to provide better results than either bottom-up or top-down pyramid approaches alone, as well as the PANet FPN [109], where the bottom-up path follows the top-down path instead of being parallel. The authors deliberate that performance is increased because the unidirectional flow of information limits the fusion of multi-scale features, while bidirectional flow makes the fusion more effective.

### 3.2.2 Semantic segmentation head

The structure of the semantic segmentation head is presented in Figure 3.4. The four outputs of the feature extractor are processed by four different branches, which are later concatenated to fuse features of different scales together. The outputs of each branch are interpolated to a uniform shape and concatenated to a single tensor, which is then fed through a convolutional layer, upsampled to the shape of the input images and fed through



**Figure 3.4.** Semantic segmentation head. Smaller scale features are fed through the DPC modules, while features of larger scale are captured with LSFE modules. MC modules are applied to correct misalignment between features of different scales.

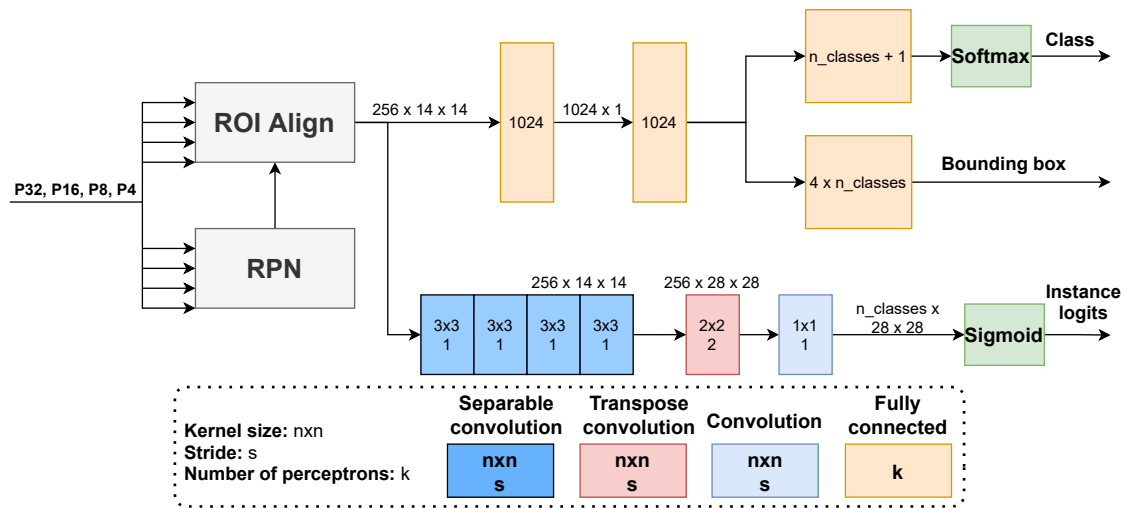
a softmax function to yield the semantic logits. A semantic mask can then be acquired with the argmax operation, *i.e.* by finding the channel from the logits corresponding to the highest confidence score for each pixel of the image.

The head has three distinct components, each targeting different requirements. The first requirement is the ability to detect large-scale features efficiently. **Large Scale Feature Extractors** (LSFE) are applied for this purpose: each module consists of two  $3 \times 3$  separable convolution layers, with 128 output filters. Second, the segmentation head should be able to capture small-scale features' context in a large range of scales. For this, **Dense Prediction Cells** (DPC) [101] are employed. They encode information in parallel branches and different scales, to capture features of varying proportions. The third requirement is the ability to attenuate mismatch between different scales, thus **Mismatch Correction** (MC) modules are placed between LSFE and DPC branches so that the model learns to align them properly. During training, the weights of the semantic segmentation head are back-propagated according to the per-pixel log-loss function [146] averaged over each batch. The proposed semantic head is shown to outperform other recent approaches in the ablation study. [115]

### 3.2.3 Instance segmentation head

The instance segmentation head – depicted in Figure 3.5 – is a modified version of the Mask-RCNN head [107]. It comprises a shared pipeline consisting of two stages followed by three output branches, which give the predicted class, bounding box and mask logits for each detected object instance. The first shared stage is the Region Proposal Network (RPN) [106]. It is a fully convolutional network that outputs several rectangular object proposals and an objectness score for each of them, in relation to each of the FPN outputs. The output features of the backbone network are passed to RPN sequentially in a sliding window manner and the resulting region proposals are then accumulated in a list. The maximum number of proposals for each window location is denoted as  $k$ . Each of the  $k$  proposals is parametrised in relation to a set of reference bounding boxes centered on the window location, called **anchors**. The scales and aspect ratios of the anchors are chosen depending on the application. The generated proposals tend to overlap, thus they are filtered with Non-Maxima Suppression (NMS).

The second shared stage of the instance segmentation head is the RoI Align module. It converts the region proposals from RPN to uniform dimensions to make them easier to process with the following output branches. It uses the proposal boxes to extract Regions of Interest from the backbone outputs, divides the RoI's into spatial bins, extracts features from said bins and aggregates them to an output. Unlike the original RoIPool module deployed with Fast RCNN [105], RoI Align – introduced with Mask-RCNN [107] – does not perform quantisation to avoid misalignments between the features and the input. It



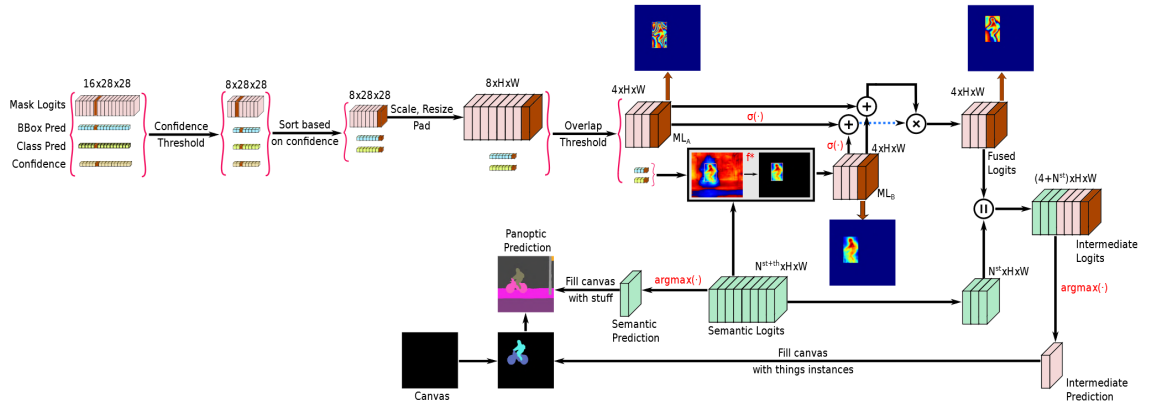
**Figure 3.5.** Instance segmentation head. RPN provides region proposals to ROI Align, which extracts features from FPN output. Objects' classes, bounding boxes and masks are inferred from the features.

instead bilinearly interpolates the feature values at four regularly sampled locations inside each ROI bin.

The object classification and bounding box regression branches share two consecutive fully connected layers and then separate to different branches. The bounding box regressor learns to align the bounding boxes specific to each class, while the classification branch outputs logits representing the confidence of the object belonging to different object classes, including a void class to catch out-of-dataset objects. The mask segmentation branch employs a fully convolutional network to predict mask logits for the area inside the bounding box. Different loss functions have been assigned to the separate parts of the head: objectness score loss and object proposal loss for RPN; and classification loss, bounding box loss and mask segmentation loss for the output branches. The total instance segmentation loss passed to the backbone in training is the sum of the five.

### 3.2.4 Panoptic fusion

The panoptic fusion module – visualised in Figure 3.6 – functions as follows. First, a set of object instances are retrieved from the instance segmentation head with object class, confidence score, bounding box and mask logits. Next, the instances are filtered by discarding the ones with confidence score lower than a given threshold. They are then sorted by their corresponding scores, padded with zeroes to retain the original aspect ratio and scaled to match the input image. Overlaps between instances are checked in the order of confidence and the overlapping instances with lower scores are discarded if the overlap is higher than a given threshold. Simultaneously, semantic logits inside each instance bounding box are extracted from the semantic segmentation head's output.



**Figure 3.6.** The panoptic fusion module of EfficientPS. [115] Object instances are fused with semantic segmentation data and pasted to a blank canvas, after which the rest is filled with semantic segmentation results.

Since the masks need to be the same size as the input image, the logits are set zero outside the area given by the bounding box. The logits from the two heads are then fused pairwise by

$$FL = (\sigma(ML_A) + \sigma(ML_B)) \odot (ML_A + ML_B), \quad (3.5)$$

where  $FL$  are the fused logits,  $ML_A$  instance mask logits and  $ML_B$  their corresponding semantic mask logits. The element-wise sigmoid function of  $M$  is defined as  $\sigma(M)$ , while  $\odot$  is the Hadamard – *i.e.* element-wise – matrix product. Argmax operation along the class dimension is used to get the intermediate instance prediction. Finally, the instance class predictions are copied to a mask template filled with void labels from the intermediate prediction and the rest of the template is completed with information from the semantic mask, ignoring areas smaller than a certain minimum threshold. This approach is shown to be significantly more accurate with respect to the Panoptic Quality metric (2.15) than simply pasting instances over semantic information.

## 4 SYSTEM DESIGN AND PERFORMANCE EVALUATION

In this chapter, the structure of the reconstruction system and related key design choices are described. The system is also evaluated quantitatively to justify the choices and to ensure it performs as intended. The objective is to design a baseline for future research, therefore the system needs to be flexible and easily extendable. It is designed specifically to be used with a handheld camera rig introduced below, but a wide range of sensors should be supported nevertheless to enable testing of different setups and applications as needed. Producing a ground-truth comparison for the whole end-to-end system would not be practical – since it would require the design and implementation of another system altogether – therefore quantitative evaluation of the system as a whole is considered out of scope for this thesis. Instead, individual components critical to the system’s performance are evaluated separately. The end-to-end results are analysed qualitatively in Chapter 5.

Metric consistency of a reconstruction is mostly dependent on localisation accuracy. Furthermore, topological consistency mostly depends on how accurately the SLAM algorithm can detect loop closures. Therefore, the most critical components in that regard are motion estimation – in this work, a device performing visual-inertial odometry introduced next – and the SLAM algorithm. Since the ground truth for testing these methods has not been generated, they are evaluated by reviewing results from other works. While there are many benchmark datasets available for offline evaluation of SLAM algorithms, RTAB-Map has already been extensively evaluated with multiple such datasets. Therefore, results already reported by its authors are referred instead and compared to similar studies of other SLAM algorithms to avoid excessive work. The level of detail in maps is also affected by the quality of depth images, but they cannot be easily evaluated quantitatively. Therefore, the evaluation of the level of detail in the reconstructions is also left to the qualitative analysis of the next chapter.

EfficientPS’s performance is first evaluated by comparing the results reported by its authors on the Cityscapes dataset [137] to other top performing works. Since there are no publicly available EfficientPS models that are trained with a dataset relevant to indoor reconstruction, the model is also trained and evaluated with the ScanNet [147] dataset. Because ground-truth poses and depth images are also provided with ScanNet – along with ground-truth labels for both individual views and reconstructed scenes – segmentation performance can be evaluated both in 2D and 3D separate from the mapping approach.



**Figure 4.1.** *The camera system used for indoor reconstruction. From top to bottom: Intel Realsense D435i RGB-D camera, Intel Realsense T265 tracking camera, Microsoft Azure Kinect RGB-D camera.*

The rest of this chapter continues as follows. The first section presents the hardware used in a practical application of this system, while the second section introduces the overall structure and important design choices. Odometry performance is evaluated Section 4.3, after which RTAB-Map is evaluated in Section 4.4. Finally, the accuracy of EfficientPS’s segmentation is evaluated in Section 4.5.

## 4.1 Hardware

Figure 4.1 presents the camera setup used in the practical applications presented in this thesis. Three different RGB-D cameras are available, two of which are present in the image: Intel Realsense D435i [148] and Microsoft Azure Kinect [149]. The third camera is the Intel Realsense D455 [150], which is a newer version of D435i with a wider stereo baseline. When connected to a laptop, the handheld camera rig can be used as a portable 3D reconstruction device.

The Intel Realsense T265 tracking camera [151] – the middle one in Figure 4.1 – is employed as an external odometry device for RTAB-Map algorithm. Its stereo camera, wide field of view and inertial sensor help provide a robust local track. However, since the underlying SLAM algorithm only maintains a small local map, it cannot close larger loops and thus is not suitable for large-scale mapping by itself. Because of its dedicated hardware, VIO is computed on the camera itself, which reduces computational load on the host system. The camera’s internal loop closure is turned off to not interfere with

RTAB-Map. Research on the performance of the T265 tracking camera is reviewed in Section 4.3 to justify its usage in lieu of other options.

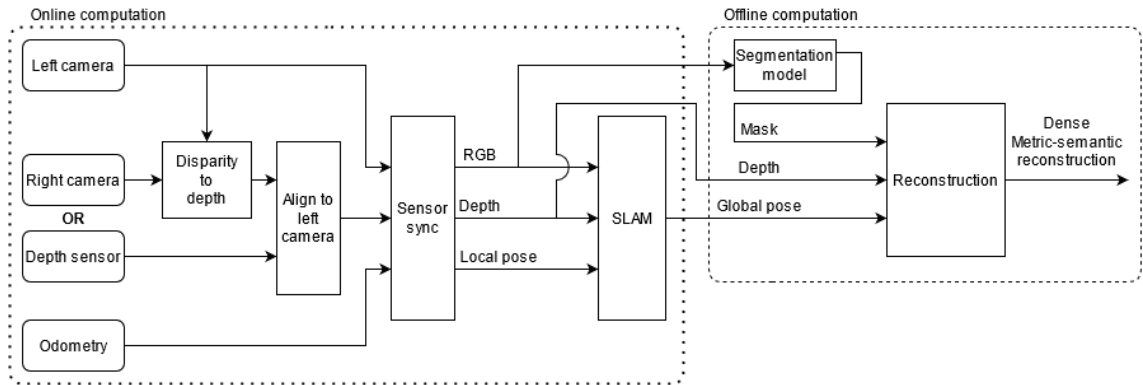
The camera rig is mostly limited to indoor use-cases because of the cameras' limited range of depth perception. However, because the cameras capture images at a high resolution, detailed three-dimensional reconstructions can be created when longer ranges are not required. Because of the inertial sensor of T265, the local track is not immediately lost even if there are not enough visual features in the effective range of the camera, but the RGB-D cameras' depth perception is limited to only a few meters.

## 4.2 Design choices and general structure

A flow diagram of the system is presented in Figure 4.2. As declared in 2.1.3, for the map to be applicable in real-world use-cases, the scale of the environment has to be captured. On the other hand, colour images are required for panoptic segmentation. Therefore, the minimal required inputs are colour- and depth images. Odometry is provided as an additional input, although it could be inferred from the sequences of colour- and depth images as well. Depth can either be given as an already processed range image or computed from stereo images. If stereo cameras are used, the image of the left camera is chosen for colour input. Depth images are aligned with colour, after which both images and possible odometry messages are synchronised according to their timestamps. Corresponding to each map node generated by SLAM, the views – including colour, depth and pose – are saved to a database with a unique id and a timestamp. The individual views can then afterwards be retrieved from the database in chronological order, segmented and reconstructed to the desired representation. This design separates sensors, SLAM, segmentation and reconstruction into separate modules, which allows different implementations of single modules to be tested without the need to modify others. RTAB-Map is employed as the visual SLAM approach in the applications of this thesis, while EfficientPS is the preferred segmentation model. Offline reconstruction is performed with tools provided in the RTAB-Map library.

**Robot Operating System** (ROS) [152] is chosen as middleware, since it is specifically designed for prototyping robotics and machine perception. With ROS, each individual task can be separated into its own topic, making them interchangeable with other topics, given their interface is the same. The communication interface between ROS topics is highly efficient and flexible. The manufacturers of the cameras introduced in the previous section – as well as most other manufacturers of robotics hardware – provide ROS drivers, which transform the images and tracking camera's output to ROS messages. Basic tools for robotics and machine vision are also provided with the library, and many widely used open-source third-party tools are available as well. Each individual module is separated into their own **Docker** [153] container, which reduces dependency issues and allows each





**Figure 4.2.** Block diagram of the end-to-end system. Individual views – including colour images, depth and pose – are captured online, after which they are segmented offline and reconstructed to a dense 3D metric-semantic reconstruction.

of them to be developed separately in their own environment.

Because the SLAM algorithm has to maintain a global map to keep the path globally consistent, online global optimisation is not possible with large maps. However – since a graph-based approach has been chosen – the map can be optimised offline preceding reconstruction. Given enough computational resources are available, both segmentation and reconstruction could be performed online as well if a globally optimised reconstruction is not necessary. While real-time reconstruction is not needed for the use-cases in this thesis, the option of online segmentation and reconstruction is still provided for the purposes of successive research. Segmentation could be performed every time a node is generated and – if a sequential reconstruction method is used – the views could be fused together as well. Distributed computing could provide enough computational resources for the system to operate in real time. If end-to-end real-time operation would be required, the system would be fairly straightforward to separate to different machines because of its implementation in ROS and Docker.

### 4.3 Odometry performance

In [44], the Intel Realsense T265 tracking camera and ORB-SLAM2 [17] are evaluated from a person tracking standpoint, while in [45], they are compared – in addition to the ZED Mini [154] stereo camera – in a mobile robotics use-case. In the former case, the ground truth is acquired with a VICON [155] motion capture system, while in the latter they are compared against a similar OptiTrack [156] system. Both conclude that the Realsense camera is the best choice for their use case.

In [44], the performance of T265 and ORB-SLAM2 in both localisation and pose tracking are found to be almost equivalent, but the T265 is deemed more reliable by the authors because of its higher framerate and its noise distribution being closer to Gaussian. On the other hand, in [45] ORB-SLAM2 localisation is found to be clearly inferior to both T265 and

ZED cameras, which both had similar accuracies over all test sequences. This difference is the result of large computational load of the odometry approach of ORB-SLAM2 on the host system: while the authors of [44] were able to run the algorithm in a stereo setup with an average output rate of  $20\text{ Hz}$ , in [45] it could only be run in monocular configuration with an average rate of  $5.5\text{ Hz}$  because of hardware restrictions in the mobile robot platform. Additionally, the T265 and ZED cameras fuse IMU readings with the visual data, while ORB-SLAM2 used purely visual data, rendering it less robust when there are fewer visual features available or there are moving objects in the environment.

Notably, because the T265 runs its SLAM algorithm on-board with dedicated hardware, it does not require computation on the host system and has less delay between measurements and pose estimation. In [45], the T265 could be connected directly to the robots Raspberry Pi board, while the ZED camera and ORB-SLAM2 required an additional Nvidia Jetson TX2 board, because in their case computation had to be performed on the host computer. Moreover, ORB-SLAM2 had to be run on monocular images, since stereo setup resulted in too low framerate to operate in real time. The T265 is capable of maintaining a stable rate of  $200\text{ Hz}$ , while ZED averaged at around  $20\text{ Hz}$  and ORB-SLAM2 at  $5.5\text{ Hz}$ . Performing the computations onboard with dedicated hardware is clearly preferable to computing on the host system, at least when computational resources are scarce and real-time positioning is required.

Based on the results referred above, the Realsense T265 -camera seems to be a good choice for odometry. However, it is mostly intended for indoor use-cases, thus the odometry method might need to be updated if the system is applied to an outdoors scenario. Since the odometry of T265 is computed entirely on its own dedicated hardware, the computational requirements for the host machine are significantly lower, which is advantageous especially for handheld operation.

#### 4.4 SLAM performance

RTAB-Map's performance in different configurations on the EuRoC MAV dataset [157] is reported in [16]. The dataset is a popular benchmark for visual SLAM systems, thus RTAB-Map's performance can be evaluated against many other similar approaches. It contains stereo images and IMU measurements collected with a Micro Aerial Vehicle and ground-truth trajectories estimated within one millimeter precision. It contains three distinct datasets: Vicon Room 1 and 2 (V1 and V2) both contain three sequences collected in a room with a VICON motion capture system with different obstacle configurations, while The Machine Hall (MH) dataset has five sequences collected in a machine hall at ETH Zurich. The ground truth poses in Vicon Room sequences have six degrees of freedom, while the ones in Machine Hall sequences have only their 3D position captured with a Leica MS50 laser scanner. The sequences of each set are ordered by increasing diffi-

**Table 4.1.** SLAM accuracy comparison on the EuRoC MAV dataset. Comparison metric is Absolute Trajectory Error (ATE) in centimeters in relation to ground-truth estimate. The results are combined from [16] and [158].

Method	EuRoC V1			EuRoC V2			EuRoC MH					$\mu$ <sup>1</sup>
	01	02	03	01	02	03	01	02	03	04	05	
ORB2	<b>3.5</b>	<b>2.0</b>	4.8	3.7	3.5	-	3.5	1.8	2.8	12	6.0	4.36
ORB3	<b>3.5</b>	2.1	4.9	3.2	2.7	36	2.5	2.2	2.7	8.9	5.8	3.85
ORB3 + IMU	3.7	1.4	<b>2.3</b>	3.7	<b>1.4</b>	<b>2.9</b>	3.7	3.1	<b>2.6</b>	<b>5.9</b>	8.6	<b>3.64</b>
Kimera	5.0	11	12	7.0	10	19	8.0	9.0	11	15	24	11.2
ORB2-RTAB	7.8	2.4	18	11	5.5	-	<b>1.8</b>	<b>1.5</b>	<b>2.6</b>	11	<b>5.3</b>	5.69

<sup>1</sup> Mean error, sequence V2\_03 is excluded for fair comparison.

culty: in MH\_01 the drone moves slowly and lighting is bright, while in later sequences the drones move faster and the lighting is darker, for example. An accurate point cloud reconstruction of the Vicon Room is also provided in both V1 and V2.

Of all visual SLAM approaches, ORB-SLAM2 [17] is perhaps the closest to RTAB-Map in its implementation. Furthermore, ORB-SLAM3 [158] improves upon ORB-SLAM2 with a more sophisticated loop closure mechanism and IMU fusion. They are also among the top performers on the EuRoC MAV dataset, and thus a good reference for RTAB-Map’s performance. Kimera [159] is a metric-semantic reconstruction system quite similar to the one designed in this thesis. It similarly performs visual-inertial odometry and loop-closures based on graph optimisation, but its segmentation is performed online. Since it has been tested on EuRoC MAV in the above paper as well, it is added to the comparison. In [16], RTAB-Map is reported to have the best performance when utilising the visual odometry of ORB-SLAM2 as an odometry input for RTAB-Map. This configuration is also interesting in the sense that odometry is the same as in the full ORB-SLAM2, thus differences between the two should only result from the differences outside the odometry approach, which in our case is evaluated separately. Therefore, the performance of RTAB-Map is reported with the ORB-SLAM2 odometry configuration.

The Absolute Trajectory Errors (ATE) [160] of the SLAM systems on the EuRoC MAV dataset are listed in Table 4.1. ORB-SLAM3 with IMU fusion is clearly the most accurate of the algorithms tested, with both the lowest mean error as well as most top accuracies in the sequences. Surprisingly, ORB-SLAM2 isn’t that far behind ORB-SLAM3 – when no IMU readings are used – and in some cases even surpasses the newer method. RTAB-Map seems to be less accurate compared to the two in Vicon Room sequences but wins in most of the Machine Hall sets. Since Machine Hall ground truth only has the 3D position of the drone while Vicon Room has the full 6-DOF pose, it seems that RTAB-Map is quite good at tracking the location of the camera, but orientation tracking might be slightly worse. Its performance is still quite good compared to Kimera, which for some reason

performs relatively poorly even with IMU fusion.

The last sequence of V2 seems to be quite hard for visual odometry: the methods with ORB2 odometry get lost and cannot finish the sequence, while ORB3 also has much larger errors compared to its results on other sequences. On the other hand, IMU fusion seems to help with the sequence a lot. This is the only occurrence where Kimera wins all the purely visual approaches, while in the case of ORB3 with IMU there isn't much difference to the rest of the sequences. This supports the claim that IMU fusion makes visual odometry more robust. The test sequence in question has dynamic objects, fast movements and motion blur, which can cause problems for VO, while inertial sensors are less sensitive to such phenomena.

Based on the results presented above, RTAB-Map is a solid choice for a baseline SLAM algorithm, at least when external odometry is provided. It provides quite good performance – which is in line with other modern approaches – and is also highly scalable and flexible as mentioned in Section 3.1. The advantages of using both visual and inertial sensors are evident in the test results as well. If more accurate mapping is required, the update of loop-closure method or bundle adjustment – for instance to ORB-SLAM3 – would most likely gain most results.

## 4.5 Segmentation performance

Table 4.2 presents a comparison of results reported in the original EfficientPS work [115] and other top-performing panoptic segmentation methods on the Cityscapes [137] dataset. The set consists of annotated video frames captured from a moving vehicle in 50 different cities. The training set contains 3475 images with detailed annotations, called the Cityscapes fine dataset, and 20000 images with coarser annotations. Every model in the comparison is trained with the fine training set. Some models are also pre-trained with additional data, in which case the dataset in question is also mentioned in the table. The models are evaluated on a test set with a hidden ground-truth – comprising 1525 images – by uploading predictions to an evaluation server. Panoptic annotations in both train and test sets contain 11 stuff classes and 8 thing classes. Inference speed can not be evaluated accurately without running all the models on the exact same hardware setup, thus the efficiency of each model is estimated by the number of trainable parameters and floating-point operations (FLOPs) performed on each forward pass.

EfficientPS outperforms all the other top-performing proposal-based models – SSAP [118], TASCNet [116] and Seamless [114] – in the comparison, as well as Panoptic Deeplab [117], the most accurate proposal-free method as of the end of 2020. However, Panoptic Deeplab seems to be significantly more accurate in predicting thing classes. In [161] – submitted in January of 2021 – the Panoptic Deeplab [117] model was extended with scalable wide residual network backbones [162]. The newly submitted results are the

**Table 4.2.** Panoptic segmentation performance comparison on the Cityscapes test set.

Model	PT <sup>3</sup>	PQ (%) <sup>1</sup>			SQ (%)	RQ (%)	Param. <sup>2</sup> (M)	FLOPs <sup>2</sup> (B)
		St	Th	All				
SSAP [118]	-	66.5	48.4	58.9	82.4	70.6	-	-
TASCNet [116]	CO	66.0	53.4	60.7	81.0	73.8	51.43	514.00
Seamless [114]	MV	67.5	56.0	62.6	82.1	75.3	51.43	514.00
Panoptic-Deeplab [117]	-	52.1	69.7	62.3	82.4	74.8	46.73	547.49
Panoptic-Deeplab [117]	MV	58.8	<b>72.0</b>	67.0	83.5	78.8	46.73	547.49
P-DL+SWideRNet [161]	-	-	-	60.8	-	-	<b>37.19</b>	576.83
P-DL+SWideRNet [161]	-	-	-	61.8	-	-	77.97	1199.85
P-DL+SWideRNet [161]	MV	<b>72.8</b>	60.9	<b>67.8</b>	<b>83.8</b>	<b>80.2</b>	>569.46 <sup>4</sup>	>1997.73 <sup>4</sup>
EfficientPS [115]	-	60.4	56.7	64.1	82.6	76.8	40.89	<b>433.94</b>
EfficientPS [115]	MV	71.6	60.9	67.1	83.4	79.6	40.89	<b>433.94</b>

<sup>1</sup> St and Th refers to mean PQ of stuff and things, All to the mean PQ of all classes.

<sup>2</sup> Floating point operations in one forward pass of the network. Input image size is 1024 × 2048.

<sup>3</sup> Data used to pre-train the model before training on Cityscapes fine dataset.

CO: MS COCO [82], MV: Mapillary Vistas [138]

<sup>4</sup> Not specified for applied model, lower limit is chosen based on closest model parameters.

most accurate of all Cityscapes submissions, but only when the backbone is scaled to increase the model’s size over ten times the size of EfficientPS. Consequently, EfficientPS is the second smallest models in terms of trainable parameters, as well as the fastest when comparing the number of floating-point operations per input image. Pre-training a model on a similar task seems to improve its performance significantly, probably because the number of images in the Cityscapes fine training set is relatively small. On larger datasets, the effect on accuracy would most likely be less significant, but the pre-trained models would learn the task in question faster because of prior knowledge. [163]

It is not practical to evaluate the segmentation model on the actual use-cases of this thesis since a large number of images would have to be annotated by hand and other models would have to be trained on the set as well to get a reference to compare EfficientPS to. Therefore, the ScanNet [147] dataset is used. It consists of video sequences containing 2.5 million RGB-D images with camera poses and ground-truth segmentations collected in different indoor scenes. The ground truth of a test partition of the dataset is hidden to assure fair evaluation. The results on the test set are evaluated by sending predictions to an evaluation server hosted by the authors of the dataset. A subset of 25 thousand images – frames sub-sampled from video sequences approximately every 100 frames – are provided with ground truth for training image segmentation models. Training with the intermediate frames most likely would not improve results much, since they are most often really similar to ones already in the subset. There are only two stuff classes in the dataset – wall and floor – which are relatively easy to segment. However, there are also 18 thing

**Table 4.3.** Panoptic quality of individual classes on the ScanNet validation set.

	wall	floor	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	refrigerator	shower curtain	toilet	sink	bath tub	other
PQ	58	74	24	29	17	38	35	22	39	4	34	21	25	24	33	48	53	31	53	44
SQ	79	84	74	75	67	82	72	70	75	76	82	64	70	75	77	80	81	66	82	78
RQ	73	88	32	39	25	47	49	31	51	6	41	33	35	33	43	60	65	47	64	56

**Table 4.4.** Average panoptic quality on ScanNet the validation set.

	PQ	SQ	RQ	#classes
Stuff	66.1	81.9	80.5	2
Things	31.8	74.8	42.0	18
All	35.3	75.5	45.9	20

classes related to objects that can be found from indoor environments, segmentation of which is significantly harder.

The dataset is divided into scenes from which the sequences are captured. To get a truthful estimate of the model’s performance on the test set – and to evaluate panoptic quality – a subset of these scenes comprising roughly five percent of the 25 thousand frames in of the training set is separated from training data and only used for validation of trained models. From here on, the subset used for validation is denoted as the validation set, while the rest of the 25 thousand images are called the training set. The test data with hidden ground truth is called the test set.

Although the ScanNet dataset can be converted to a panoptic format for training, there is no panoptic benchmark provided for test data. Therefore, the performance of panoptic segmentation is only evaluated on the validation set and performance on the test set is evaluated on semantic segmentation, both in 2D and 3D. To transform the segmentation into the 3D evaluation format, scenes are reconstructed from individual segmented views in addition to depth images and ground truth poses provided with the dataset. Because the evaluation format requires vertices to be labelled in the original mesh provided with the dataset, their labels are inferred from the reconstruction with an approximate nearest neighbour search implemented with the Annoy [164] library. The label of the vertex is determined by a majority vote over a number of nearest points in the reconstruction.

Table 4.3 presents panoptic quality on the validation set on each individual class, while Table 4.4 provides a more concise summary of the results. Panoptic quality seems to be on par with Cityscapes results on stuff classes, but considerably lower on thing classes.

**Table 4.5.** Semantic segmentation performance on the ScanNet test set.

Model	2D IoU <sup>1</sup>			3D IoU <sup>2</sup>		
	stuff	things	all	stuff	things	all
ScanNet Baseline [147]	0.632	0.296	0.330	0.612	0.272	0.306
MSeg1080_RVC [165]	0.793	0.451	0.485	-	-	-
RFBNet [166]	<b>0.869</b>	0.561	0.592	-	-	-
SSMA [167]	0.837	0.548	0.577	-	-	-
AdaptNet++ [167]	0.814	0.469	0.503	-	-	-
MCA-Net [168]	0.837	<b>0.569</b>	<b>0.595</b>	-	-	-
PanopticFusion [123]	-	-	-	0.709	0.509	0.529
OccuSeg [169]	-	-	-	<b>0.932</b>	<b>0.745</b>	<b>0.764</b>
EfficientPS (ours)	0.785	0.440	0.475	0.623	0.394	0.417

<sup>1</sup> Average intersection over union of images compared to 2D ground truth.

<sup>2</sup> Average intersection over union of reconstruction compared to 3D ground truth.

Segmentation quality is quite uniform across the set of labels, but recognition quality is a lot lower, which generates the difference in PQ.

There seems to be a lot of variance between different thing classes. There could be many reasons for this, which are hard to determine explicitly because of the black-box nature of neural networks. Some classes are undoubtedly easier to mix with one another: for example, upon inspecting a confusion matrix on semantic segmentation results on the validation set – provided in Appendix A due to its large size – the ‘cabinet’ class is often mistaken for ‘bookshelf’ or ‘door’, which is understandable since most cabinets have doors and are also often really similar to bookshelves. Similarly, desks, tables and counters can appear visually really similar, as well as shower curtains compared to other curtains. The problem could also be amplified by the uneven distribution of classes. For example, since there are much fewer shower curtains than regular ones in training data – 0.04% versus 0.7% of all labelled points according to [147] – the model could be biased towards the latter, generating false-positive regular curtain detections and missing most shower curtains. These mistakes are easy to make if the larger context is missed, *e.g.* that the curtain in question is located in a bathroom. Based on these findings, these effects are also investigated visually in the next chapter for further proof.

Table 4.5 presents a comparison of segmentation performance on the ScanNet test set. The 3D baseline is generated with the volumetric CNN introduced in [170], which has been projected to the original views to generate the 2D baseline. EfficientPS clearly outperforms the baseline provided with the original paper in 2017, both in segmentation of individual views and reconstructed 3D scenes. An HRNet trained with MSEG [165] – MSeg1080\_RVC; the only other model in the comparison trained with purely 2D visual data – is a bit more accurate than EfficientPS. Its performance is quite remarkable con-

sidering that it isn't trained on ScanNet data. However, the HRNet in question is also a significantly larger model: it contains 65.8 Million trainable parameters and requires 696.2 GFLOPs per image. [96] When running the models on ScanNet data, panoptic segmentation with EfficientPS takes 0.22 seconds on average per image on an Nvidia Tesla V100 graphics card, while HRNet segments one image in around 1.60 seconds on the same exact hardware. Furthermore, the results on HRNet presented in this thesis are generated by passing images through the model in multiple scales, which takes around 12 seconds per image. Also, since it is focused solely on the semantic segmentation task it might learn it more effectively. The combination of datasets in MSEG also helps the model to generalise better for multitude of tasks – as is demonstrated in the next chapter – the downside being an extremely long training process.

RFBNet [166], SSMA [167], AdaptNet++ [167] and MCA-Net [168] are semantic segmentation models that also utilise depth information in different ways. They are significantly more accurate than the models not applying depth in addition to colour images. PanopticFusion [123] – which was already introduced earlier in Section 2.2.4 – is a views-based panoptic 3D segmentation approach with a sophisticated instance tracking method. It also regularises reconstructed scenes with a fully connected CRF. Similar to the original panoptic segmentation paper [72], it applies PSPNet [89] and Mask RCNN [107] separately with a naive panoptic fusion for 2D segmentation. Therefore, superiority over EfficientPS is most likely the result of the 3D tracking approach and regularisation. Occuseg [169] – a volumetric map-based segmentation method – is currently the most accurate method on the ScanNet semantic segmentation benchmark. Map-based methods seem to perform much better than view-based segmentation when the point clouds are as accurate as the ones in ScanNet data. However, with lower quality point clouds, view-based methods could be more robust. If point clouds are deemed accurate enough, map-based methods could also be considered to be used with the system in the future.
























## 5 APPLICATION IN PRACTICE

This Chapter presents results from practical applications of the system and analyses them quantitatively. The purpose is to visually identify the causes to – and effects of – results in the previous quantitative study, assess how well the system works as a whole and evaluate its value in practical applications. Together with the qualitative evaluation of the previous chapter, bottlenecks can be identified to find which parts of the system should be improved for greatest performance gains.

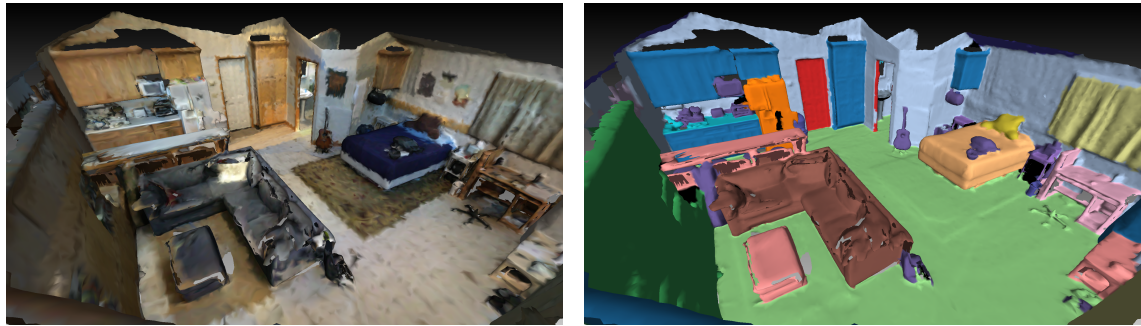
First, the quality and level of detail of segmentation are assessed by visually inspecting segmentation results on the ScanNet dataset with known camera poses. This way, localisation performance does not affect the results. Next, the system is evaluated on data collected with our handheld camera rig on two indoor locations in university campus environments. The localisation configuration remains the same in both cases: the Realsense T265 tracking camera is applied for odometry and RTAB-Map is used for global positioning. However, different RGB-D cameras and segmentation approaches are tested. The first dataset is collected with a Realsense D455 camera, while the second is captured with an Azure Kinect.

In addition to the EfficientPS model trained on ScanNet data, an HRNet [96] model trained with the MSEG [165] dataset is also applied to both cases. Models trained on MSEG tend to generalise really well to many different tasks, thus it provides a good comparison for the ability of the ScanNet-trained EfficientPS to generalise outside of training data. The authors of MSEG have publicly provided HRNet models trained on the dataset, thus it is chosen to avoid the additional work of training a model with MSEG, which – given the size of the dataset – would take a really long time. Unfortunately, although the authors have also announced a Panoptic FPN [112] model to be trained on MSEG data it has not yet been published as of writing this thesis. Panoptic results are visualised as semantic segmentation to avoid having to track instance ID's – which is not necessary in the scope of this thesis – and to compare them against the semantic segmentation of HRNet.

Each class in both datasets is associated with a unique colour for visualisation. Figure 5.1 lists colours associated with ScanNet evaluation classes. They are a subset of 20 classes out of the 40 total in the dataset that are used for evaluating the hidden test set. The 2D training data also only contains these classes, thus they are the ones that our EfficientPS

 0 : void	 6 : sofa	 12 : counter	 34 : sink
 1 : wall	 7 : table	 14 : desk	 36 : bathtub
 2 : floor	 8 : door	 16 : curtain	 39 : otherfurniture
 3 : cabinet	 9 : window	 24 : refrigerator	
 4 : bed	 10 : bookshelf	 28 : shower curtain	
 5 : chair	 11 : picture	 33 : toilet	

**Figure 5.1.** Colours associated with ScanNet evaluation classes.



**(a)** Original 3D reconstruction

**(b)** Ground-truth annotations

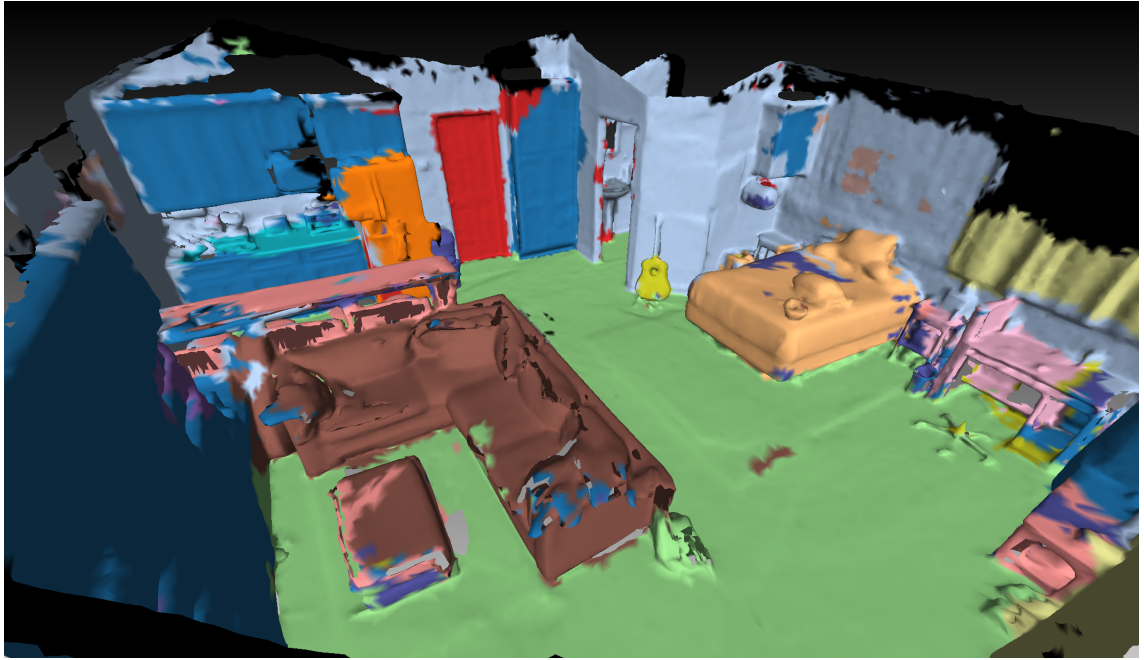
**Figure 5.2.** A scene from ScanNet training set.

model can predict. Due to the large number of classes in MSEG, their associated colours are listed in appendix B. Also – because of the number of classes – some of the randomly assigned MSEG colours are also quite close to each other in appearance, thus these classes can sometimes be quite hard to differentiate visually. The class labels in MSEG that match ScanNet evaluation classes have been associated with the same colours as presented in Figure 5.1.

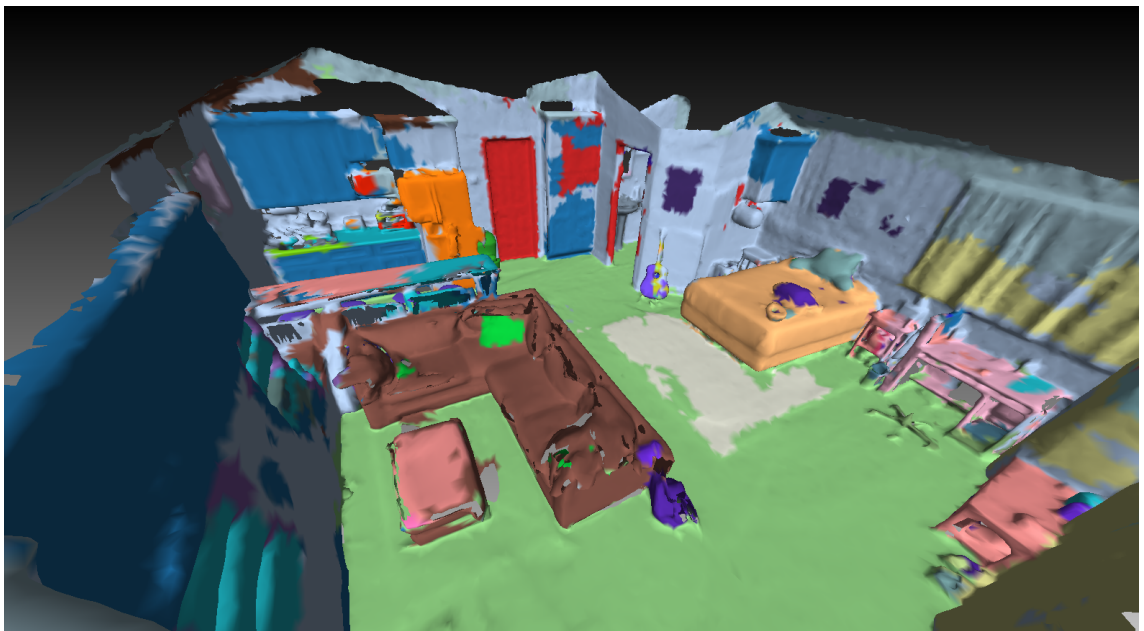
## 5.1 3D segmentation of the ScanNet Dataset

For the sake of brevity, only two ScanNet scenes are inspected in this section. Additional examples are provided in Appendix C. Because ground truth annotations are only provided for the training set, a scene of training data is inspected first. Afterwards, a scene from the test set is analysed to see the difference in performance on scenes the model has not seen before. Figure 5.2 presents an overall view of the training set scene. The mesh is the one provided with the dataset, and ground-truth labels are projected to it with the official ScanNet toolset. The 3D ground-truth also contains other classes not present in the evaluation set.

Figure 5.3 presents the EfficientPS segmentation projected to the mesh, while 5.4 similarly shows segmentations from the HRNet trained with MSEG. In both cases, segmentation quality seems quite good: stuff classes are segmented quite accurately, and most things are labelled correctly. Naturally, some objects are also labelled wrong, and there

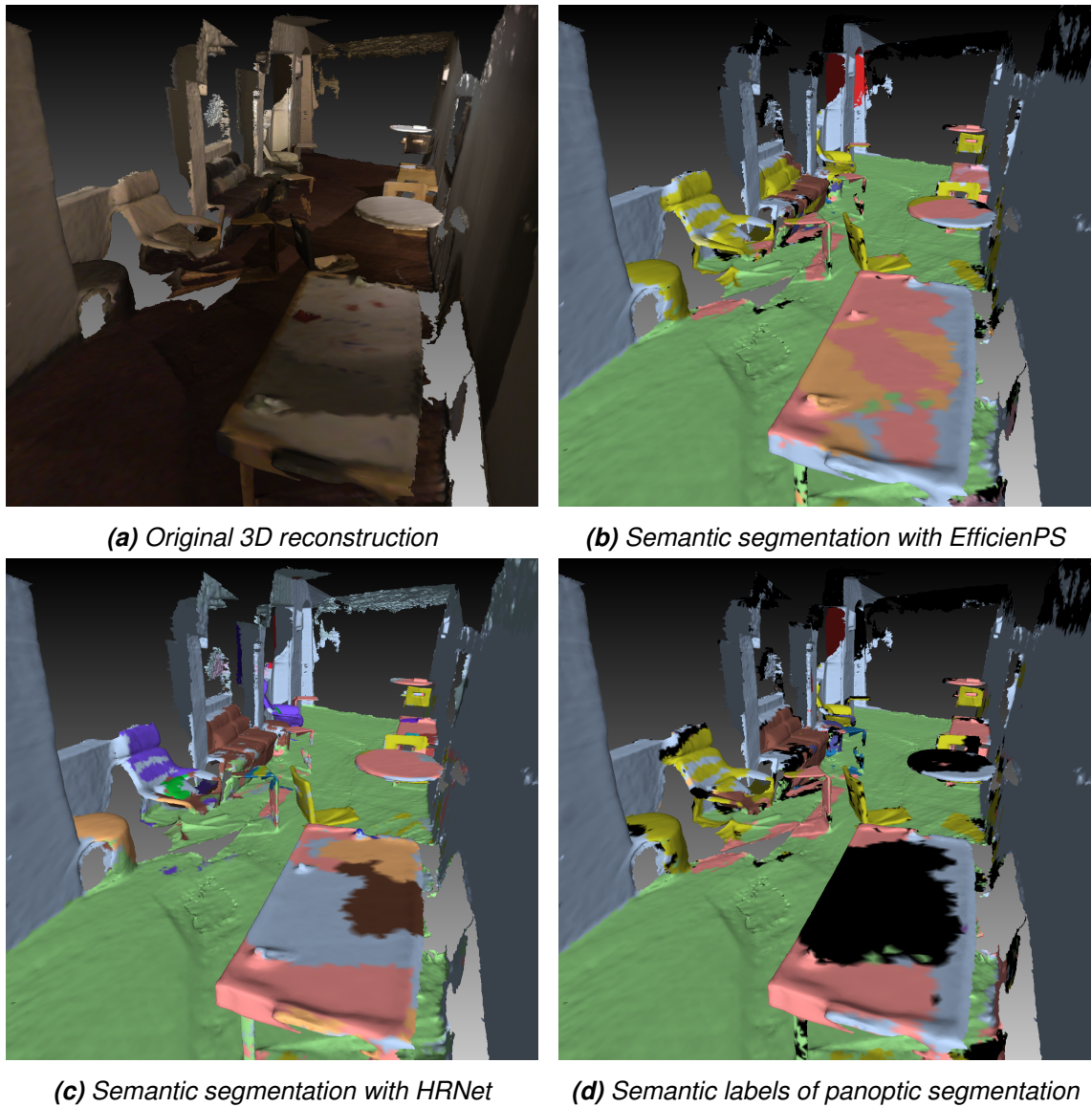


**Figure 5.3.** The ScanNet training scene segmented with EfficientPS trained on ScanNet.



**Figure 5.4.** The ScanNet training scene segmented with HRNet trained on MSEG.

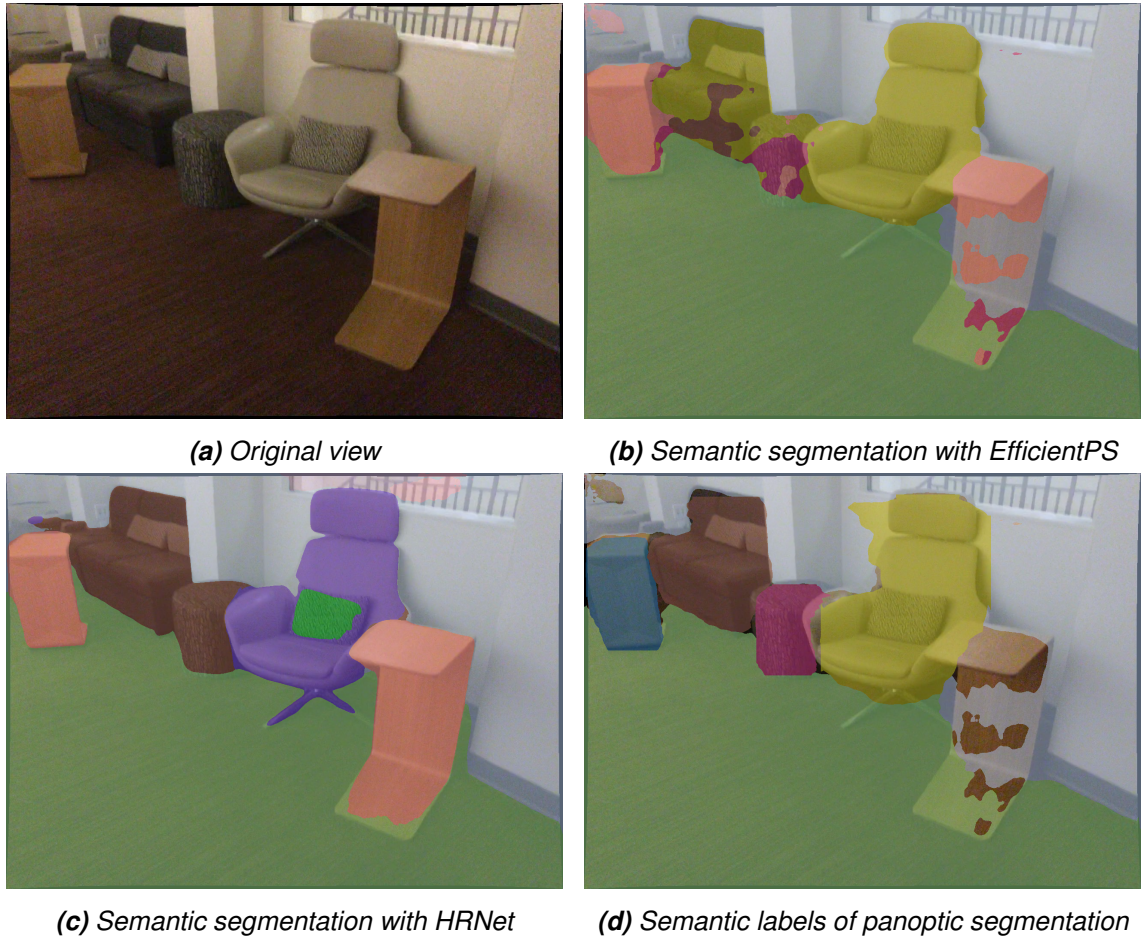
is some segmentation noise. As mentioned earlier, some classes are quite close to each other, which is pretty apparent in these images: for example, tables and counters get confused with one another often and some parts of cabinet doors are labelled as doors, not cabinets. Like in all datasets captured in the real world, there are some erroneous and confusing ground-truth annotations as well. For example, none of the pictures seen on walls are annotated in ground truth, and the footstool next to the sofa – made of similar cloth – is annotated as a table. Due to view-based segmentation, some labels are also projected behind the segmented objects, if edges of the segmentation are spread outside



**Figure 5.5.** A scene from ScanNet test set.

the object in question. For example, chair labels are projected past the office chair – of which only the base is visible due to errors in reconstruction – into the desk in the right side of Figure 5.3. HRNet seems to be able to generate more precise edges, thus the effect is less apparent in Figure 5.4. The huge number of classes in MSEG clearly has both benefits and downsides: many objects not found in ScanNet – like carpets and pillows – are labelled, but on the other hand, segmentations are quite noisy in some cases and the model confuses labels more often.

The quality of test scenes seems roughly similar to training scenes. Results on a scene from ScanNet test set have been collected in Figure 5.5. There is a bit more confusion of labels present in this scene, especially in the EfficientPS’s semantic segmentation head’s output. Even though instances cannot be tracked in 3D with the current system, converting panoptic segmentation to semantic segmentation format helped in cases



**Figure 5.6.** An individual view from the ScanNet test scene.

where segmentations were noisier. The noise in the semantic segmentation head’s output is most likely because it does not predict a ‘void’ label – a class where it could place regions of the image it is unsure of. While the instance segmentation head generates a single label for each object instance, the semantic head can generate multiple labels to different areas of an object. Panoptic segmentation, however, seems to also place some objects correctly predicted by the semantic segmentation head to the void class. Therefore, the choice between the two is a trade-off between precision – the ability to filter out false positives – and recall – the ability to find more true positives. The degree to which false positives – and some true positives with them – are filtered can be adjusted with the confidence threshold in the panoptic fusion module of EfficientPS. Thus using the panoptic output even in cases where only semantic labels are applied to the reconstruction could be beneficial.

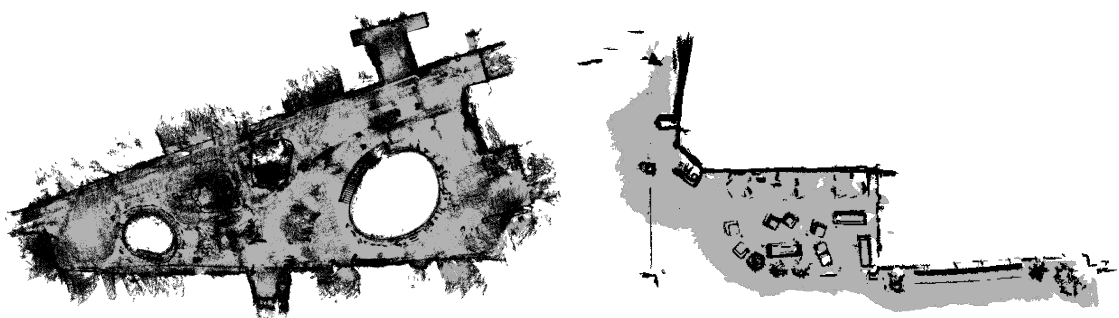
Figure 5.6 presents 2D results on a single view of the scene presented in Figure 5.5. The segmentation quality of HRNet is significantly better: edges are a lot smoother and follow outlines of objects more closely than in either of the EfficientPS outputs. As mentioned earlier, labels outside of an object’s boundaries will be projected to areas behind them in the reconstruction, thus HRNet’s segmentations are a lot cleaner in 3D as well. Also, al-

though labelling in EfficientPS's panoptic output is more coherent, its segmentation quality seems to be worse than the semantic segmentation head's.

## 5.2 Indoor reconstruction of campus environments

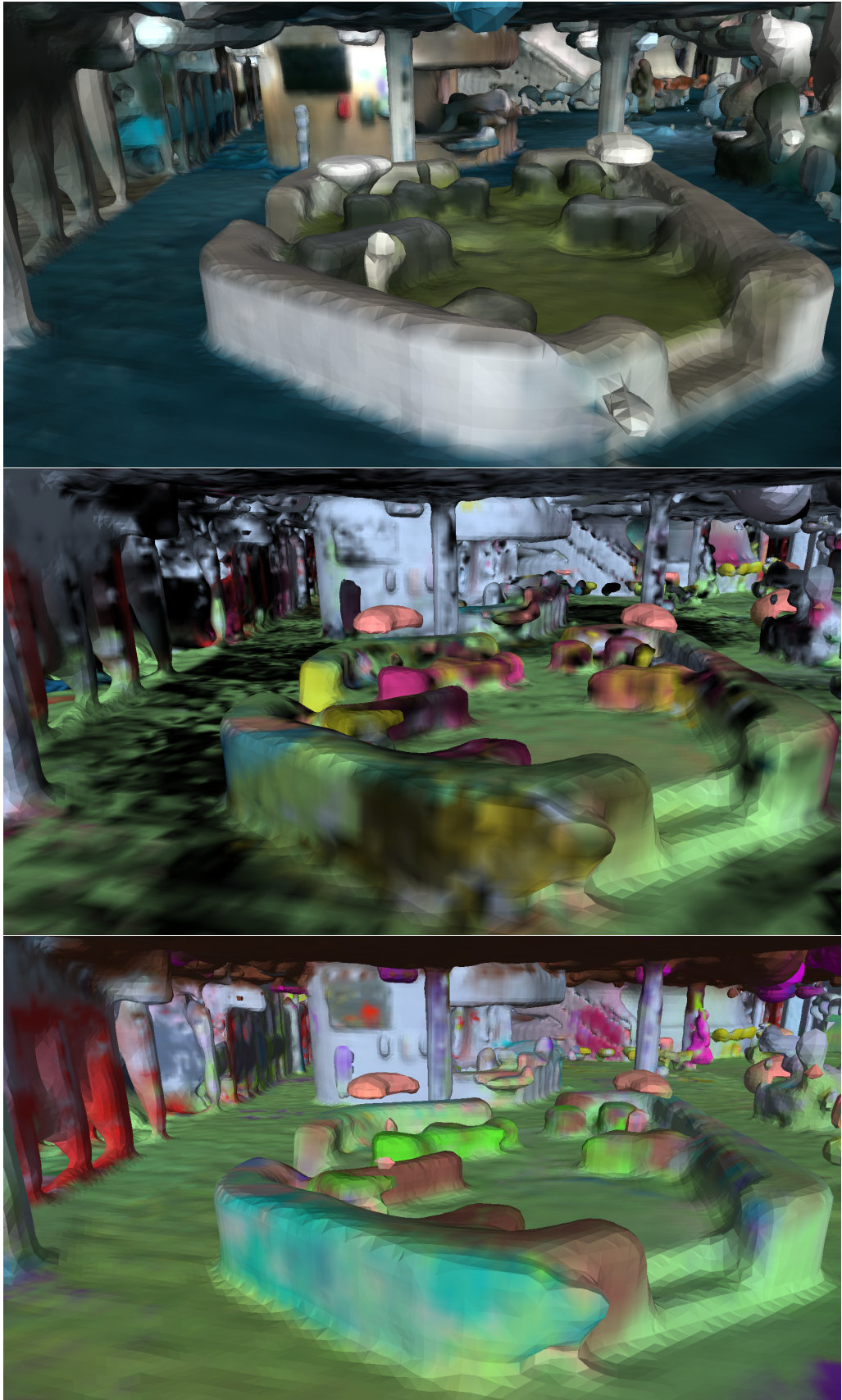
In this section, metric-semantic 3D reconstructions are generated from scratch with data collected from two campus environments with two different RGB-D cameras. The first dataset contains one floor of a campus building captured with the Realsense D455 camera, while the second is a smaller set captured with Azure Kinect in a lobby of another building. The Realsense T265 tracking camera is used in both cases for local motion estimation and RTAB-Map is applied for global positioning. The areas are segmented both with EfficientPS trained on ScanNet and HRNet trained on MSEG. Reconstruction is performed with tools included in RTAB-Map. Similar to the earlier section, only a small portion of the data can be reviewed, while more examples are provided in Appendix C.

Two-dimensional reconstructions of both environments are presented in Figure 5.7, while Figure 5.8 presents a 3D view from the first dataset reconstructed from D455 images and Figure 5.10 shows a view from the second dataset reconstructed from Kinect images. The mesh generated from images captured with D455 is clearly less detailed than the one generated from Kinect data: while individual objects are clearly recognisable and fairly detailed in the latter, in the former most objects are unrecognisable without prior knowledge of the location. The differences are also apparent in 2D: the reconstruction of the first dataset is much noisier. The reason to the large difference in quality becomes obvious when comparing depth images generated by the two sensors. Examples of both cases are provided in Figure 5.9. Kinect is the clear winner in terms of depth quality: objects boundaries are more well-defined, the image in general is less noisy and its post-processing also seems to filter errors more effectively.

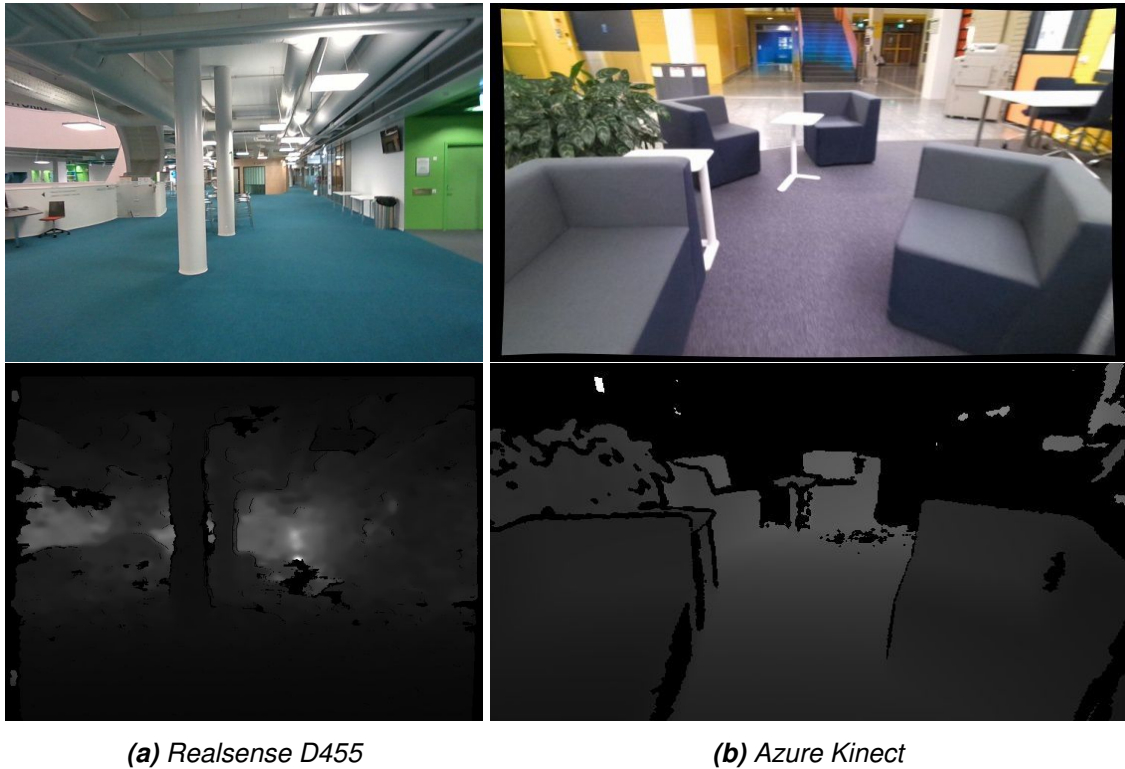


*(a) A floor of a university building reconstructed with Realsense D455 (b) An indoor scene reconstructed with Azure Kinect*

**Figure 5.7.** 2D Occupancy grids generated with the indoor reconstruction system. Black pixels represent occupied space, while grey represents traversable areas.



**Figure 5.8.** A three-dimensional view from a floor of a university building reconstructed with Realsense D455. Top: reconstruction from original RGB-D images, middle: EfficientPS segmentation, bottom: HRNet segmentation.



**Figure 5.9.** Comparison of depth image quality of cameras applied in indoor reconstruction. The further an object is from the camera, the lighter it appears. Black pixels are sections filtered out by the cameras' post-processing, e.g. because of noisy readings or reflections.

Localisation is accurate enough to generate metrically consistent reconstructions of the environment both in 2D and 3D. There are no large discontinuities in the examples, thus they seem topologically consistent in both cases. Since every keyframe is placed approximately to its corresponding real-world coordinate, global positioning has to be quite accurate. Metric quality seems to mostly be limited by the quality of depth images, although small ripples can be seen in the mesh generated from Azure Kinect images in Figure 5.10. These are most likely generated by small misalignments between images. These effects become more severe the further away from camera objects are. When collecting data with the Kinect camera, it was found necessary to limit the range of the depth sensor to reduce geometrical errors caused by errors in camera pose.

Examples of 3D segmentations generated with both models are shown in Figures 5.8 and 5.10, while some 2D segmentations are shown in Figures 5.11 and 5.12. Although EfficientPS performed quite well on the ScanNet dataset, it does less so on data outside the dataset. Segmentation with the HRNet trained on the MSEG dataset seems to generalise a lot better. This is to be expected, since MSEG has a lot more data from a larger range of environments and image sources, while images in ScanNet are focused on a smaller set of environments and all captured with the same camera model. HRNet segmentations seem to be of similar quality to ScanNet results on most occasions.





**Figure 5.10.** A lobby of of a university building reconstructed with Azure Kinect. Top: reconstruction from original RGB-D images, middle: EfficientPS segmentation, bottom: HRNet segmentation.

EfficientPS seems to mix labels quite easily, and the edges of segmentations are often spread far from actual objects. However, the reconstruction process seems to have a regularising effect on these errors: because 3D reconstructions result from many frames that often overlap, and most of the time segmentations are correct, the resulting 3D segmentation is roughly correct, although quite noisy. The outputs of semantic segmentation head alone were too noisy to extract any useful information, thus EfficientPS's results in this section are semantic labels from the panoptic output of the model, similar to Figures 5.5d and 5.6d. With some post-processing, useful information could still likely be extracted from the reconstruction. For example, it would be possible to fit planes to points labelled as floors and walls, and object labels could be averaged across a certain radius around them to reduce noise. If object instances detected by the instance segmentation head were tracked in 3D, it would also most likely reduce confusion similarly to Figure 5.6d. HRNet's segmentations are also far from perfect, although some variance in object colours is explained by the large number of classes in MSEG: for example, there are six different classes for seats, which could easily be confused with one another, while ScanNet only has two.

Reconstructions generated with either cameras could most likely be used for robot navigation, although the ones produced from Kinect images could allow for more detailed trajectory planning and navigation in tighter spaces. RTAB-Map computes local occupancy grids in real time, thus obstacle avoidance does not need to rely on the global map, which cannot be optimised online in large environments. Localisation could be utilised in augmented reality applications, which could also employ semantic labels to generate content or provide information about the environment. For example, virtual objects could be placed on tables, and the application could help the user to find certain objects. If object instances were to be tracked in 3D, they could also be replaced with more detailed 3D models for easier interaction and prettier visualisations.

Naturally, the system isn't only limited to detecting floors, walls and furniture, although with current hardware, large open spaces like factories or outdoors environments could be hard to reconstruct. If annotated data is available, the segmentation model could be trained for many different segmentation tasks. For instance, it could be used to remove errors generated from dynamic objects – like people walking past the camera – by removing points associated with them from the cloud. It could also be used to automatically label certain important static objects to a reconstruction, like fire extinguishers, exit signs or exits themselves. Real-time segmentation would extend the possible range of applications immensely: robots and AR devices could perform segmentation online, which would allow them to compare their results to the reconstruction. This would also allow for more robust localisation, as semantics could also be used in addition to spatial information.



(a) EfficientPS panoptic segmentation

(b) HRNet semantic segmentation

Figure 5.11. 2D segmentation of two views from the first dataset



(a) EfficientPS panoptic segmentation

(b) HRNet semantic segmentation

Figure 5.12. 2D segmentation of two views from the second dataset

## 6 POSSIBLE IMPROVEMENTS BASED ON RESULTS

The system seems to perform quite well already, but there is still a lot of room for improvement. The baseline system could already be used in some basic indoor applications, at least when mapping with Azure Kinect and HRNet. Some areas have already been studied in other works, while others could provide entirely new research topics. Based on the results presented earlier, possible improvements and directions to take the research further are discussed here from three perspectives. Improvements on hardware and localisation software are analysed first, after which the effects of training data on end results are assessed. Finally, possible upgrades to segmentation models are discussed.

### 6.1 Hardware and localisation

Based on results presented in Section 4, global positioning could still be made a bit more accurate by employing a more accurate loop-closure approach. In the previous chapter, it was also mentioned that pose errors limited the range at which depth could be acquired before severely affecting metric quality. From the results of Table 4.1 it seems that RTAB-Map isn't as good at tracking camera rotation as the ORB-SLAM variants, even if visual odometry is the same in both systems. Thus an upgrade to the loop closure detection or bundle adjustment could improve map quality. However, the effect of Realsense T265 on pose accuracy compared to the one in RTAB-Map is still unclear. With the current range limits in place, however, metric quality would likely be improved most with more accurate depth images. If more accurate – and therefore heavier or more expensive – devices are not available, depth completion methods [171] could also be applied to acquire more consistent depth images, which could in turn also improve localisation.

Operation in larger open spaces or outdoor environments most likely would not be possible with hardware used in this thesis. A stereo camera with a wider baseline – or a LiDAR – could be necessary for applications that require longer range. Although it is really accurate in small indoor environments, the local tracking of T265 could be less accurate in environments where visual features are further away, because of the narrow stereo baseline. With larger distances, quality would also be more affected by errors in pose estimates. However, mapping would most likely not be performed with handheld devices in these scenarios, in which case camera movement could be much more predictable.

## 6.2 Training data

Models that rely on learning their intended task from examples are only as good as the data they are trained on. [172] Sophisticated model architectures and effective training algorithms can reduce the required level of quality and amount of data, but the data is still needed regardless. [173] If training data does not represent the application domain well enough, even the most sophisticated segmentation models will fail. This fact can be observed in the results of this thesis as well: although training on ScanNet data will provide good results on that specific dataset, the model could fail to provide satisfactory results on other tasks. The MSEG dataset has a larger variety of data that has been collected from many different sources, thus models trained on it generalise to a wider variety of tasks than ones trained on ScanNet. Even though the task and environments in question are quite close to original ScanNet data, the model trained on MSEG performs better than EfficientPS trained on ScanNet by a large margin.

The authors of MSEG have announced a panoptic version of their dataset, thus EfficientPS could be trained on it when it is released. This would provide a fairer comparison between the two models, and could enable EfficientPS to generalise as well as HRNet does in this work. However, the large variety of labels in MSEG can cause problems as well: on many occasions, objects are labelled to multiple similar classes. This could be avoided by re-mapping some of the classes in the dataset based on the application. Some classes can be considered out of context for the task at hand, thus they would be best to ignore completely. For example, since the examples in this thesis is consider indoor reconstruction of campus buildings, many classes most often not found in the environment – like 'road', 'car' or 'giraffe' – could be mapped to 'void' class, and some classes similar to each other could be combined. For example, most applications would not likely need to separate different kinds of seats from one another, thus the six different seat classes in MSEG could be fused as one.

Beyond simply adding more data, the model can be fit better to a task by fine-tuning it with task-specific data. If the model is pre-trained on a relevant larger dataset, the amount of data required for fine-tuning gets lower. The method of pre-training on more general data and fine-tuning to a more specific task is called transfer learning [174], and has been proven to be a really effective way of training neural networks for specific tasks. For example – as is seen in Table 4.2 – models pre-trained with other relevant data seem to perform a lot better on Cityscapes than ones trained only on the relatively limited training set. The effect of pre-training on accuracy will, however, get smaller the larger the fine-tuning dataset is. [163] If the system designed in this thesis was to be applied to a specific task, best results would most likely be achieved by first pre-training a segmentation model with MSEG, and then fine-tuning it with a small dataset captured from the intended environment and purposefully annotated with its application in mind.

### 6.3 Segmentation models

From the results of Table 4.2 it seems that proposal-free panoptic segmentation methods could be more accurate on thing classes. A proposal free method would also potentially be more efficient since the generation of proposals is a pretty complex task. On the other hand, proposal-free panoptic fusion is computationally more demanding, although the efficiency of such algorithms seems to be getting better. [118] Panoptic fusion in proposal-free methods also does not generate bias to either things or stuff. It might be beneficial to combine ideas introduced in EfficientPS [115] with proposal-free methods like [118] and [117].

Since depth is often available in reconstruction tasks, it could be used to provide spatial context to segmentation. As seen in the results of Table 4.5, utilising depth images in segmentation increases accuracy significantly. The RGB-D segmentation methods presented in the table all adopt the popular approach of middle-fusion: two separate feature extractors are trained for RGB and depth and their features are fused within the model. This seems to be effective, but also leads to significantly larger models because of the additional feature extractor. Augmenting convolution operations in the extractor with depth data seems to provide similar results to middle-fusion methods without adding more parameters to the model [79], although the approach hasn't been tested as much and hasn't seen much use. Another interesting idea is introduced in [175], where the scene is split into regions of different scales based on range information.

In [123] it was shown that regularising the reconstructed 3D segmentation with a Conditional Random Field – results of which are also presented in Table 4.5 – could also improve segmentation quality. Although the model applied in the work by itself should perform worse than EfficientPS, the approach has a lot higher accuracy on ScanNet test data. The instance tracking method introduced in the same work would also enable the use of panoptic segmentation in 3D.

Applications that require interaction with the environment could benefit from real-time segmentation as well. Semantic information could also be useful in localisation. [59, 176, 177] The system itself would be easily modified to run all computation in real time, but since the segmentation model requires quite a lot of computation, it most likely would not meet real-time requirements in its current stage. However, a system capable of real-time panoptic segmentation of videos was introduced in [127]. Moreover, unlike [127], segmentation could only be performed to keyframes in the SLAM graph – which would reduce computational requirements if a high framerate isn't really needed. A Bayesian filter could also be applied to estimate tracks of detected objects between frames. Distributed computing is also quite easy to set up with ROS, thus segmentation could also be done on a separate machine if a fast enough wireless network connection is available.

## 7 CONCLUSION

Metric-semantic 3D reconstruction with visual data was studied in this thesis – especially in the context of indoor environments. Based on the theory and history of the research field presented in the beginning of the work, a baseline system was designed. The system was evaluated both quantitatively and qualitatively and applied in real-world scenarios. Although there are many areas that could be improved, results are already quite promising and the reconstructions produced by the system were already estimated to be accurate enough for basic applications. Four research questions were posed in the beginning of this thesis, answers to which will be summarised next. References to chapters with more details are provided.

What are the basic building blocks of a metric-semantic reconstruction system? Firstly, to acquire a map of a previously unknown environment, one needs a method for solving the simultaneous localisation and mapping problem. In turn, a solution to SLAM contains at least means for motion estimation and a loop closure method. Ways to perceive the environment, as well as ways to process the information are also essential. SLAM methods in the context of this thesis are covered in Section 2.1. There are many ways to capture semantic meaning from the environment and incorporate it into the reconstruction, but from the perspective of dense visual reconstruction, a method for processing RGB-D images to a 3D representation is required, as well as either a method for segmenting images or the reconstruction itself. This thesis focuses more on the view-based approach of scene segmentation, methods to which are described in Section 2.2.

How to choose the best options for practical applications? Regarding SLAM, the first choice is between filtering methods and approaches based on nonlinear graph optimisation. Graph-based methods are the most popular solution to full SLAM – since they scale really well – while filtering methods can be advantageous in motion estimation because of their low drift. Motion estimation in visual SLAM is usually performed with visual odometry. However, sensor fusion with proprioceptive sensors should also be applied when available to increase robustness. Feature-based VO is usually the better choice over direct methods because of its larger tolerance for changes in viewpoint and lighting. Image-to-image loop closure methods have been shown to scale really well, thus they are the most common choice. For visual scene segmentation, the view-based approach is the safer choice over map-based methods – since it is not affected by camera

misalignments as much – although with accurate enough reconstructions, a map-based approach could potentially be more accurate and efficient. There are many approaches to image segmentation, and the best choice depends on the application. If only semantic- or instance segmentation is required, a method focused on either could be more efficient, while panoptic segmentation can provide both. Both proposal-based and proposal-free methods for instance- and panoptic segmentation have their advantages, thus no clear winner can be found. However, because proposal-based methods have been studied more and applied more frequently, they could be the safer choice right now. Choosing the right training data is also crucial for learning models: one has to assess how well the training data represents the intended task, and how well a model trained on it could extrapolate in cases outside training examples. A large part of Chapter 2 is dedicated to comparing different options, some of which are evaluated in Chapter 4.

What are the essential performance bottlenecks? While there are various aspects of SLAM algorithms that still could be improved upon, the quality and applications of visual metric mapping in this work seems to be mostly limited by the hardware. However – since the choice of hardware is a trade-off between cost, weight and accuracy – it could not be exactly considered a bottleneck. With good enough sensors, the most limiting factors in SLAM would likely be graph optimisation in terms of computational cost, motion estimation for local metric consistency and loop closure for topological consistency. However, in the context of this work, the segmentation model is clearly the part limiting practical applications the most. There is a lot to be improved upon regarding segmentation accuracy, and because of intense computational requirements, end-to-end real-time operation is also quite hard to achieve. The systems performance is evaluated in Chapters 4 and 5.

To what possible directions could one take this research in the future? In general, the most obvious research subjects are the ones improving upon existing work. Some improvements and changes to the system are discussed in Chapter 6. Ways to achieve more synergy between individual parts could also be researched: topology and geometry could improve semantics, and on the other hand, effective utilisation of semantic information could result in both topologically and metrically more accurate maps and more robust localisation. Apart from technology demonstrations, there haven't been many practical applications of machines using metric-semantic reconstructions, but – owing to huge interest and fast advances in the field – we might also get to see more practical applications in the near future. However, there are still many problems to solve – like safety considerations and operation in dynamic environments – before the techniques could be applied to real use-cases in areas like industry [178], autonomous driving [179, 180] or robot navigation [181, 182]. Many of these problems are also related to the black-box nature of deep neural networks and how to render them more robust considering real-world applications. [183] With advances in relevant research fields and emergence of new technologies, entirely new kinds of applications and research subjects could arise in the future as well.



## REFERENCES

- [1] Chatila, R. Robot Mapping: An Introduction. *Robotics and Cognitive Approaches to Spatial Mapping*. Springer Berlin Heidelberg, Mar. 2008, 9–12.
- [2] Jefferies, M. E. and Yeap, W.-K. *Robotics and Cognitive Approaches to Spatial Mapping*. eng. Vol. 38. Springer tracts in advanced robotics. Berlin/Heidelberg: Springer Berlin / Heidelberg, 2008.
- [3] Thrun, S. Simultaneous localization and mapping. *Robotics and Cognitive Approaches to Spatial Mapping*. Springer Berlin Heidelberg, Mar. 2008, 13–41.
- [4] Durrant-Whyte, H. and Bailey, T. Simultaneous localization and mapping: part I. Vol. 13. 2. Oct. 2006, 99–110.
- [5] Bailey, T. and Durrant-Whyte, H. Simultaneous localization and mapping (SLAM): Part II. Vol. 13. Oct. 2006, 108–117.
- [6] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. and Leonard, J. J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. Vol. 32. 6. 2016, 1309–1332.
- [7] Yousif, K., Bab-Hadiashar, A. and Hoseinnezhad, R. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. Vol. 1. Nov. 2015.
- [8] Schönberger, J. L. and Frahm, J.-M. Structure-from-Motion Revisited. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, 4104–4113.
- [9] Özyesil, O., Voroninski, V., Basri, R. and Singer, A. A survey of structure from motion. *Acta Numerica* 26 (2017), 305–364.
- [10] Russell, S. J. and Norvig, P. *Artificial intelligence : a modern approach*. 3rd ed. Upper Saddle River (NJ): Prentice Hall, 2009.
- [11] Gallistel, C. R. Dead Reckoning, Cognitive Maps, Animal Navigation and the Representation of Space: An Introduction. *Robotics and Cognitive Approaches to Spatial Mapping*. Springer Berlin Heidelberg, Mar. 2008, 137–143.
- [12] Thrun, S., Burgard, W. and Fox, D. *Probabilistic robotics*. Cambridge, Mass: MIT Press, 2005.
- [13] Särkkä, S. *Bayesian Filtering and Smoothing*. eng. Vol. 3. Institute of Mathematical Statistics Textbooks. West Nyack: Cambridge University Press, 2013.
- [14] Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. Vol. 60. 2. Hingham, MA, USA, Nov. 2004, 91–110.
- [15] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. Speeded-Up Robust Features (SURF). Vol. 110. 3. June 2008, 346–359.

- [16] Labbé, M. and Michaud, F. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. Vol. 36. 2. 2019, 416–446.
- [17] Mur-Artal, R. and Tardós, J. D. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. Vol. 33. 5. 2017, 1255–1262.
- [18] Hess, W., Kohler, D., Rapp, H. and Andor, D. Real-Time Loop Closure in 2D LIDAR SLAM. *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, 1271–1278.
- [19] Scaramuzza, D. and Fraundorfer, F. Visual Odometry [Tutorial]. Vol. 18. Dec. 2011, 80–92.
- [20] Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P. and Milford, M. J. Visual Place Recognition: A Survey. Vol. 32. 1. 2016, 1–19.
- [21] Leonard, J. J. and Durrant-Whyte, H. F. Mobile robot localization by tracking geometric beacons. Vol. 7. 3. 1991, 376–382.
- [22] Moutarlier, P. and Chatila, R. An Experimental System for Incremental Environment Modelling by an Autonomous Mobile Robot. *1st International Symposium on Experimental Robotics*. 1989.
- [23] Williams, S. B., Dissanayake, G. and Durrant-Whyte, H. An efficient approach to the simultaneous localisation and mapping problem. *IEEE International Conference on Robotics and Automation*. Vol. 1. 2002, 406–411 vol.1.
- [24] Tardos, J., Neira, J., Newman, P. and Leonard, J. Robust Mapping and Localization in Indoor Environments Using Sonar Data. Vol. 21. Apr. 2002.
- [25] Doucet, A., Freitas, N. d., Murphy, K. P. and Russell, S. J. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. *16th Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA, 2000, 176–183.
- [26] Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. Nov. 2002.
- [27] Grisetti, G., Kümmerle, R., Stachniss, C. and Burgard, W. A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), 31–43.
- [28] Triggs, B., Mclauchlan, P., Hartley, R. and Fitzgibbon, A. Bundle adjustment - A modern synthesis. Jan. 2000, 198–372.
- [29] Lourakis, M. and Argyros, A. SBA: A Software Package for Generic Sparse Bundle Adjustment. Vol. 36. Jan. 2009.
- [30] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W. G2o: A general framework for graph optimization. *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, 3607–3613.
- [31] Grisetti, G., Stachniss, C. and Burgard, W. Nonlinear Constraint Network Optimization for Efficient Map Learning. Vol. 10. 3. 2009, 428–439.

- [32] Endres, F., Hess, J., Sturm, J., Cremers, D. and Burgard, W. 3-D mapping with an RGB-D camera. Vol. 30. Feb. 2014, 177–187.
- [33] Moravec, H. Obstacle avoidance and navigation in the real world by a seeing robot rover. PhD dissertation. Stanford University, Stanford, CA, 1980.
- [34] Davison, Andrew J. Real-time simultaneous localisation and mapping with a single camera. *Proceedings Ninth IEEE International Conference on Computer Vision*. Vol. 2. 2003, 1403–1410.
- [35] Mur-Artal, R., Montiel, J. M. M. and Tardós, J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. Vol. 31. 5. 2015, 1147–1163.
- [36] Strasdat, H., Montiel, J. and Davison, A. Real-time monocular SLAM: Why filter?: May 2010, 2657–2664.
- [37] Klein, G. and Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. Dec. 2007, 225–234.
- [38] Nister, D., Naroditsky, O. and Bergen, J. Visual odometry. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2004.
- [39] Fraundorfer, F., Scaramuzza, D. and Pollefeys, M. A constricted bundle adjustment parameterization for relative scale estimation in visual odometry. *2010 IEEE International Conference on Robotics and Automation*. 2010, 1899–1904.
- [40] Sünderhauf, N., Konolige, K., Lacroix, S. and Protzel, P. Visual Odometry Using Sparse Bundle Adjustment on an Autonomous Outdoor Vehicle. Jan. 2005, 157–163.
- [41] Konolige, K. and Agrawal, M. Large-Scale Visual Odometry for Rough Terrain. Vol. 66. Jan. 2011.
- [42] Qin, T., Li, P. and Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. Vol. 34. 4. 2018, 1004–1020.
- [43] Tardif, J. P., George, M., Laverne, M., Kelly, A. and Stentz, A. A New Approach to Vision-Aided Inertial Navigation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2010, 4161–4168.
- [44] Ouerghi, S., Ragot, N., Boutteau, R. and Savatier, X. Comparative Study of a Commercial Tracking Camera and ORB-SLAM2 for Person Localization. Jan. 2020, 357–364.
- [45] Alapetite, A., Wang, Z., Hansen, J., Zajączkowski, M. and Patalan, M. Comparison of Three Off-the-Shelf Visual Odometry Systems. Vol. 9. July 2020, 56.
- [46] Engel, J., Sturm, J. and Cremers, D. Semi-Dense Visual Odometry for a Monocular Camera. *IEEE/CVF International Conference on Computer Vision (ICCV)*. Sydney, Australia, Dec. 2013.
- [47] Newcombe, R. A., Lovegrove, S. J. and Davison, A. J. DTAM: Dense tracking and mapping in real-time. *2011 International Conference on Computer Vision*. 2011, 2320–2327.

- [48] Rublee, E., Rabaud, V., Konolige, K. and Bradski, G. ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision*. 2011, 2564–2571.
- [49] Torr, P. and Zisserman, A. Feature Based Methods for Structure and Motion Estimation. *Workshop on Vision Algorithms*. 1999.
- [50] Irani, M. and Anandan, P. About Direct Methods. Jan. 1999, 267–277.
- [51] Mur-Artal, R. and Tardós, J. D. Fast relocalisation and loop closing in keyframe-based SLAM. *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, 846–853.
- [52] Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I. and Tardós, J. A comparison of loop closing techniques in monocular SLAM. Vol. 57. 12. Inside Data Association. 2009, 1188–1197.
- [53] Nister, D. and Stewenius, H. Scalable Recognition with a Vocabulary Tree. *IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 2006, 2161–2168.
- [54] Fraundorfer, F. and Scaramuzza, D. Visual Odometry: Part II: Matching, Robustness, Optimization, and Applications. Vol. 19. 2. 2012, 78–90.
- [55] Fischler, M. A. and Bolles, R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Vol. 24. 6. New York, NY, USA, June 1981, 381–395.
- [56] Wang, F. and Zhao, Z. A survey of iterative closest point algorithm. *2017 Chinese Automation Congress (CAC)*. 2017, 4395–4399.
- [57] Landgraf, Z., Falck, F., Bloesch, M., Leutenegger, S. and Davison, A. Comparing View-Based and Map-Based Semantic Labelling in Real-Time SLAM. *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.
- [58] Sünderhauf, N., Pham, T. T., Latif, Y., Milford, M. and Reid, I. Meaningful maps with object-oriented semantic mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, 5079–5085.
- [59] Chancán, M. and Milford, M. DeepSeqSLAM: A Trainable CNN+RNN for Joint Global Description and Sequence-based Place Recognition. 2020.
- [60] Li, F., Chen, W., Xu, W., Huang, L., Li, D., Cai, S., Yang, M., Xiong, X., Liu, Y. and Li, W. A Mobile Robot Visual SLAM System With Enhanced Semantics Segmentation. Vol. 8. 2020, 25442–25458.
- [61] Qi, C. R., Yi, L., Su, H. and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*. 2017, 5099–5108.
- [62] Pham, Q.-H., Nguyen, D. T., Hua, B.-S., Roig, G. and Yeung, S.-K. JSIS3D: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

- [63] Otsu, N. A Threshold Selection Method from Gray-Level Histograms. Vol. 9. 1. 1979, 62–66.
- [64] Nameirakpam, D., Singh, K. and Chanu, Y. Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm. Vol. 54. Dec. 2015, 764–771.
- [65] Plath, N., Toussaint, M. and Nakajima, S. Multi-class image segmentation using conditional random fields and global classification. *ICML*. 2009, 817–824.
- [66] Minaee, S., Boykov, Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N. and Terzopoulos, D. Image Segmentation Using Deep Learning: A Survey. 2020.
- [67] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*. Vol. 36. 1980, 193–202.
- [68] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. and Jackel, L. Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1990, 396–404.
- [69] Zhou, V. *Machine Learning for Beginners: An Introduction to Neural Networks*. <https://victorzhou.com/blog/intro-to-neural-networks/>. Accessed: 2021-02-12. 2019.
- [70] Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. MIT Press, 2016.
- [71] Chatterjee, H. S. *A Basic Introduction to Convolutional Neural Network*. <https://victorzhou.com/blog/intro-to-neural-networks/>. Accessed: 2021-02-12. 2019.
- [72] Kirillov, A., He, K., Girshick, R., Rother, C. and Dollar, P. Panoptic Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.
- [73] Caruana, R. Multitask Learning. Vol. 28. 1997, 41–75.
- [74] Blair, D. C. *Information Retrieval*. 2nd ed. Vol. 30. 6. 1979, 374–375.
- [75] Yang, T.-J., Collins, M. D., Zhu, Y., Hwang, J.-J., Liu, T., Zhang, X., Sze, V., Papan-dreou, G. and Chen, L.-C. *DeeperLab: Single-Shot Image Parser*. 2019.
- [76] Vandenhende, S., Georgoulis, S., Gansbeke, W. V., Proesmans, M., Dai, D. and Gool, L. Multi-Task Learning for Dense Prediction Tasks: A Survey. 2020.
- [77] Jiao, J., Wei, Y., Jie, Z., Shi, H., Lau, R. and Huang, T. S. Geometry-Aware Distillation for Indoor Semantic Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, 2864–2873.
- [78] Zhang, Y. and Funkhouser, T. Deep Depth Completion of a Single RGB-D Image. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.
- [79] Wang, W. and Neumann, U. Depth-aware CNN for RGB-D Segmentation. *European Conference on Computer Vision (ECCV)*. Sept. 2018.

- [80] Lee, S., Park, S. and Hong, K. RDFNet: RGB-D Multi-level Residual Feature Fusion for Indoor Semantic Segmentation. *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, 4990–4999.
- [81] Lin, D., Chen, G., Cohen-Or, D., Heng, P. and Huang, H. Cascaded Feature Network for Semantic Segmentation of RGB-D Images. *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, 1320–1328.
- [82] COCO Consortium. *COCO panoptic segmentation challenge*. <https://competitions.codalab.org/competitions/19507>. 2018.
- [83] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. and Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. <https://cityscapes-dataset.com/benchmarks/#panoptic-scene-labeling-task>. 2016.
- [84] COCO Consortium. *COCO object detection challenge*. <https://competitions.codalab.org/competitions/20796>. 2018.
- [85] Long, J., Shelhamer, E. and Darrell, T. Fully convolutional networks for semantic segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, 3431–3440.
- [86] Krizhevsky, A., Sutskever, I. and Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. Vol. 25. Jan. 2012.
- [87] Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*. 2015.
- [88] Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. Going deeper with convolutions. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, 1–9.
- [89] Zhao, H., Shi, J., Qi, X., Wang, X. and Jia, J. Pyramid Scene Parsing Network. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 6230–6239.
- [90] Noh, H., Hong, S. and Han, B. Learning Deconvolution Network for Semantic Segmentation. *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2015, 1520–1528.
- [91] Badrinarayanan, V., Kendall, A. and Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. Vol. 39. 12. 2017, 2481–2495.
- [92] Ronneberger, O., Fischer, P. and Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Oct. 2015, 234–241.
- [93] Milletari, F., Navab, N. and Ahmadi, S. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *2016 Fourth International Conference on 3D Vision (3DV)*. 2016, 565–571.

- [94] Adelson, E., Anderson, C., Bergen, J., Burt, P. and Ogden, J. Pyramid Methods in Image Processing. Vol. 29. Nov. 1983.
- [95] Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. Feature Pyramid Networks for Object Detection. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 936–944.
- [96] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W. and Xiao, B. Deep High-Resolution Representation Learning for Visual Recognition. 2019.
- [97] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A. L. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *International Conference on Learning Representations (ICLR)*. 2015.
- [98] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. June 2016.
- [99] Chen, L.-C., Papandreou, G., Schroff, F. and Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. 2017.
- [100] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *European Conference on Computer Vision (ECCV)*. 2018.
- [101] Chen, L.-C., Collins, M., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H. and Shlens, J. Searching for Efficient Multi-Scale Architectures for Dense Image Prediction. *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018, 8699–8710.
- [102] Valada, A., Mohan, R. and Burgard, W. Self-Supervised Model Adaptation for Multimodal Semantic Segmentation. Vol. 128. May 2020.
- [103] Lafferty, J., McCallum, A. and Pereira, F. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Jan. 2001, 282–289.
- [104] Girshick, R., Donahue, J., Darrell, T. and Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, 580–587.
- [105] Girshick, R. Fast R-CNN. *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2015, 1440–1448.
- [106] Ren, S., He, K., Girshick, R. and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett. Vol. 28. Curran Associates, Inc., 2015, 91–99.
- [107] He, K., Gkioxari, G., Dollár, P. and Girshick, R. Mask R-CNN. *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, 2980–2988.

- [108] He, K., Zhang, X., Ren, S. and Sun, J. Deep Residual Learning for Image Recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, 770–778.
- [109] Liu, S., Qi, L., Qin, H., Shi, J. and Jia, J. Path Aggregation Network for Instance Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, 8759–8768.
- [110] Chen, L.-C., Hermans, A., Papandreou, G., Schroff, F., Wang, P. and Adam, H. MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features. June 2018, 4013–4022.
- [111] Chen, X., Girshick, R. B., He, K. and Dollár, P. TensorMask: A Foundation for Dense Object Segmentation. 2019, 2061–2069.
- [112] Kirillov, A., Girshick, R., He, K. and Dollár, P. Panoptic Feature Pyramid Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, 6392–6401.
- [113] Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E. and Urtasun, R. UPSNet: A Unified Panoptic Segmentation Network. 2019, 8810–8818.
- [114] Porzi, L., Bulò, S. R., Colovic, A. and Kotschieder, P. Seamless Scene Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.
- [115] Mohan, R. and Valada, A. EfficientPS: Efficient Panoptic Segmentation. 2020. URL: <https://github.com/DeepSceneSeg/EfficientPS>.
- [116] Li, J., Raventos, A., Bhargava, A., Tagawa, T. and Gaidon, A. Learning to Fuse Things and Stuff. 2018.
- [117] Cheng, B., Collins, M. D., Zhu, Y., Liu, T., Huang, T. S., Adam, H. and Chen, L. L.-C. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, 12472–12482.
- [118] Gao, N., Shan, Y., Wang, Y., Zhao, X., Yu, Y., Yang, M. and Huang, K. SSAP: Single-Shot Instance Segmentation With Affinity Pyramid. 2019, 642–651.
- [119] Tan, M. and Le, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *Proceedings of the International Conference on Machine Learning (ICML)*. 2019.
- [120] Yonghuai, L., Rosin, P. L. and Huber, P. *3D imaging, analysis and applications*. 1st ed. London: Springer, 2012.
- [121] Chen, C., Wang, B., Lu, C. X., Trigoni, A. and Markham, A. A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence. 2020.
- [122] Weiss, S., Achtelek, M. W., Kneip, L., Scaramuzza, D. and Siegwart, R. Intuitive 3D maps for MAV terrain exploration and obstacle avoidance. Vol. 61. 2011, 473–493.



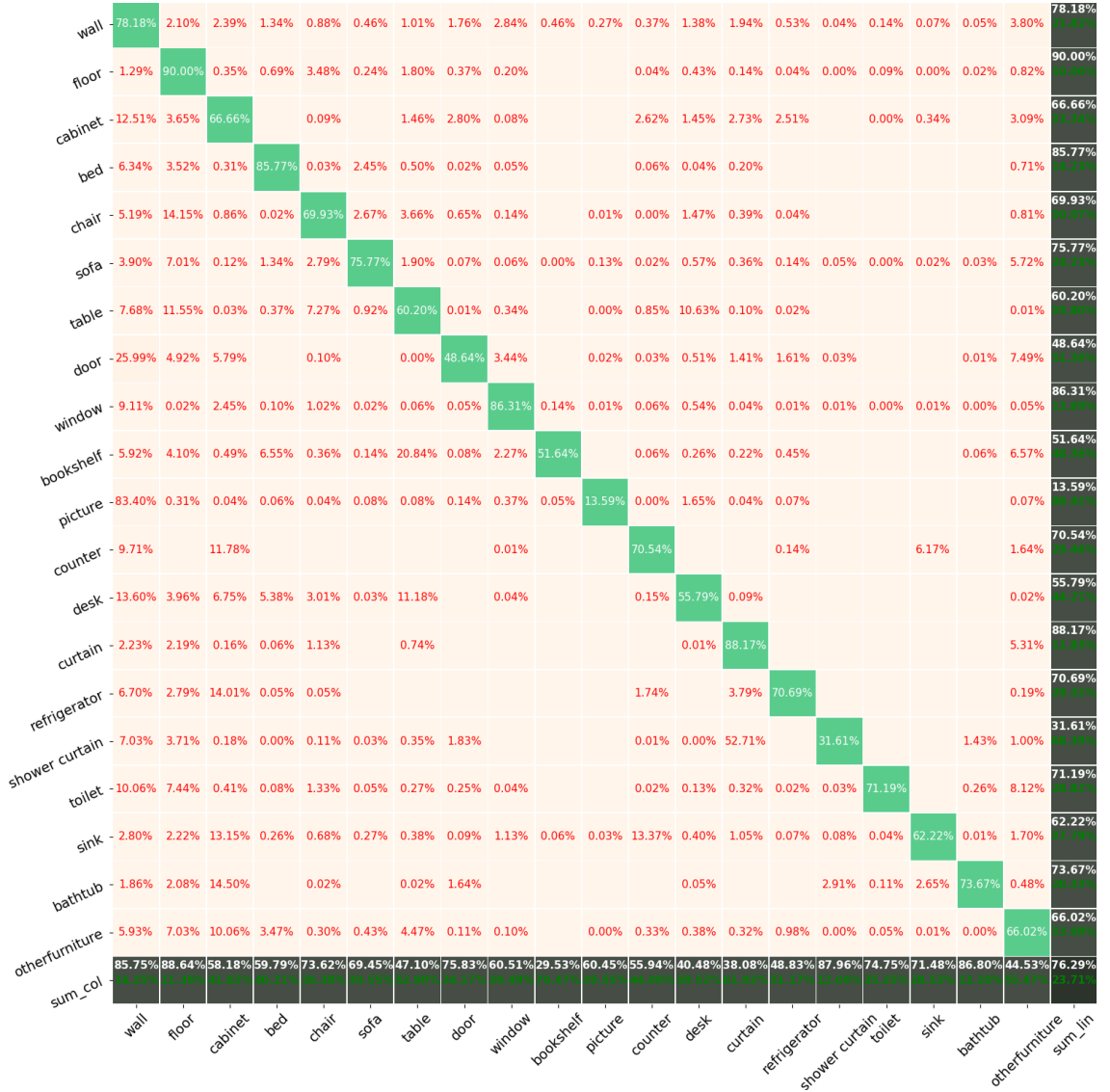
- [123] Narita, G., Seno, T., Ishikawa, T. and Kaji, Y. PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things. 2019, 4205–4212.
- [124] Vanneste, S., Bellekens, B. and Weyn, M. 3DVFH+: Real-Time Three-Dimensional Obstacle Avoidance Using an Octomap. *International Conference on Mechatronics, Robotics, and System Engineering*. 2014.
- [125] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C. and Burgard, W. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. Vol. 34. 2013, 189–206.
- [126] Liu, Y., Shen, C., Yu, C. and Wang, J. Efficient Semantic Video Segmentation with Per-frame Inference. 2020.
- [127] Hou, R., Li, J., Bhargava, A., Raventos, A., Guizilini, V., Fang, C., Lynch, J. and Gaidon, A. Real-Time Panoptic Segmentation From Dense Detections. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [128] Kim, D., Woo, S., Lee, J.-Y. and Kweon, I. S. Video Panoptic Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [129] Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R. and Nieto, J. Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, 1366–1373.
- [130] Curless, B. and Levoy, M. A volumetric method for building complex models from range images. 1996.
- [131] Labbé, M. and Michaud, F. Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation. Vol. 29. 3. 2013, 734–745.
- [132] Bradski, G. The OpenCV Library. 2000. URL: <https://opencv.org/>.
- [133] Labbé, M. and Michaud, F. Long-term online multi-session graph-based SPLAM with memory management. Vol. 42. 2018, 1133–1150.
- [134] Muja, M. and Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. *International Conference on Computer Vision Theory and Applications (VISAPP)*. 2009, 331–340.
- [135] Angeli, A., Filliat, D., Doncieux, S. and Meyer, J. Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words. Vol. 24. 5. 2008, 1027–1037.
- [136] Georgia Tech BORG Lab. *GTSAM Library*. URL: <https://gtsam.org/>.
- [137] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. and Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [138] Neuhold, G., Ollmann, T., Rota Bulò, S. and Kotschieder, P. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. *IEEE/CVF International*

- Conference on Computer Vision (ICCV)*. 2017. URL: <https://www.mapillary.com/dataset/vistas>.
- [139] Varma, G., Subramanian, A., Namboodiri, A., Chandraker, M. and Jawahar, C. V. IDD: A Dataset for Exploring Problems of Autonomous Navigation in Unconstrained Environments. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, 1743–1751.
- [140] Geiger, A., Lenz, P. and Urtasun, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.
- [141] Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 1800–1807.
- [142] Nair, V. and Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning (ICML)*. ICML'10. Haifa, Israel: Omnipress, 2010, 807–814.
- [143] Khalid, M., Baber, J., Kasi, M. K., Bakhtyar, M., Devi, V. and Sheikh, N. Empirical Evaluation of Activation Functions in Deep Convolution Neural Network for Facial Expression Recognition. *International Conference on Telecommunications and Signal Processing (TSP)*. 2020, 204–207.
- [144] Rota Bulò, S., Porzi, L. and Kotschieder, P. In-place Activated BatchNorm for Memory-Optimized Training of DNNs. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018, 5639–5647.
- [145] Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K. Aggregated Residual Transformations for Deep Neural Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 5987–5995.
- [146] Bulò, S. R., Neuhold, G. and Kotschieder, P. Loss Max-Pooling for Semantic Image Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 7082–7091.
- [147] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. and Nießner, M. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [148] *Intel Realsense D435i*. <https://www.intelrealsense.com/depth-camera-d435i/>. Accessed: 2020-10-15.
- [149] *Microsoft Azure Kinect DK*. <https://azure.microsoft.com/en-us/services/kinect-dk/>. Accessed: 2020-10-15.
- [150] *Intel Realsense D455*. <https://www.intelrealsense.com/depth-camera-d455/>. Accessed: 2020-02-01.
- [151] *Intel Realsense T265*. <https://www.intelrealsense.com/tracking-camera-t265/>. Accessed: 2020-10-15.

- [152] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. and Ng, A. ROS: an open-source Robot Operating System. *IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. Kobe, Japan, May 2009. URL: <https://www.ros.org>.
- [153] Merkel, D. Docker: lightweight linux containers for consistent development and deployment. Vol. 2014. 239. 2014, 2.
- [154] *Zed Mini Mixed reality Camera*. <https://www.stereolabs.com/zed-mini/>. Accessed: 2020-10-15.
- [155] *Vicon virtual production*. <https://www.vicon.com/>. Accessed: 2020-10-15.
- [156] *OptiTrack motion capture systems*. <https://optitrack.com/>. Accessed: 2020-10-15.
- [157] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. and Siegwart, R. The EuRoC micro aerial vehicle datasets. Vol. 35. Jan. 2016.
- [158] Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M. M. and Tardós, J. D. *ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM*. 2020. arXiv: 2007.11898 [cs.R0].
- [159] Rosinol, A., Abate, M., Chang, Y. and Carlone, L. Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping. *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.
- [160] Zhang, Z. and Scaramuzza, D. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, 7244–7251.
- [161] Chen, L.-C., Wang, H. and Qiao, S. Scaling Wide Residual Networks for Panoptic Segmentation. Vol. abs/2011.11675. 2020.
- [162] Wu, Z., Shen, C. and van den Hengel, A. Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. Vol. 90. 2019, 119–133.
- [163] He, K., Girshick, R. B. and Dollár, P. Rethinking ImageNet Pre-Training. 2019, 4917–4926.
- [164] Bernhardsson, E. *Annoy: Approximate Nearest Neighbors in C++/Python*. 2018. URL: <https://pypi.org/project/annoy/>.
- [165] Lambert, J., Liu, Z., Sener, O., Hays, J. and Koltun, V. MSeg: A Composite Dataset for Multi-domain Semantic Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [166] Deng, L., Yang, M., Li, T., He, Y. and Wang, C. *RFBNet: Deep Multimodal Networks with Residual Fusion Blocks for RGB-D Semantic Segmentation*. 2019. arXiv: 1907.00135 [cs.CV].
- [167] Valada, A., Mohan, R. and Burgard, W. Self-Supervised Model Adaptation for Multimodal Semantic Segmentation. Special Issue: Deep Learning for Robotic Vision. July 2019.

- [168] Shi, W., Zhu, D., Zhang, G., Chen, L., Wang, L., Li, J. and Zhang, X. Multilevel Cross-Aware RGBD Semantic Segmentation of Indoor Environments. *IEEE International Conference on Cyborg and Bionic Systems (CBS)*. 2019, 346–351.
- [169] Han, L., Zheng, T., Xu, L. and Fang, L. OccuSeg: Occupancy-Aware 3D Instance Segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, 2937–2946.
- [170] Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M. and Guibas, L. J. Volumetric and Multi-view CNNs for Object Classification on 3D Data. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, 5648–5656.
- [171] Zhang, Y. and Funkhouser, T. Deep Depth Completion of a Single RGB-D Image. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.
- [172] Torralba, A. and Efros, A. A. Unbiased look at dataset bias. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, 1521–1528.
- [173] Yuille, A. and Liu, C. Deep Nets: What have They Ever Done for Vision?: vol. 129. Mar. 2021, 1–22.
- [174] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. and He, Q. A Comprehensive Survey on Transfer Learning. Vol. 109. 2021, 43–76.
- [175] Lin, D., Chen, G., Cohen-Or, D., Heng, P. and Huang, H. Cascaded Feature Network for Semantic Segmentation of RGB-D Images. *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, 1320–1328.
- [176] Nicholson, L., Milford, M. and Sünderhauf, N. QuadricSLAM: Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM. Vol. 4. 1. 2019, 1–8.
- [177] Yu, C., Liu, Z., Liu, X., Xie, F., Yang, Y., Wei, Q. and Qiao, F. DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments. 2018, 1168–1174.
- [178] de Souza Cardoso, L. F., Mariano, F. C. M. Q. and Zorzal, E. R. A survey of industrial augmented reality. Vol. 139. 2020, 106159.
- [179] Ozguner, U., Stiller, C. and Redmill, K. Systems for Safety and Autonomous Behavior in Cars: The DARPA Grand Challenge Experience. Vol. 95. 2. 2007, 397–412.
- [180] Campbell, M., Egerstedt, M., How, J. and Murray, R. Autonomous driving in urban environments: approaches, lessons and challenges. Vol. 368. 2010, 4649–4672.
- [181] Crespo, J., Castillo, J. C., Mozos, O. M. and Barber, R. Semantic Information for Robot Navigation: A Survey. Vol. 10. 2. 2020.
- [182] Grimmert, H., Buerki, M., Paz, L., Pinies, P., Furgale, P., Posner, I. and Newman, P. Integrating metric and semantic maps for vision-only automated parking. *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, 2159–2166.
- [183] Zhang, X., Liu, C. and Suen, C. Y. Towards Robust Pattern Recognition: A Review. Vol. 108. 6. 2020, 894–922.

## A A CONFUSION MATRIX OF SCANNET RESULTS



**Figure A.1.** Confusion matrix of 3D evaluation on the ScanNet validation set. Rows represent class predictions, while columns present the ground-truth labels of mesh vertices. True positives are represented by light green cells, while false positives for a class are represented by red cells in its corresponding row. False negatives are found from the red cells of the corresponding column. An empty cell means there are no predictions matching the corresponding case. For instance, 10.63% of vertices labelled as 'table' belonged actually to a 'desk', 85.75% of ground-truth 'wall' vertices were labelled correctly, and 10.00% of predicted 'floor' vertices actually belonged to the 'wall' class.

## B MSEG DATASET CLASSES

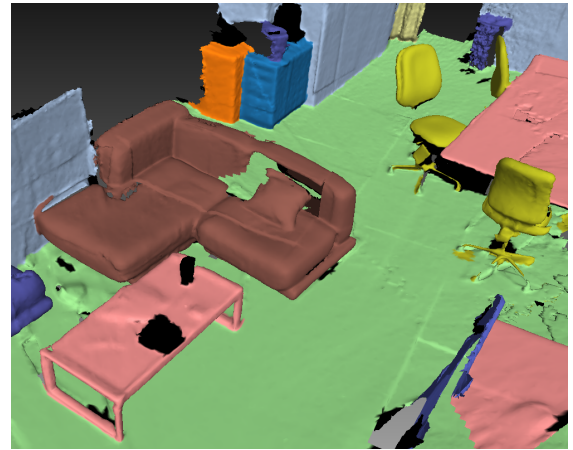
0 : backpack	49 : broccoli	98 : road	147 : mountain_hill
1 : umbrella	50 : carrot	99 : snow	148 : rock
2 : bag	51 : hot_dog	100 : sidewalk_pavement	149 : frisbee
3 : tie	52 : pizza	101 : runway	150 : skis
4 : suitcase	53 : donut	102 : terrain	151 : snowboard
5 : case	54 : cake	103 : book	152 : sports_ball
6 : bird	55 : fruit_other	104 : box	153 : kite
7 : cat	56 : food_other	105 : clock	154 : baseball_bat
8 : dog	57 : chair_other	106 : vase	155 : baseball_glove
9 : horse	58 : armchair	107 : scissors	156 : skateboard
10 : sheep	59 : swivel_chair	108 : plaything_other	157 : surfboard
11 : cow	60 : stool	109 : teddy_bear	158 : tennis_racket
12 : elephant	61 : seat	110 : hair_dryer	159 : net
13 : bear	62 : couch	111 : toothbrush	160 : base
14 : zebra	63 : trash_can	112 : painting	161 : sculpture
15 : giraffe	64 : potted_plant	113 : poster	162 : column
16 : animal_other	65 : nightstand	114 : bulletin_board	163 : fountain
17 : microwave	66 : bed	115 : bottle	164 : awning
18 : radiator	67 : table	116 : cup	165 : apparel
19 : oven	68 : pool_table	117 : wine_glass	166 : banner
20 : toaster	69 : barrel	118 : knife	167 : flag
21 : storage_tank	70 : desk	119 : fork	168 : blanket
22 : conveyor_belt	71 : ottoman	120 : spoon	169 : curtain_other
23 : sink	72 : wardrobe	121 : bowl	170 : shower_curtain
24 : refrigerator	73 : crib	122 : tray	171 : pillow
25 : washer_dryer	74 : basket	123 : range_hood	172 : towel
26 : fan	75 : chest_of_drawers	124 : plate	173 : rug_floorpat
27 : dishwasher	76 : bookshelf	125 : person	174 : vegetation
28 : toilet	77 : counter_other	126 : rider_other	175 : bicycle
29 : bathtub	78 : bathroom_counter	127 : bicyclist	176 : car
30 : shower	79 : kitchen_island	128 : motorcyclist	177 : autorickshaw
31 : tunnel	80 : door	129 : paper	178 : motorcycle
32 : bridge	81 : light_other	130 : streetlight	179 : airplane
33 : pier_wharf	82 : lamp	131 : road_barrier	180 : bus
34 : tent	83 : sconce	132 : mailbox	181 : train
35 : building	84 : chandelier	133 : cctv_camera	182 : truck
36 : ceiling	85 : mirror	134 : junction_box	183 : trailer
37 : laptop	86 : whiteboard	135 : traffic_sign	184 : boat_ship
38 : keyboard	87 : shelf	136 : traffic_light	185 : slow_wheeled_object
39 : mouse	88 : stairs	137 : fire_hydrant	186 : river_lake
40 : remote	89 : escalator	138 : parking_meter	187 : sea
41 : cell phone	90 : cabinet	139 : bench	188 : water_other
42 : television	91 : fireplace	140 : bike_rack	189 : swimming_pool
43 : floor	92 : stove	141 : billboard	190 : waterfall
44 : stage	93 : arcade_machine	142 : sky	191 : wall
45 : banana	94 : gravel	143 : pole	192 : window
46 : apple	95 : platform	144 : fence	193 : window_blind
47 : sandwich	96 : playingfield	145 : railing_banister	255 : void
48 : orange	97 : railroad	146 : guard_rail	

**Figure B.1.** Colours associated with the MSEG dataset in this work. Void is defined as anything not belonging to other classes.

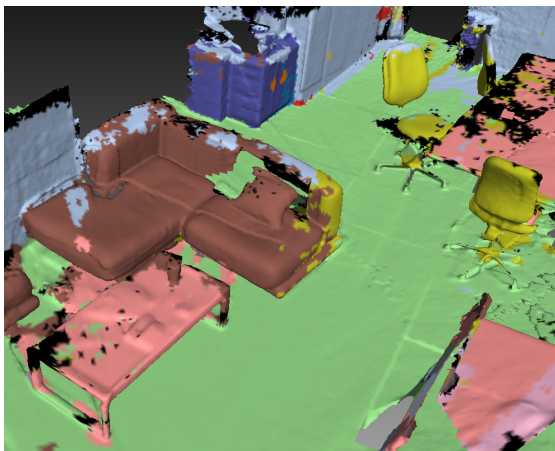
## C MORE IMAGES FROM PRACTICAL APPLICATIONS



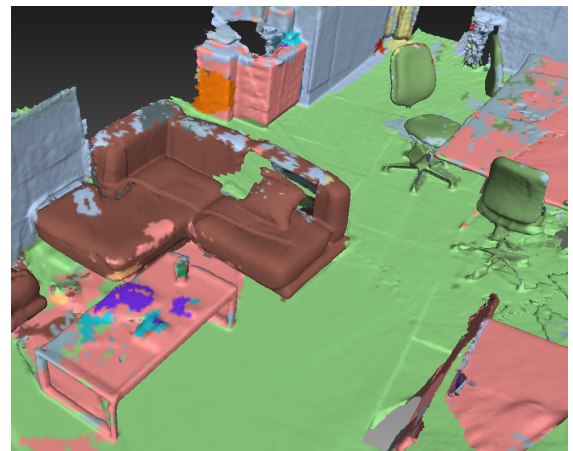
(a) Original 3D reconstruction



(b) Ground truth labels

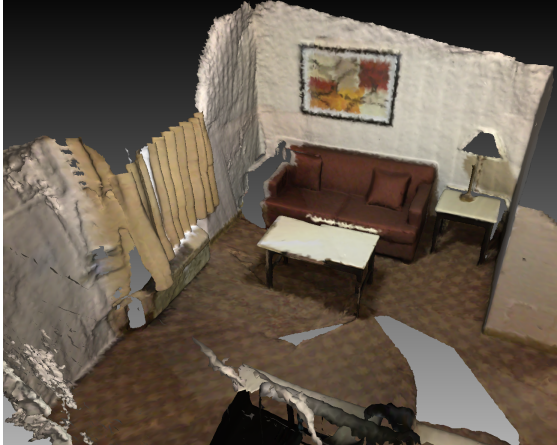


(c) Semantic segmentation with EfficienPS

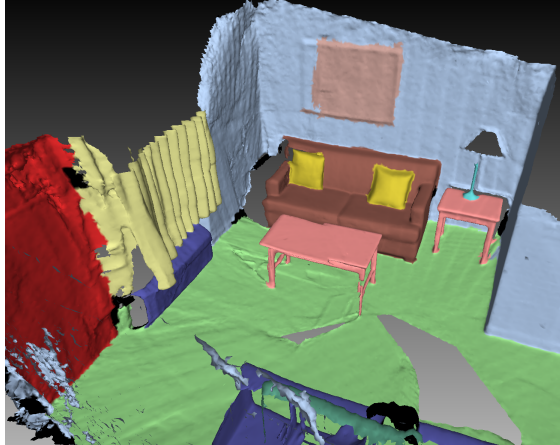


(d) Semantic segmentation with MSEG-HRNet

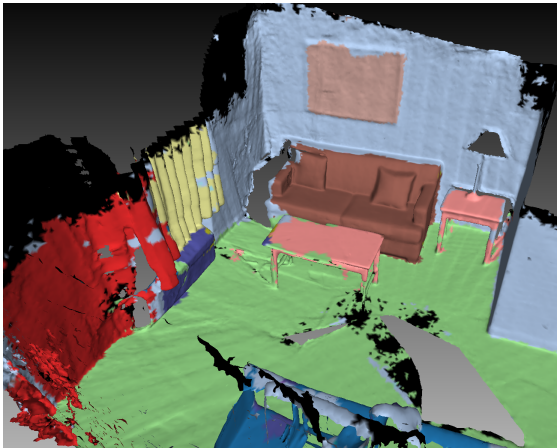
**Figure C.1.** Scene 0317\_01 from ScanNet validation set.



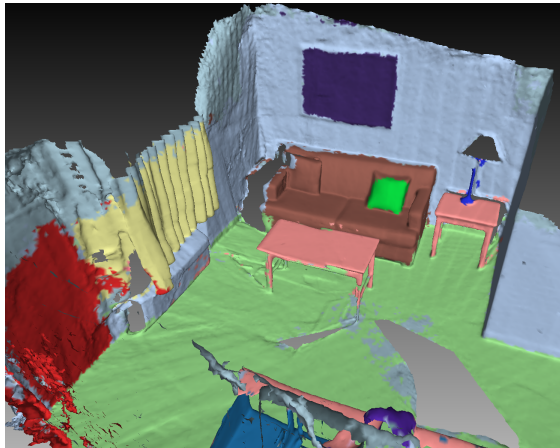
(a) Original 3D reconstruction



(b) Ground truth labels



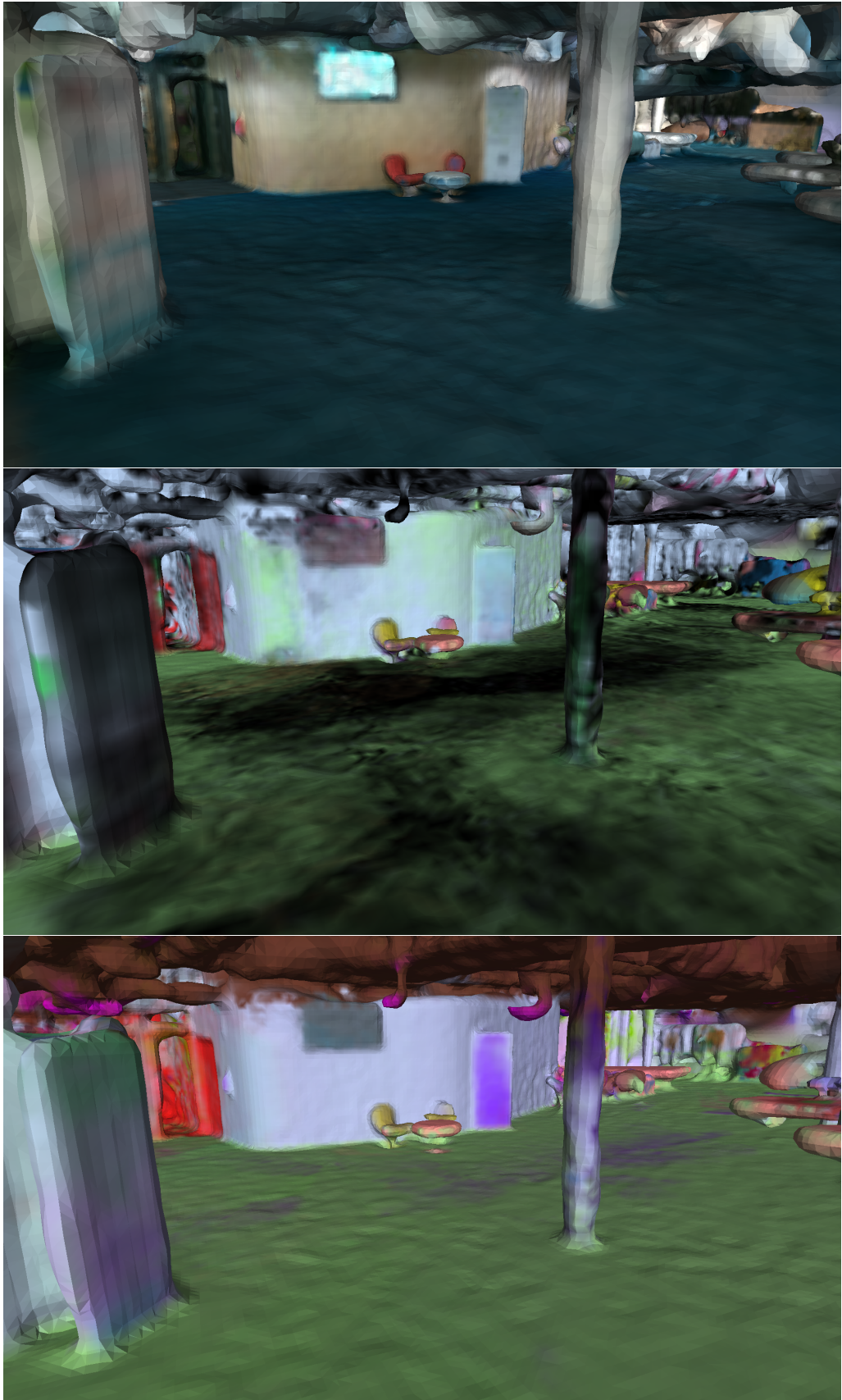
(c) Semantic segmentation with EfficienPS



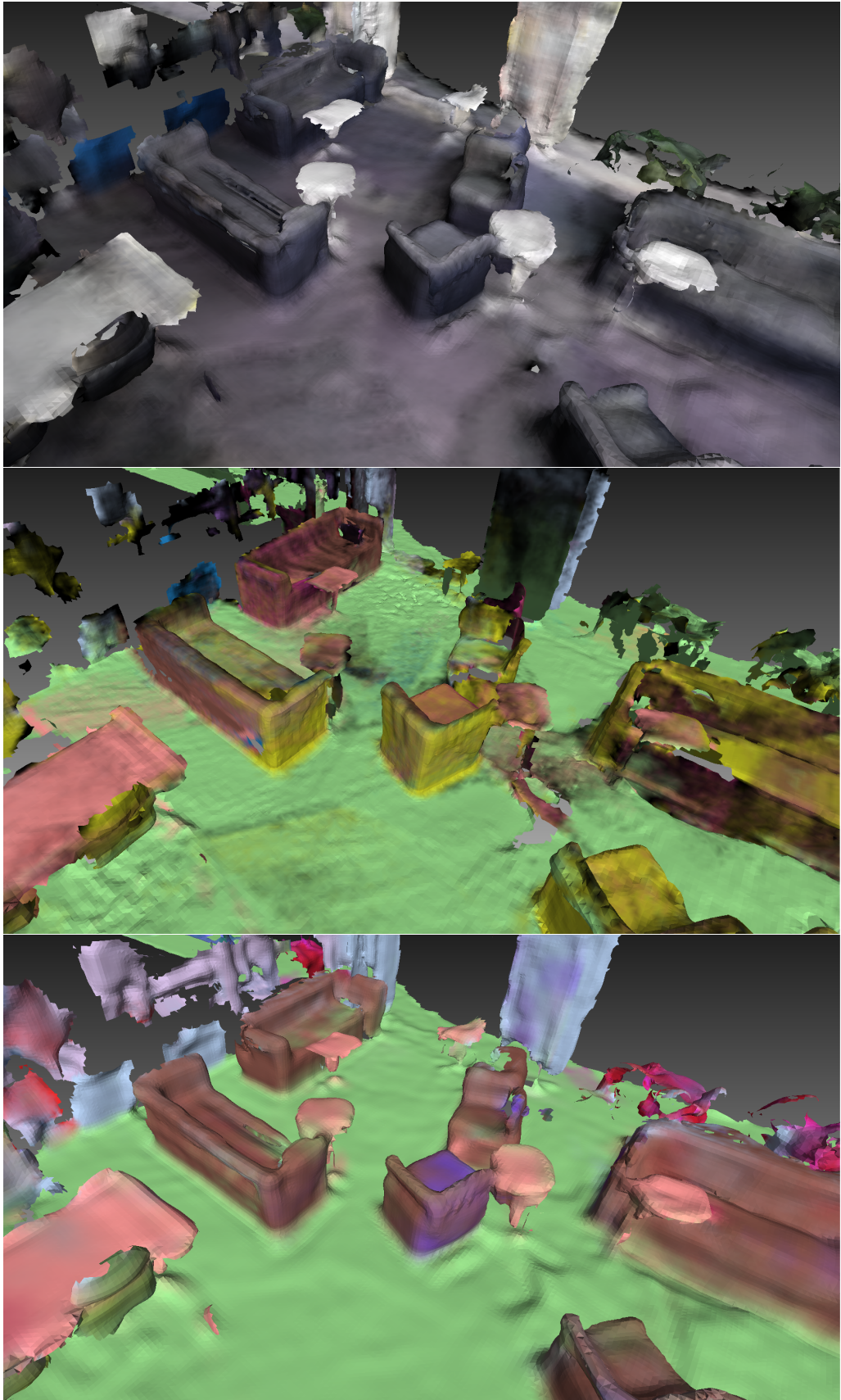
(d) Semantic segmentation with MSEG-HRNet

Figure C.2. Scene 0362\_00 from ScanNet validation set.





**Figure C.3.** Another three-dimensional view from the floor of a university building reconstructed with Realsense D455. Top: reconstruction from original RGB-D images, middle: EfficientPS segmentation, bottom: HRNet segmentation.



**Figure C.4.** Another view from the lobby of a university building reconstructed with Azure Kinect. Top: reconstruction from original RGB-D images, middle: EfficientPS segmentation, bottom: HRNet segmentation.



(a) EfficientPS semantic segmentation

(b) HRNet semantic segmentation

Figure C.5. More 2D comparisons on the ScanNet validation set.