Imperial College London
Department of Computing

Rapid Room Understanding From Wide-Angle Vision

Robert Lukierski

September 2016

Supervised by Prof. Andrew Davison and Dr. Stefan Leutenegger

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

Copyright Declaration

Abstract

There is an increasing pressure on mobile robotics, especially of the low-cost variety, to perform tasks, with ever increasing complexity, in an uncontrolled environment. Not so long ago mobile robots were expensive devices, employed rarely in the space exploration or industry, where the environment was created to be predictable and the cost was not a major factor. This is not valid anymore, as we see larger and larger numbers of robots reaching the service sector and end consumers. Examples come every year, from vacuum cleaners to the recent advances in autonomous cars. This thesis focuses precisely on mobile robot sensing capabilities and the ways to extend it. Our particular focus is on a low cost household mobile robots equipped with an omnidirectional camera, enabling extremely wide field of view. As this type of vision sensing is, while not entirely novel, still rather uncommon in the computer vision literature, we had to face multiple challanges due to the lack of reference datasets, algorithm implementations or ground truth sources.

These adversities shaped this thesis to a very large extent, therefore the structure mimicks the progress of computer vision that was done on the classical cameras, so we were able to reach higher levels. We start with a presentation of camera models and calibration techniques. Then we present both sparse and dense SLAM pipelines, allowing us to estimate the poses of the camera accurately and reconstruct dense depth maps respectively. Based on such foundations, we present an occupancy grid free space mapping method followed by a room shape estimation method, both purely relying on omnidirectional vision inputs.

All the presented methods were experimentally verified on synthetic data and on a large number of real world datasets, spanning various sizes and environment types. A separate chapter solely focuses on the engineering efforts required to build the necessary platforms, computing techniques and obtaining valid ground truth.

Contents

Abstract **iv**

Dedication **viii**

Acknowledgements **ix**

1 Introduction **1**

1.1 Background 4
1.2 Notation 11
1.3 Publications 12
1.4 This Thesis 12

2 System Design **15**

2.1 Robotic Platforms 15
2.2 Remote Control and Data Collection 16
2.3 Automatic Differentiation 21
2.4 GPGPU 22
2.5 Virtual Camera 24
2.6 Conclusion 28

3 Camera Models and Calibration **33**

3.1 Sensors, Lenses and Optical Path 33
3.2 Perspective Camera Model 36
3.3 Generic Axial Camera Model 37
3.4 Fisheye Model 38
3.5 Calibration 39
3.6 Spherical Panoramic Model 41
3.7 Conclusion 43

4 Omni Sparse SLAM **45**
4.1 Motivation . . . 45
4.2 BRISK Feature Detector and Descriptor . . . 46
4.3 Match Filtering . . . 48
4.4 Bundle Adjustment . . . 48
4.5 Additional Pose Graph Constraints . . . 50
4.6 Camera to Laser Scanner Calibration . . . 51
4.7 Camera to Ground Truth Tracker Calibration . . . 52
4.8 Experimental Results . . . 54
4.9 Conclusion . . . 56

5 Omni Dense Depth Map Estimation **59**
5.1 Dense Representation . . . 59
5.2 Cost Volume and Plane Sweep . . . 60
5.3 TV-L1 Regularization . . . 60
5.4 Depth Standard Deviation Estimation . . . 62
5.5 Experimental Results . . . 63
5.6 Conclusion . . . 74

6 Room Understanding **77**
6.1 Introduction . . . 77
6.2 Free Space Mapping . . . 78
6.3 Parameterized Model Fitting . . . 108
6.4 Conclusions . . . 134

7 Conclusions **137**
7.1 Contributions . . . 137
7.2 Recommendations . . . 139
7.3 Future Research . . . 141

Bibliography **143**

Dedication

To my family and friends.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Andrew Davison for giving me a chance to join his lab and pursue my interests in robotics and computer vision. His support and patience were essential, especially when I had more doubts and loosing faith in my own work. He created a lab with very open, infromal and welcoming work environment that helped me exchange and pursue novel ideas. This thesis would not be possible without his help. Thank you Andy!

I am also grateful for my second supervisor, Dr. Stefan Leutenegger, for his expertise, attention to detail and advice. His valuable comments improved this thesis considerably.

I would like to, posthumously, thank my Masters supervisor, Dr. Marek Wnuk, who many years ago, got me interested in robotics and vision at a more than hobby level.

During the past four years it was a pleasure to work with many brilliant collegues in our lab, alas, too numerous nowadays to fully list here. In particular, I would like to thank Jacek Zienkiewicz for many fruitful discussions, collaboration and understanding. Ankur Handa and Jan Jachnik for sharing their mathematical prowess. Steven Lovegrove and Hauke Strasdat for developing and sharing so many useful tools I could rely on. Owen Nicholson for making the lab run smoothly and helping me with college bureaucracy.

Finally, I would like to thank my family and friends, to whom I dedicate this thesis. My parents always gave me love, support and encouragment that I needed when choosing and pursuing my interests. I would also like to thank my friends, both from Wrocław and London, who often helped me through difficult times.

Research presented in this thesis has been funded by Dyson Technology Ltd. for which I am extremely grateful.

Chapter 1

Introduction

Contents

1.1 Background **4**

1.1.1 Camera Models and Calibration 4

1.1.2 Sparse SLAM 5

1.1.3 Dense SLAM 7

1.1.4 Free Space Mapping 8

1.1.5 Parametrized Model Fitting 8

1.1.6 Datasets 10

1.2 Notation **11**

1.2.1 Basic Symbols 11

1.2.2 Spaces and Manifolds 11

1.2.3 Frames and Transformations 11

1.2.4 Camera Projections 12

1.3 Publications **12**

1.4 This Thesis **12**

In January 2007 Bill Gates, the founder of Microsoft, published an article in the Scientific American titled “A Robot in Every Home”. There, with an incredible foresight, he predicted that the rapid progress in the field of computing and dissemination of personal computers across the globe that happened in the late XX/early XXI century, will repeat itself in the field of robotics. Gates writes a paragraph that could describe his early days in the IT industry:

“Imagine being present at the birth of a new industry. It is an industry based on groundbreaking new technologies, wherein a handful of well-established corporations sell highly specialized devices for business use and a fast-growing number of start-up companies produce innovative toys, gadgets for hobbyists and other interesting niche products. But it is also a highly fragmented industry with few common standards or platforms. Projects are complex, progress is slow, and practical applications are relatively rare. In fact, for all the excitement and promise, no one can say with any certainty when or even if this industry will achieve critical mass. If it does, though, it may well change the world."

Almost ten years later, an ample amount of time in new technology related fields, his paragraph could, as he predicted, as well describe the current state of affairs in the robotics sector. We already have a number of blue-chip companies manufacturing high end robotic systems for business, ranging from the military (General Atomics drones, Boston Dynamics), through medical (DaVinci) to the well established manufacturing robots (ABB, KUKA, Fanuc). In addition to that we are starting to observe the robotics entering the consumer markets. In the early days these were gadgets for keen enthusiasts (the first Roomba vacuum cleaner from iRobot in 2002), but nowadays we are starting to see robotic devices fully capable to perform their intended primary functions (Dyson 360 Eye or Robomow lawn mowers). As Gates envisioned we also see, in recent years, a staggering number of start-up companies, trying to build novel household robots, autonomous cars, 3D printers or improve the sensing capabilities with deep learning techniques.

The prediction contains a forewarning. Robotics projects are still exceptionally complex, spanning multiple fields of science and engineering, and progress is slow. More than in computing (like it was the case with "Moore‘s law") it is likely to be bound by physical limitations and applications are still relatively rare. However, we think that the field is still uncharted territory (especially in the applied and industrial contexts) that gives us great hope and craving for the unexplored avenues and the possibilities it can lead to.

This work presents another step on the road to fully intelligent and autonomous household robotics. In a large part the problems we had to solve were brought to us by the captains of our industrial sponsor. This unique relationship enabled us to perform our scientific duties without loosing the sight on the practical aspects and possible future applications.

Early generations of mobile household robots had extremely limited sensing. Often it was just a set of bumpers, either mechanical or short range infrared sensors. Such robots had simple reactive behaviour, bouncing against the obstacles and randomly covering the surface area of the floor. This was not very efficient, but had the advantages of simplicity and low cost. However, the popularity of smart phones brought unexpected side benefits. The powerful embedded processors, along with the sensors used in such devices (many kinds of cameras, IMUs) became significantly cheaper due to the very large scale of manufacturing. These technologies are now entering the field of robotics, enabling the consumer market and bringing the change it sought for years.

Recent vacuum cleaner robots have considerably improved in this aspect and are equipped with better sensors. Some have more traditional approach, similar to the research platforms of the past, and equip the robot with an ingenious laser scanner (Neato) that, while not as precise, is much cheaper than professional sensors of such type. This enabled basic navigation and localization. It might not seem much, but this alone allowed such robots to clean much more efficiently (for example avoid cleaning the same spot twice).

Navigation and localization are often realized with the method called SLAM — Simultaneous Localization And Mapping. It is an interesting problem, where the question is: "How, from sensor data, build the map of the environment and estimate our position within this map at the same time". At the first sight it seems impossible, as it sounds like chicken-and-egg problem. However, there are well established techniques to solve such problems, and we'll discuss them in greated detail in the following subsections.

SLAM can be done on the laser scanner data and it is what happens inside Neato robot. Another major breakthrough was Visual SLAM — localization and mapping from vision. For many years it was a purely academic endeavour, with real-time implementations published quite recently (in the grand scheme of things). Despite that this methods have already found its way into commercial products. For example the Dyson 360 Eye that performs the Visual SLAM from the omnidirectional camera.

The omnidirectional camera was an interesting choice, as its wide angles correspond well with the requirements imposed on the robot's localization. It is much easier to recognize the place if we can search through the wide field of view. We will discuss the research on such extreme wide-angle cameras in the following subsections.

This robot however, despite being a tour-de-force and bringing sophisticated robotics to the masses, does not use the camera for mapping as such, only localization. The mapping is realized by the short range IR sensors on the front, so when the robot encounters an obstacle (and it knows its position at all times) it marks the point of the map as occupied. This therefore requires a lengthy exploration of the environment to build a complete map. One of the methods presented in this thesis aims to lift this limitation.

We can clearly see the pattern here. With each generation of household robots we get more sensors, thus more processing power required and the effect is more complex behaviour. We predict that this trend will continue, as there is still a lot of room for improvement. More detailed image analysis could help the robot avoid small (but surprisingly debilitating) obstacles such as cables on the floor or chair legs in between which the robot could get stuck. On the other end of the scale there is whole room and object understanding. Such robots could classify and later recognize rooms, which could be important from the Human-Machine interaction point of view (so the user can refer to the robot with room names, e.g. “kitchen”, “bedroom” etc.). Such large scale understanding could also help the robot go between the rooms (doors) or avoid falling down the stairs. In this thesis we present a novel approach for purely geometric room estimation from omnidirectional camera.

1.1 Background

1.1.1 Camera Models and Calibration

Omnidirectional catadioptric lenses have been present in the fields of robotics and computer vision since the late 80’s — early 90’s. It combined the ideas of a fish-eye lens imaging and catadioptric systems (as used in telescopes). The turning point was the seminal work by Shree K. Nayar and Baker [BN98] on catadioptric image formation, analyzing possible configurations of mirrors that preserve the single viewpoint constraint. Another important piece of work was by Christopher Geyer and Konstantinos Daniilidis [GD00], who proposed a unifying model of generalized central cameras. Detailed analysis of epipolar geometry for catadioptric cameras, proving that in such systems the epipolar lines are general conics, was presented by Svoboda and Pajdla in [SP02].

In the late nineties the robotics community exhibited higher interest in this type of optical system. The problems involved were navigation [GWSV00], localization [MGS07], visual odometry [TPD08] and obstacle detection [KMS02].

Camera calibration field has a long history, dating back to the World War I and aerial imaging. Early calibration methods either involved adding additonal optics to correct for lens distortions, a stellar calibration method, as the position of the stars is known with high accuracy and, more down to earth, calibration methods involving precisely known objects, such as a series of plumb lines or a known 3D object composed of 3 planes [Fau93]. The difficulty lies in manufacturing or measuring the calibration standard with high accuracy.

A turning point was the work of Zhang [Zha00], who proposed a method that required multiple images of a planar pattern, preferably with easily detectable corners. This sparked the development of multitude automatic camera calibration tools and novel kinds of patterns that could be easily printed.

The approach of Zhang was extended to the subfield of omnidirectional vision by Scaramuzza [SS07] and Mei [MR07].

1.1.2 Sparse SLAM

To perform sparse 3D reconstruction we need to detect landmarks and be able to estimate jointly their positions and camera poses. Landmarks are based on image features, like FAST corners [RD06] or SIFT features [Low99]. Detection of these features in the images is carried out using standard image filtering techniques. Detection is followed by augmenting the detected location with a sort of signature/-descriptor that encodes the local structure around it to make it descriptive to aid matching across different frames. Features can be either described by a simple image patch and then template matching or a specific feature descriptor, e.g. BRIEF [CLSF10]. If we know the camera model, have some estimates of the camera poses and we know where each feature is present in other images in a sequence, we can perform numerical optimization to recover precise camera poses and 3D landmark locations. The initial estimates for the camera poses can be calculated with 5 point [Nis04] or 8 point [Har95] algorithms for each stereo pair. This initialization serves as an aid to quick convergence towards a locally optimal solution.

One of the first visual SLAM systems with real-time performance is MonoSLAM [DMRS07], which employs the Extended Kalman Filter to perform on-line filtering.

The state of the Kalman filter consists of the current best estimates of the camera poses and landmark locations. At every step there is a prediction phase, where the prediction based on the model is performed, thus estimating the new state from the previous state. The subsequent phase is the update, where a new measurement is introduced as an “innovation” (residual), which corrects and updates the prediction phase estimate to obtain the new state.

Interesting approach in an application identical to the motivation behind this thesis, i.e. an indoor mobile robot appliance, similar conceptually to MonoSLAM, was presented in [JM05]. It features an EKF based Visual-SLAM, as well as, a relocalization method. The method employs a single perspective camera, unconventionally, pointed upwards at the ceiling, which turned out to be advantageous for tracking performance.

Another important approach was PTAM [KM07], also a feature based camera tracking and landmark reconstruction system. PTAM uses Bundle Adjustment [TMHF99], a batch method and sophisticated control logic to manage features, keyframes and execute in parallel threads. PTAM, similarly to MonoSLAM, uses plain image patches instead of feature descriptors. Similar, more modern, state of the art systems were presented in [FPS14], [ESC14] and [MAMT15].

Unfortunately, the non-standard projection geometry of omnidirectional cameras means that building a 3D vision system involves more complication than with standard lenses, and often much published omnidirectional work has concentrated on modelling and calibration (e.g. [Gey03, Bar04]) or estimating vanishing points (e.g. [BP12]). All the SLAM systems mentioned above make heavy assumptions on the camera model being perspective and epipolar lines being straight. One recent exception is [CEC15] that extends the method of Engel [ESC14] to fisheye cameras by replacing the closed form epipolar line equation with a Tylor expansion approximation.

The most comprehensive descriptions of omnidirectional systems in the context of SLAM can be found in the PhD theses of Christopher Mei [Mei06] and Davide Scaramuzza [Sca08]. Independently, they have described efficient calibration methods and employed the omnidirectional vision system, on a mobile robot with a laser scanner, to perform feature based SLAM, mainly for tracking and visual odometry.

1.1.3 Dense SLAM

Recently, there was a shift from sparse, feature bases systems towards dense reconstruction and tracking, mostly enabled by the advent of GPGPU computing providing teraFLOPS of computing power. The key work was the PhD thesis of Thomas Pock [Poc08], who employed Total Variation methods to solve inverse problems, such as image de-noising, optical flow and multi-view stereo, in a dense and high speed manner (thanks to GPGPU computing). In dense methods every piece of data (often every pixel) is used in the energy function (not only some specific feature points like described in the previous section) and the challenge arises in formulating the energy functions such that it models the problem we want to solve (e.g. image corrupted with noise). It is feasible to find the minimum of this function through numerical methods (thus preferably a convex function) and that the function balances the data term with a regularizer to achieve the desired result (e.g. preserving discontinuities). A novel, real-time, application of the dense optimization was the 3D dense reconstruction and tracking system DTAM by Newcombe *et al.* [NLD11].

Mainstream depth cameras are now beginning to be commonplace, producing dense depth maps, often of high quality, straight from the peripheral device. This sparked numerous new approaches to dense mapping (e.g. [NIH+11, WMK+12, WHH+11]). However, with passive RGB cameras, recovering dense depth and free space information is much more challenging, but there has been good progress in the vast computer vision literature on multi-view stereo (MVS) [SS01] that improved the quality of the resulting depth maps. An interesting approach was presented in [CC15], where the authors perform dense reconstruction with planar structure assumptions where the depth estimates would be invalid (textureless planar regions).

In some cases we may skip geometric 3D reconstruction entirely and infer depth even from a single image, as presented in [SCN05, SSN09]. These methods use machine learning together with a probabilistic model based on a Markov Random Field to estimate the depth map of the image. It is important to note that in such 3D reconstruction approaches we heavily rely on the training sets (as this is a machine learning method) and the complexity of the algorithms (and the size of the learning data) often makes it infeasible to run in real-time, although this changes recently due to more computing power and novel machine learning techniques.

Dense methods however still have to gain popularity with non-classical camera researchers. One notable dense work is [BVF09], in which the authors experimented with dense reconstruction from a sequence of omni images. However, the results were far from impressive and, at that time, not real-time. Therefore, for a long time, the sparse 3D reconstruction has remained, by large, a *de-facto* method. Only recently very interesting omnidirectional dense SLAM was presented by Miriam Schoenbein [Sch14], albeit non-monocular, but accompanied with thorough analysis of the camera models and presenting sparse ego-motion estimation alongside the dense mapping.

1.1.4 Free Space Mapping

Using occupancy grids as a free space map representation was introduced in [Elf89] and remains very popular in the robotics community to this day. Reconstructing 2D free-space maps from ultrasound range finders to laser scanners has been a standard robotics capability for many years (e.g. [Thr03, GSB05]). Some researchers extended that to the vision sensors as well [ML00].

Occupancy grids are not the only possible model of the world. In the early work of Chatila [CL85] the author proposes geometric primitives such as lines to represent the structure of the environment. In [MT02] the authors propose a system of planar surface representation, extending the map to 3D, but the input data for the estimator originates only from the laser scanners. The panoramic camera provides just the texture.

In our subfield of omnidirectional vision it is important to mention [KMS01, MNS02], where the authors estimate free space information, but from an omnidirectional stereo pair rather than a single moving camera that we describe in this thesis.

1.1.5 Parametrized Model Fitting

There are various previous approaches to room shape estimation from images, often from a single image which usually requires a machine learning approach to overcome the large amount of ambiguity present. One single image approach was published by Hedau *et al.* [HHF09]. This was a rather handcrafted approach to single image room estimation based on vanishing points and extracting line segments. This information was used to produce a set of possible layouts. These layouts were then scored with an evaluation function obtained from learned models. The best scoring box layout was then refined by estimating surface labels, using superpixel segmentation.

The work of Zhang *et al.* [ZSTX14] extended this to panoramic images and added more context to the room estimation (estimating other objects, like furniture). As in [HHF09] the method employs line segment extraction, vanishing point estimation, and hypothesis generation, both from line segments and image segmentation. A random forest classifier assigns classes to extracted object cuboids. Equipped with both whole room hypotheses and classified object cuboids, a trained SVM classifier is employed to choose the best hypothesis.

With the 3D map of the environment we can move to the more abstract level of understanding. An example of this can be applying the Manhattan world assumption, where the map is composed of orthogonal planes. One could argue that it provides basic semantic information, but mostly simplifies the map, which might be enough for the robot to operate. A successful approaches to Manhattan world in 3D reconstruction were presented in [FMR11] and [FCSS09]. The first employs a Bayesian method of maximum a posteriori estimation and relies on the training data to teach the classifier. The second method uses Structure from Motion on the initial stage and then employs Markov Random Field to associate points with a correct plane hypotheses. Unfortunately, the execution time of both methods makes them non-real-time (ranging from one to tens of seconds).

Probably the conceptually closest approach to ours is the one by Cabral and Furukawa [CF14]. It also employs wide field of view cameras and 3D reconstruction. The sparse SfM reconstruction is then converted into a type of occupancy grid. Then a graph cut is used to extract the floorplan from occupancy grid evidence. Additionally, the input image undergoes superpixel segmentation and dynamic programming labelling to classify pixels into floor, wall and ceiling classes. The overall results are impressive, going beyond cuboid estimation, but the method is aimed at high quality visualizations. The input comes from mosaicing the images from high resolution DSLR cameras, used by a skilled human operator, and the method runtime is in tens of minutes thus making it rather unsuitable for a low cost mobile robot.

In our work we have decided to try to go as far as possible without resorting to the machine learning techniques, not only due to performance reasons, or higher familiarity with the geometric methods, but mostly due to the lack of training data. There are no datasets from similar omnidirectional cameras from a point of view of a small mobile robot. We would need a lot of these, especially for deep learning methods, all with segmented and labelled scenes (e.g. walls) or on the object level. Such extensive dataset collection and labelling is a large scale enterprise that normally involves whole teams, running the robots in tens to hundreds of houses and a number of paid staff to perform the labelling (either directly or via Amazon Mechanical Turk). In addition, the nature of our application might present privacy concerns (private properties), for which the legal framework would have to be agreed.

The model fitting in our approach (see Section 6.3) was inspired by the differentiable renderer idea presented in [LB14], though in our approach the forward process operates on depth maps and not image intensities. We use an omnidirectional camera, thus making it incompatible with OpenGL that OpenDR is coupled with (OpenGL supports only perspective camera model). Therefore our approach, to be precise, could be called a differentiable raytracer.

1.1.6 Datasets

High quality datasets with ground truth are the established way in the field to test and evaluate new methods. As the field of omnidirectional computer vision is rather niche the availability of datasets is limited as well. Out of the few available none match our intended application - that is indoor mobile robot. NAIST-TRAKMARK [TK09] and The New College Vision and Laser Data Set [SBC+09] datasets are taken outdoors and employ the Ladybug camera (stitched image) positioned approximately at the level of human height. The dataset by Miriam Schoenbein [Sch14] is also outdoor and for an automotive application. SUN-360 [XEOT12] dataset contains very high resolution omnidirectional indoor panoramas, stitched from multiple high quality DSLR photographs. While the images are from the indoor environment they do not reflect the point of view of the robot (down on the floor) and the images are single and do not form a sequence that would be useful for evaluating SLAM systems. None of these datasets provide ground truth depth.

1.2 Notation

1.2.1 Basic Symbols

a	A lower-case symbol denotes a scalar (with common capital exceptions).
$\mathbf{a}$	A bold lower-case symbol denotes a vector.
$\boldsymbol{a}$	A bold lower-case italic symbol denotes a homogeneous vector.
$\mathbf{A}$	A bold capital symbol denotes a matrix.
$\mathbf{1}$	The identity matrix, optionally with dimension as subscript.
$\mathbf{0}$	A zero matrix, optionally with dimensions as subscripts.
$\mathcal{A}$	A set.

1.2.2 Spaces and Manifolds

$\mathbb{R}$	The Real numbers.
$\mathbb{C}$	The Complex numbers.
$\mathbb{R}^3$	The 3D Euclidean space.
$SO(3)$	The 3D rotation group.
$SE(3)$	The Special Euclidean group.

1.2.3 Frames and Transformations

$\underrightarrow{\mathcal{F}}_A$	A frame of reference in $\mathbb{R}^3$.
${}_A\mathbf{a}$	The vector $\mathbf{a}$ expressed in $\underrightarrow{\mathcal{F}}_A$.
${}_A\mathbf{r}_P$	The position vector from the origin of $\underrightarrow{\mathcal{F}}_A$ to point P expressed in $\underrightarrow{\mathcal{F}}_A$.
$\mathbf{C}_{AB}$	The rotation matrix transforming a vector ${}_A\mathbf{a} = \mathbf{C}_{AB}\,{}_B\mathbf{a}$.
$\boldsymbol{T}_{AB}$	A homogeneous transformation matrix.

As a composition of rotation matrix and position vector, we write a homogeneous transformation matrix as

$$\boldsymbol{T}_{AB} = \begin{bmatrix} \mathbf{C}_{AB} & {}_A\mathbf{r}_B \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{1.1}$$

For a detailed description of the Lie group and algebra formalism see [Var74, MLS94] and tutorial by Ethan Eade [Ead14].

1.2.4 Camera Projections

$\mathbf{u} = \pi\left(\mathbf{p}, \mathbf{x}\right)$	Projecting point $\mathbf{x}$ to pixel $\mathbf{u}$ with camera model parameters $\mathbf{p}$ (often omitted as assumed known and constant).
$\mathbf{x} = \pi^{-1}\left(\mathbf{p}, \mathbf{u}, d\right)$	Lifting pixel $\mathbf{u}$ at distance d into a point $\mathbf{x}$ with camera model parameters $\mathbf{p}$ (often omitted as assumed known and constant).

1.3 Publications

The work described in this thesis resulted in the following publications:
Rapid Free-Space Mapping From a Single Omnidirectional Camera [LLD15]
Robert Lukierski, Stefan Leutenegger and Andrew J. Davison
Proceedings of the European Conference on Mobile Robotics (ECMR), 2015
Room Layout Estimation from Rapid Omnidirectional Exploration [LLD17]
Robert Lukierski, Stefan Leutenegger and Andrew J. Davison
Proceedings of the International Conference on Robotics and Automation (ICRA), 2017
Additionally, coauthored the following paper:
Dense, Auto-Calibrating Visual Odometry from a Downward-Looking Camera [ZLD13]
Jacek Zienkiewicz, Robert Lukierski and Andrew J. Davison
Proceedings of the British Machine Vision Conference (BMVC), 2013
Further description is provided by the first author in his thesis.

1.4 This Thesis

This theis describes a full journey towards an omnidirectional camera equipped robotic system capable of higher level environment understanding. The major scientific contributions are:

- Thorough analysis of the dense mapping method (see Section 5.5),
- Free-space mapping from a single omnidirectional camera (see Section 6.2),
- Parametrized model fitting from a single omnidirectional camera (see Section 6.3).

In addition there are important engineering contributions:

- Generation of ground truth depth map from ElasticFusion models (see Section 2.5),
- Collection of multiple, application-specific, datasets. Each with ground truth depth map. Some with VICON ground truth tracking.

The structure and relations between various parts of this thesis, highlighting major contributions, is presented in Figure 1.1.

The structure of this thesis is as follows: in Chapter 2 we describe the system and engineering efforts to create the tools and methods necessary for further research, from the robotic platforms, through software techniques to methods of obtaining omnidirectional ground truth data. Chapter 3 gives an insight into the imaging performance and challenges of non-classical systems, along with a brief description of the camera models and our approaches to camera calibration. Chapter 4 presents a sparse SLAM system we have designed to, primarily, accurately estimate camera poses. In this chapter we also describe additional calibration procedures (laser, ground truth tracker) that rely on the pose graph structure of Bundle Adjustment. In Chapter 5 we demonstrate an omnidirectional dense mapping pipeline, accompanied with an extensive evaluation of the method and multiple filtering techniques against high quality ground truth. The subsequent chapters build on top of the sparse and dense pipelines to deliver higher level understanding. Chapter 6 describes the most important and novel contributions of this thesis. Section 6.2 introduces occupancy grid mapping from omnidirectional depth data and presents an extremely thorough experimental validation on synthetic, household or office datasets. Next, Section 6.3 describes an advanced parametrized model estimation method that fits a cuboid to the omnidirectional depth map, enabling room shape understanding. This method was also thoroughly tested on a high number of datasets, including border cases. Finally, we conclude with Chapter 7, summarizing our findings, recommendations and indicating possible future directions.

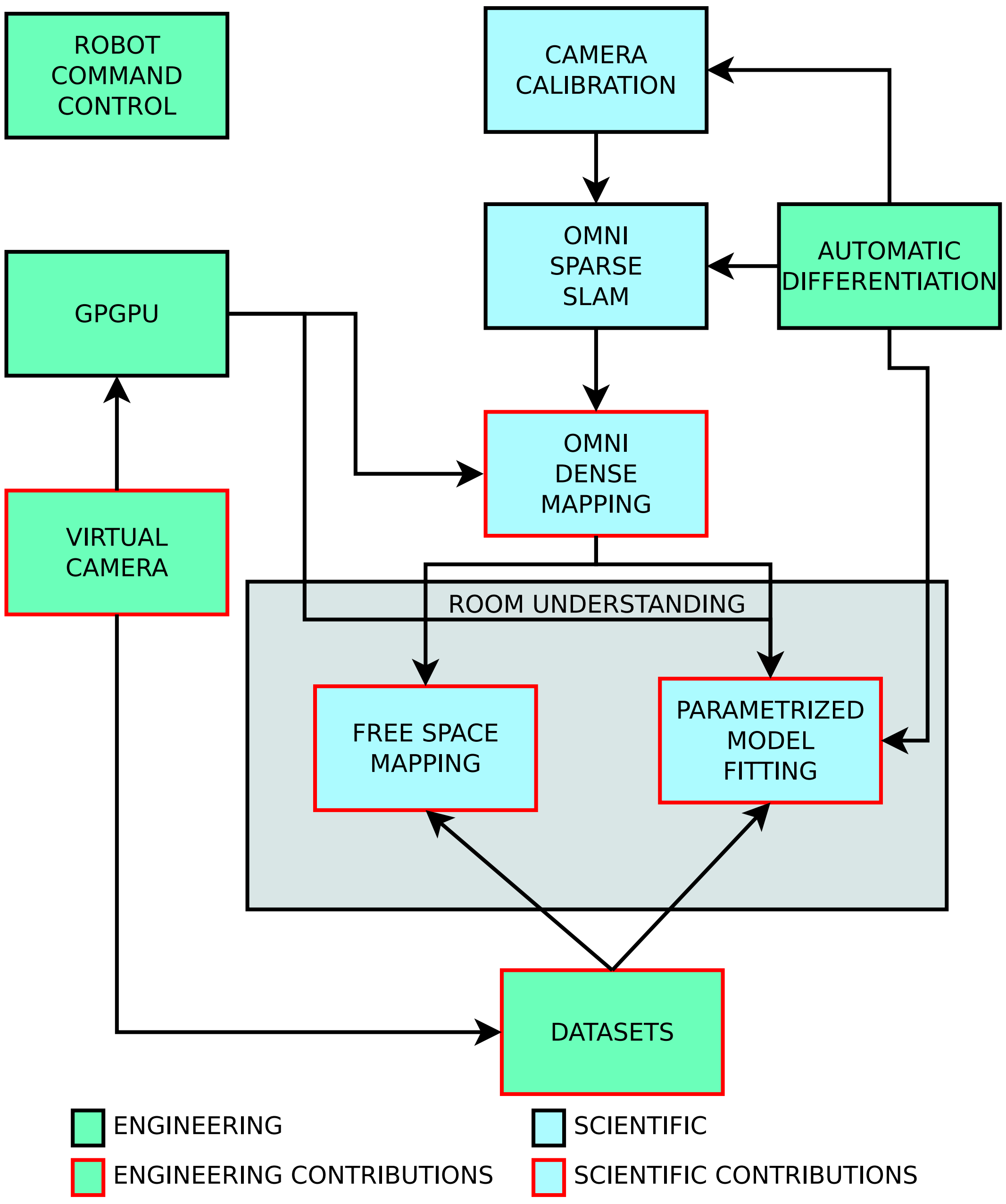


Figure 1.1: Diagram of structure and contributions.

Chapter 2

System Design

Contents

2.1 Robotic Platforms **15**

2.2 Remote Control and Data Collection **16**

2.2.1 Middleware and Protocols 17

2.2.2 Distributed System Architecture 19

2.3 Automatic Differentiation **21**

2.4 GPGPU **22**

2.5 Virtual Camera **24**

2.5.1 Evaluation 28

2.6 Conclusion **28**

In this chapter we describe the engineering efforts required for research and experimentation carried out in this thesis. The major focus point was flexibility of the system, from techniques that allowed interchangeable camera models to a high level distributed modular system architecture.

2.1 Robotic Platforms

Throughout this research we have used multiple mobile robot platforms, evolving as the requirements for the experiments changed. The major goal was to provide a research ready platform that will mimick the vision capabilities of the Dyson 360 Eye (Figure 2.1). Our first platform was the popular Pioneer 3DX platform, with an AMD APU MiniITX computer onboard (see Figure 2.2a). While it made a nice embedded solution, the AMD GPU supports OpenCL only, which is very limiting in prototyping and research use. Therefore we moved to a laptop equipped with an nVidia GPU that was carried by a Pioneer 3DX robot. This robot's mechanical ruggedness was helpful when collecting data with a SICK LMS500 laser scanner, as the sensor itself is rather bulky and heavy. The robot's weight and later size became detrimental, when the platform had to be carried to run experiments in the field, capturing dozens of datasets in multiple rooms and houses.

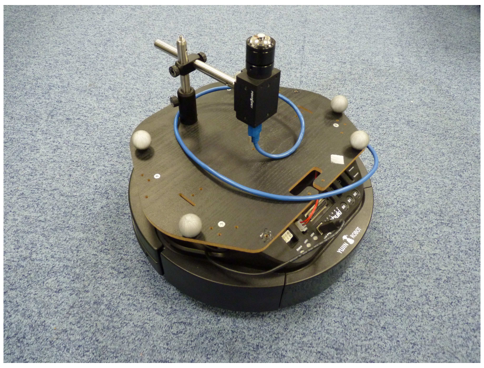
Figure 2.1: Dyson 360 Eye Robot. Courtesy of Dyson Technology Ltd.

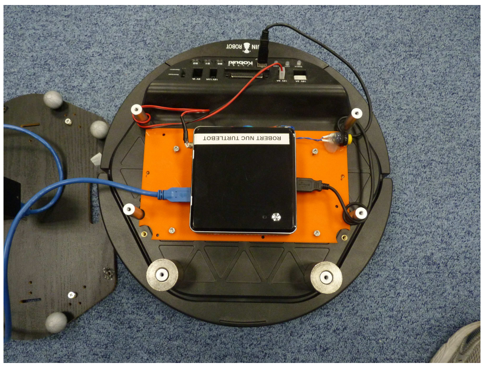
Therefore the primary platform was changed to a TurtleBot (see Figure 2.2b), a much simpler and lighter robot. Here the challenge was to embed at least basic computing resources to allow for dataset captures. Firstly, an nVidia Jetson TK1 board was integrated onboard. This is an ARM development board released by nVidia which, rarely for such platforms, supports CUDA GPU parallel computing. On the other hand the ARM architecture sometimes was problematic due to the lack of support from the tool vendors (e.g. VICON DataStream SDK). Thus, in the final version, the primary platform is equipped with an Intel NUC Skylake based computer (see Figure 2.2c). This does not feature CUDA capabilities (OpenCL as with AMD APU), but those were not required for data collection tasks.

2.2 Remote Control and Data Collection

A system to remotely control the robot was required, allowing full control over the robot and its sensors while at the same time capturing data streams of interest to a dataset file. Also, for remote operation live transmission of the data was required; mostly video feed, but other telemetry as well. As the research progressed the robotic platforms changed and the sensors were added, so the system should be modular to provide such flexibility.

(a) Pioneer 3DX with onboard AMD MiniITX computer.





(b) TurtleBot.

(c) Intel NUC onboard computer.

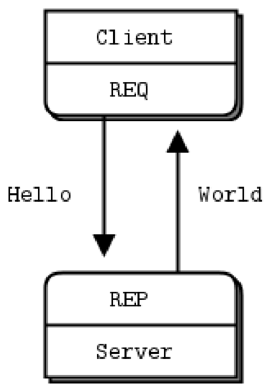
Figure 2.2: Mobile Platforms.

To fullfil these requirements a distributed system was designed and developed, with the focus on good engineering practices, flexibility and performance. The backbone of the system is middleware that enables reliable and flexible connectivity of the system modules.

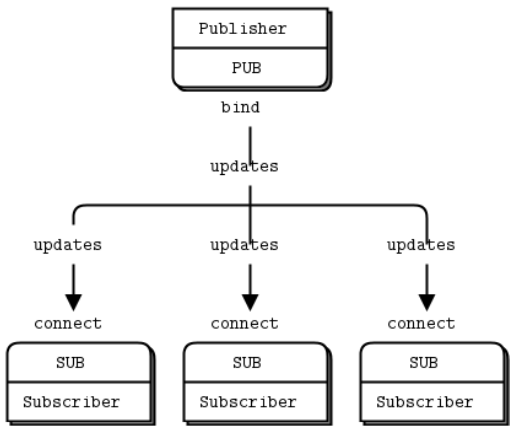
2.2.1 Middleware and Protocols

Our system employs the ØMQ (ZeroMQ) high performance asynchronous messaging library to provide connectivity between system modules running on the robot, as well as to the remote clients/operators. ØMQ was designed with performance in mind and was the highest performing communication library according to CERN's 2011 evaluation [DCE+11]. CERN's requirements are extremely demanding, as the particle accelerator infrastructure has 4000 servers, controlling 80000 devices that provide 2000000 properties (IO lines, actuator controls or sensor readings).

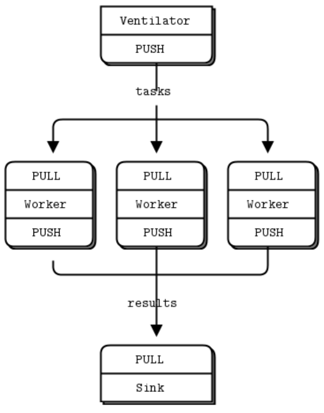
What is often more important is to understand the design patterns employed in distributed systems. The ØMQ library provides the basic building blocks, shown in Figure 2.3, from which the end user can create more advanced structures. In our case the Request/Response pattern is used for Remote Procedure Call (control) and the Publish/Subscribe pattern for sensor data.



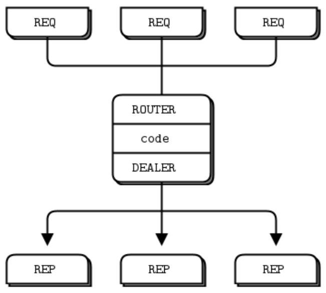
(a) Request/Response pattern.



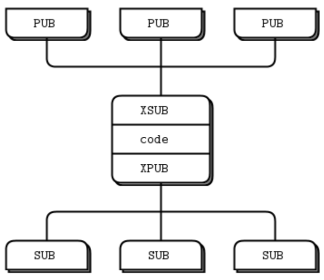
(b) Publish/Subscribe pattern.



(c) Push-Pull pattern.



(d) Router/Dealer pattern.



(e) Extended Publish/Subscribe pattern.

Figure 2.3: ØMQ connectivity design patterns.

The communication sockets can reconnect automatically without dropping the connection and have multiple endpoints, which enables multi-homing features known from, for example, the SCTP protocol that is used in telecom signaling layers. The sockets can use either TCP/IP, PGM Multicast, IPC (Interprocess Communication) or inprocess (inter-thread) communication. In our system we use inprocess within a single server app running on the robot, as this one has superior performance (based internally on a lock-free queue), and TCP/IP transport to communicate with the outer world (operator etc.). Inprocess transport also offers zero-copy functionality when transferring data buffers.

The over-the-wire binary data that is transferred with the ØMQ library is mostly Google ProtoBuf binary serialized data from respective data structures plus a number of binary payloads (e.g. ProtoBuf for frame metadata and payload for actual image data). The same binary format that is transmitted over the ØMQ library is also used when storing data to a dataset file, allowing the reuse of packet serialization/deserialization infrastructure.

2.2.2 Distributed System Architecture

Our system is composed on the robot side (server) of multiple modular services that form a processing and control pipeline. The main server app has the following responsibilities:

- Parse input configuration file (XML),
- Load respective modules (shared libraries), configure and create their instances,
- Provide brokers (patterns 2.3d 2.3e) to route the control and data streams from the outside world to the internal services,
- Provide event based profiling, collecting performance data from modules and transmitting the current metrics to the client (optional).

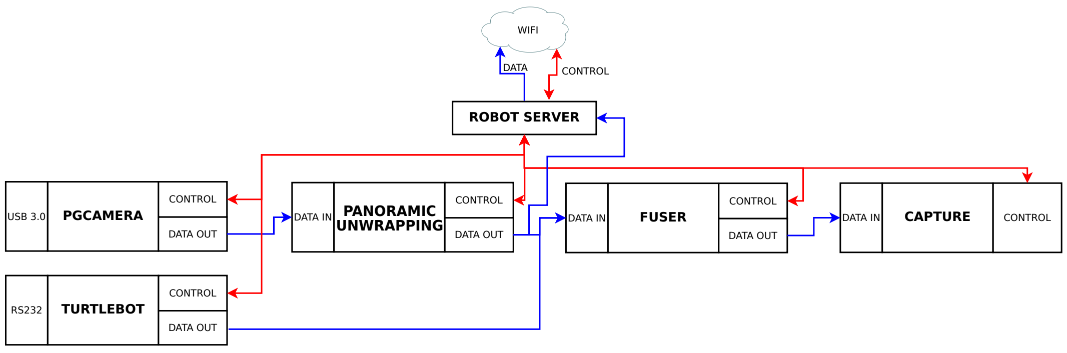


Figure 2.4: Basic System Setup.

A simplified example of the most common usage is shown in Figure 2.4. Red flows indicate control pathways (pattern 2.3a) and blue ones denote data flow pathways (pattern 2.3b). Data flow connectivity is largely defined by the input configuration file, thus the user can select which outputs to redirect to the capture, which should be only internally connected and which should be transmitted over the network for live feed (and optionally compressed). Each module (service) runs independently with its own internal control and data threads. This provides lock-free thread safety and division of responsibilities between modules.

Implemented services:

Capture	Responsible for opening new dataset file and receives packet to be stored on the HDD.
CLImageProcess	Generic OpenCL image processing.
TurtleBot	Interface to the TurtleBot mobile platform (via RS232).
Fuser	Combining input streams, based on their timestamps, into composite packet types. $SE(2)$ interpolation as well.
FWCamera	IEEE1394 Camera driver.
LMS500	SICK LMS500 Laser Scanner driver.
Panorama	Fixed function panoramic unwrapping, as described in 3.6.
PGCamera	PointGrey Camera driver.
Pioneer	Interface to the Pioneer P3DX mobile platform (via RS232).
USBCamera	V4L Camera driver.
VICONReceiver	VICON tracker client interface.

On the client side each server side module has a respective client API. The top level robot server is also seen by the client as a service, always available regardless of the configuration file contents, and allows queries for available services and their current operational state. The client API provides both C++ and Python interfaces and encapsulates basic processing to provide an uniform interface (e.g. decompression if the input live stream is compressed).

The modular architecture of the system has proved itself very succesful, being used and developed in the past 4 years, and it allowed for smoother transitions when the hardware or the requirements changed (changed the mobile platform, camera sensor, added more sensors etc.).

While systems with similar functionality exist ([QCG+09]), their design is rather poor and often exhibits the “Not Invented Here” anti-pattern. That being said the author acknowledges the recently started ROS 2.0 efforts to redesign the system with proper engineering standards and best practices in mind.

2.3 Automatic Differentiation

Automatic Differentiation (AD) is, as the name suggests, an automatic way for a computer program to evaluate the deriviative of a function. This technique depends on the chain rule (Equation 2.1) and the fact that a computer program is composed of simple mathematical operations and functions and all these have defined counterparts when computing the deriviative. The technique is fairly old, dating back to 1964 and the work [Wen64], but despite that it is still relatively rarely used in the fields of computer vision and robotics.

$$\frac{dy}{dx} = \frac{dy}{dw}\frac{dw}{dx}. \tag{2.1}$$

It is important to understand the difference between Automatic Differentiation and symbolic and numeric differentiation. In the symbolic case the input and the output are symbolic expressions. A numerical deriviative computes the function values and calculates finite differences (which often leads to round-off and discretization errors, especially when using 32 bit float types). Automatic Differentiation does not fall into any of these categories. It can be thought of simply as a source code transformation, in the C++ case done at compile time, where the code of a function is transformed into the code of the deriviative of the function. This code is then executed and results in the deriviative value at a point.

In this work we use this technique extensively, thanks to C++ operator overloading, both when running non-linear least squares optimizers on the CPU as well as when computing dense cost functions on the GPU. This allowed most of the algorithms presented in this thesis to operate on interchangeable (templated) camera models. Automatic Differentiation probably is not always as efficient as precalculated, hard-coded Jacobians (although the optimization skills of modern compilers should not be underestimated), but the convenience and flexibility that AD brings in prototyping/research use outweights slight performance losses. This is also the reason for the complete lack of manually derived Jacobians in this thesis. The method of Automatic Differentiation was employed at the camera calibration stage (see Chapter 3), in the sparse SLAM system (see Chapter 4) and at the heart of the parametrized model fitting method (see Section 6.3).

2.4 GPGPU

One of the key enabling technologies introduced in computer vision in recent years was General Purpose Graphics Processing Units. A slight deviation from fixed function 3D rendering pipelines allowed the graphics cards' computational power to be harnessed and employed to solve a much broader range of problems. In computer vision this technology was an enabler, both allowing the old algorithms to run in real time and sparking development of completely new methods.

The methods that benefited the most from the newly aquired computing power were the methods that could operate in a parallel manner, as required by the GPU architecture. This was natural execution environment for dense methods, operating on every pixel or every voxel of data.

In this work, while we have experimented with OpenGL Compute Shaders and OpenCL, we focus mostly on the nVidia CUDA programming environment. It is one of the first introduced on the market and the most flexible out of all three, allowing the use of almost complete C++11 in the device code. The only drawback is vendor lock-in, as the CUDA environment is nVidia proprietary.

In Figure 2.5 we show a basic diagram of a CUDA enabled GPU and interactions with the host.

The GPU is composed of multiple Streaming Multiprocessor units, each containing a number of processing units, sharing shared memory and executing a block of threads at the same time. Each processing unit has a local register file — that is the fastest memory, then shared memory and the slowest access one is the device memory, as it is shared by all the Streaming Multiprocessors and located off-chip. Due to the nature of this memory and its controller in the GPU it is vital to understand the importance of memory access patterns to achieve decent performance.

The memory on the GPU is connected with a very wide bus (e.g. 256 bits) and the memory chips itself (e.g. GDDR5) utilize double data rate with prefetch buffers, where the memory chip grabs adjacent datawords on the same row in memory to be read out in bursts. Therefore the code that accesses the device memory must be aware of this memory requirement and access memory in consecutive manner; for example, all threads in a block access the data from the same row (range of physical addresses). The key here is spatial and temporal coherency in the memory access.

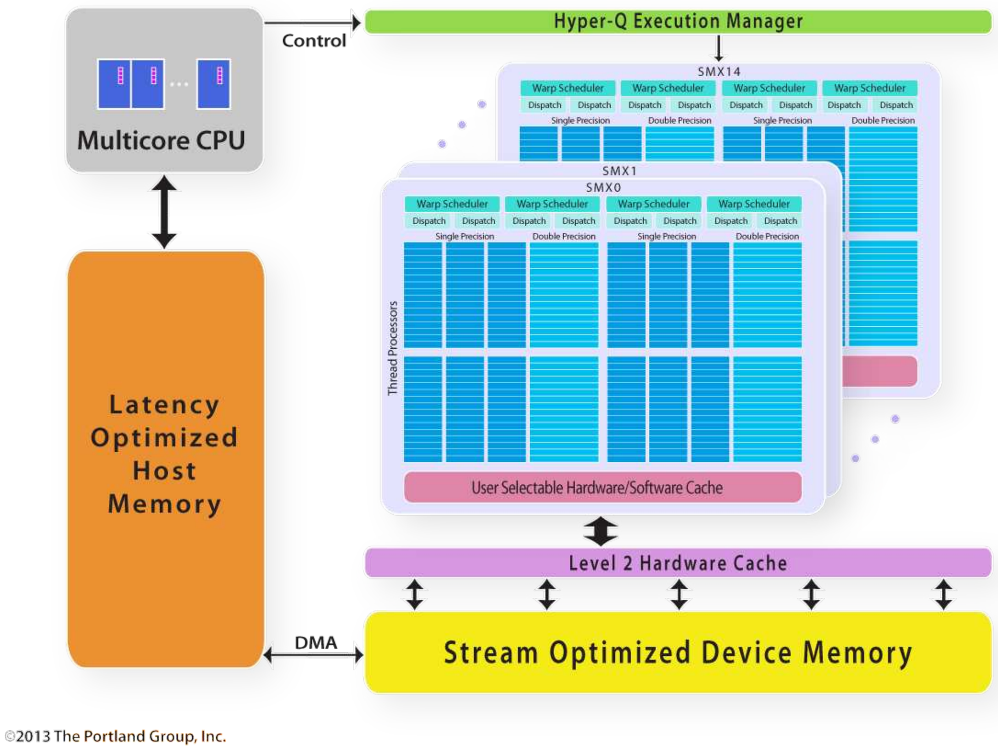

Control
Hyper-Q Execution Manager
Multicore CPU
SMX14
SMX1
SMX0
Warp Scheduler
Dispatch
Single Precision
Double Precision
Thread Processors
User Selectable Hardware/Software Cache
Latency Optimized Host Memory
Level 2 Hardware Cache
DMA
Stream Optimized Device Memory
©2013 The Portland Group, Inc.

Figure 2.5: CUDA Block Diagram.

Another thing worth mentioning is the host-device boundary. While there is Unified Memory in more recent CUDA versions and similar technologies in other parallel processing frameworks, these largely function thanks to the memory mapping unit and provide more convenience for the developer than significant performance improvement [LZCH14]. The data from the CPU (host side) has to travel via PCIe bus to rach the GPU (device side). The PCIe bus is usually the bottleneck and care must be taken when deciding which data and when to transfer over the host-device boundary. For example, for small buffers and simple processing algorithms the time it takes to issue host-device copy and dispatch compute kernels on the GPU might be greater than processing this data on the CPU in the first place.

In this work CUDA technology is used extensively, from simple image processing or unwrapping tasks, through dense depth estimation described in the Chapter 5 to the final tour-de-force described in the Section 6.3, where parallel processing computes the complex non-linear cost function over the entire depth map, calculating automatically derived partial Jacobians with chain ruled numerical deriviatives from the depth map data, and finally combining them before returning them to the CPU domain to be used in an non-linear least squares library [AKO] performing Levenberg-Marquardt iterative optimization.

2.5 Virtual Camera

One of the major problems encountered, and the main reason behind the extensive engineering work presented in this chapter, is the fact that catadioptric cameras are not mainstream in our field and there are no datasets and benchmarks (with ground truth) available (see Section 1.1.6).

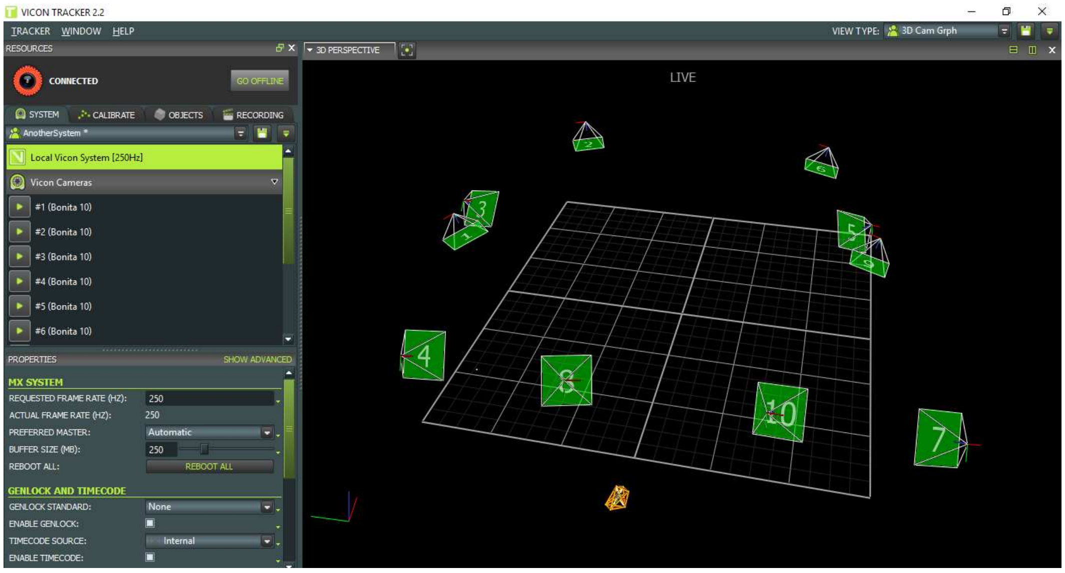
To verify our algorithms we needed a source of ground truth measurements, preferably of an order of magnitude more accurate (Test Uncertainty Ratio). For some methods we managed to use off the shelf equipment, for example a VICON tracker to evaluate tracking or a planar laser scanner to evaluate free space recovery. Testing and verifying dense reconstruction algorithms was much more challenging. Alas, we had no access to Velodyne 3D laser scanner, so we had to find alternative ways.

One of the possible ways is to generate fully synthetic data, both images and depth maps with perfectly known poses, of a 3D modelled scene with a ray tracing software. This approach is described in greater detail in section 5.5.1.

However, some readers might question the usefulness of synthetic data, created even with the best raytracer, when evaluating methods that should operate in real world conditions. To address this issue we came up with what could be considered a semi-synthetic ground truth.

The system relies on our precise VICON tracker (see Figure 2.6), so the datasets with groundtruth can be only captured in spaces with such system. The tracker is accurate to fractions of a milimeter and a degree over short time spans (below an hour). If longer capture is needed then the tracker might drift out of calibration due to environmental conditions (diurnal temperature changes), especially in our setup where the tracker cameras are mounted on plasterboard walls, thus not extremely rigid. A more detailed discussion of the VICON tracker's accuracy is presented in Section 4.8.

We have modified a PrimeSense RGBD camera and added the necessary tracking markers, as shown in Figure 2.7. The spatial relationship between the RGB and



MX SYSTEM

REQUESTED FRAME RATE (HZ): 250
ACTUAL FRAME RATE (HZ): 250
PREFERRED MASTER: Automatic
BUFFER SIZE (MB): 250
REBOOT ALL: REBOOT ALL

GENLOCK AND TIMECODE

GENLOCK STANDARD: None
ENABLE GENLOCK: ☐
TIMECODE SOURCE: Internal
ENABLE TIMECODE: ☐

Figure 2.6: VICON Tracker App showing 10 cameras in our experiment area.

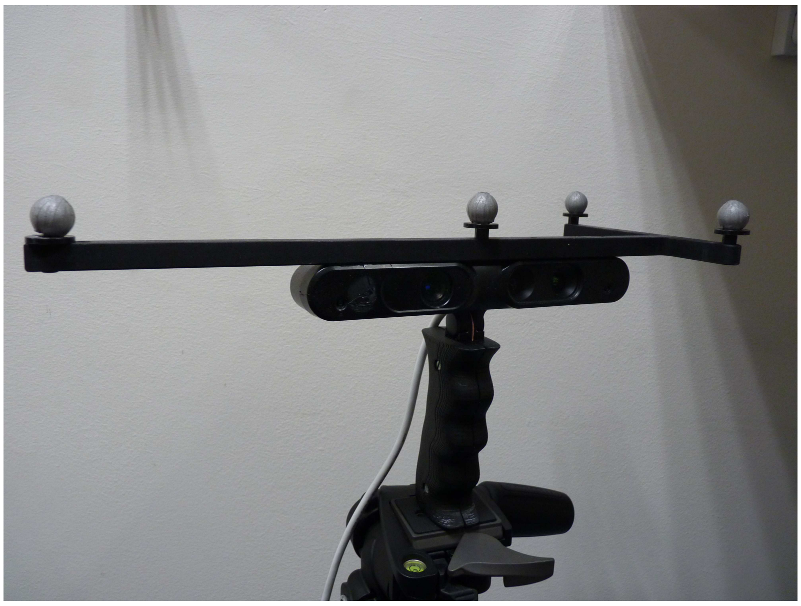


Figure 2.7: PrimeSense RGBD Camera with VICON tracker markers.

infrared sensors of this camera and the set of tracking markers was estimated with a purposefully manufactured calibration pattern as shown in Figure 2.8. The details are presented in Section 4.7 where the same method is used to estimate the initial pose change between the omnidirectional camera and the set of markers on the robot. The depth camera is accurate to centimetre levels, especially at close range.

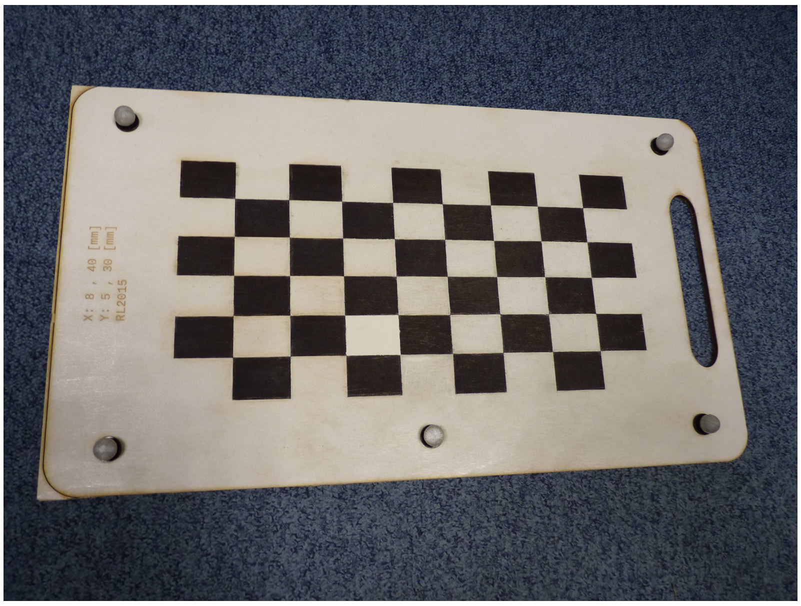


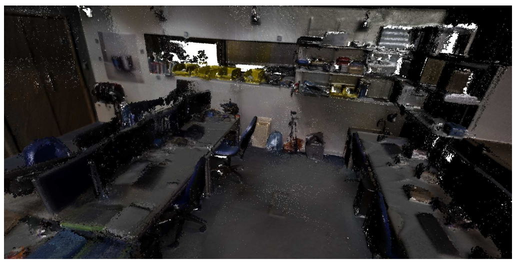
Figure 2.8: Combined vision and ground truth tracker calibration pattern.

With the modified RGBD camera and the known relationship between the tracker and camera's sensors we could start mapping the environment. We have used the ElasticFusion system [WLSM$^+$15], with some modifications (e.g. disabled built-in tracking). With this setup we are able to scan an almost watertight model of the surrounding environment, with no drift, with no scale ambiguity known from some SLAM systems and within a single common coordinate frame. The result is the model shown in Figure 2.9.

Now, having produced such a model, we can use OpenGL techniques and put a virtual camera inside this scene to render novel RGB and depth views. As OpenGL supports only perspective cameras we overcome this issue by rendering the scene from a selected viewpoint six times onto a cube (cube mapping), capturing all possible directions (see Figure 2.10). This cube map, both RGB and depth, can be then modified by a CUDA kernel that for each required ray from the target camera model (per pixel) samples the respective cube wall for color and depth values, thus producing the output RGB image and depth map. This approach works for any central camera model and any model parameters.

With the ability to generate any type of camera view from any point in a metrically accurate 3D scene model, we can generate ground truth data for a camera equipped with tracking markers. Our robot with an omnidirectional camera had the

(a) Model from the outside.


(b) Model from the inside.

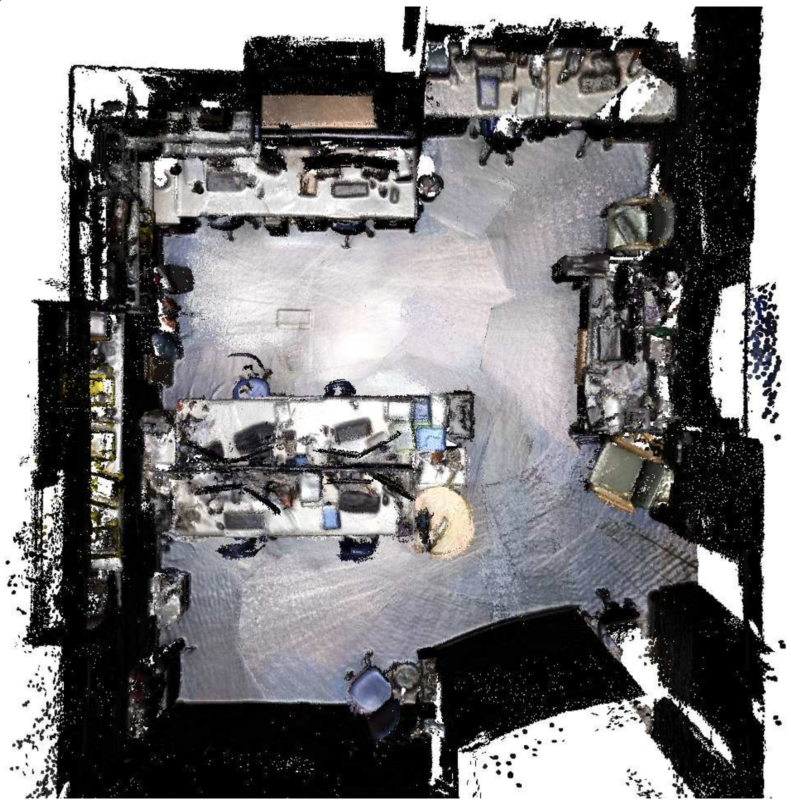

(c) Top down view with the ceiling removed.

Figure 2.9: Scanned full room model.

markers attached and the transformation between the markers and the omnidirectional camera itself was established as described in the Section 4.7. Therefore we are able to generate novel images and depth maps in the exact spots where the real camera on the real robot captured the frames. The virtual omnidirectional camera uses exactly the same parameter settings as the ones estimated for the real camera, including distortions. The operation of the system is shown in Figure 2.11.

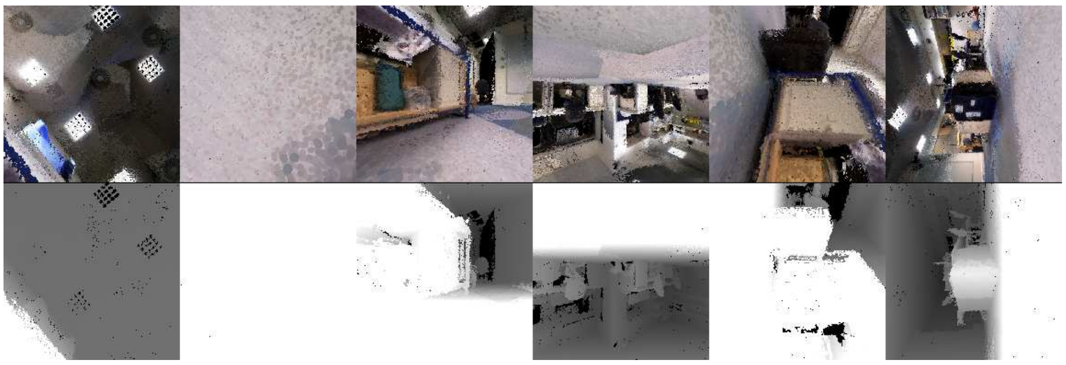


Figure 2.10: Scene rendered 6 times to produce a cube map.

2.5.1 Evaluation

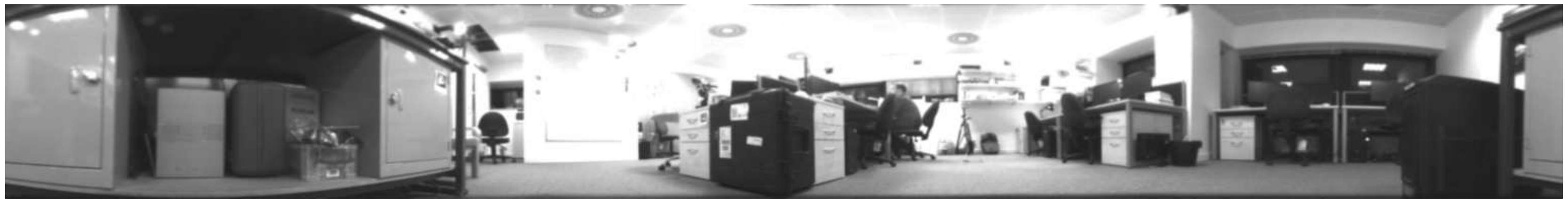
Axis	Ground truth $[mm]$	Measured $[mm]$	Figure
1	1801	1809.0	2.12a
2	724	730.9	2.12b
3	1399	1408.7	2.12c

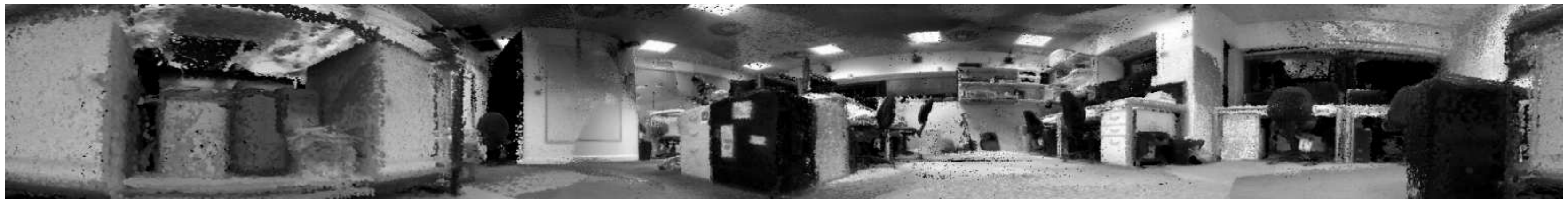
Table 2.1: Virtual Camera verification.

As a sanity check we have evaluated the dimensional accuracy of our 3D model. To do that we have measured 3 pieces of furniture in orthogonal directions, using 3D modeling software and compared the measurements with the readings of a Bosch PLR15 laser range finder in the same spots. The result can be seen in Table 2.1 and Figures 2.12a, 2.12b and 2.12c. We conclude that our 3D model is accurate to sub-centimetre levels, thus perfectly adequate for depth map ground truth purposes.

2.6 Conclusion

In this chapter we have described the engineering efforts required to perform the experiments presented in the remainder of this thesis. The type of camera this thesis is focused on brings multiple challenges to the researcher. Due to the low popularity in the field of computer vision, the availability of off the shelf datasets, tools, SLAM libraries etc. ranges from limited (camera calibration toolboxes) to nonexistent (most open source SLAM libraries make heavy assumptions on the perspective camera model and straight epipolar lines). Therefore this thesis required fundamental engineering efforts to be able to reach the kind of environment that is present for perspective camera researchers. We would like to highlight, in particular, the wide usage of Automatic Differentiation (2.3) and thoroughly evaluated and rather novel method of obtaining ground truth data (2.5).


(a) Real image frame captured from the omnidirectional camera.


(b) Omnidirectional image generated from the 3D model.

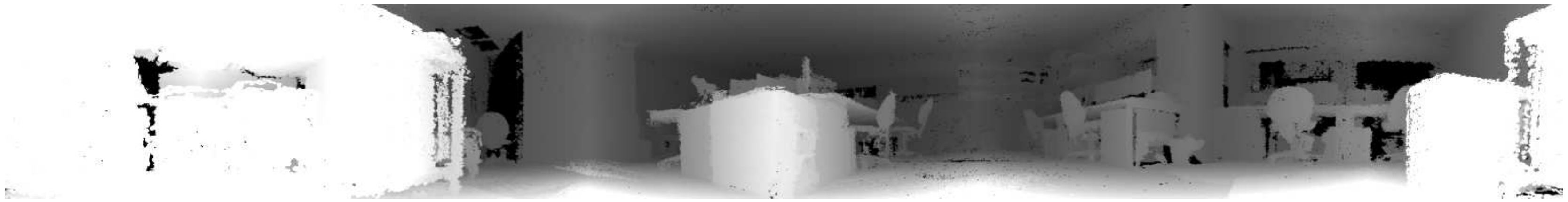
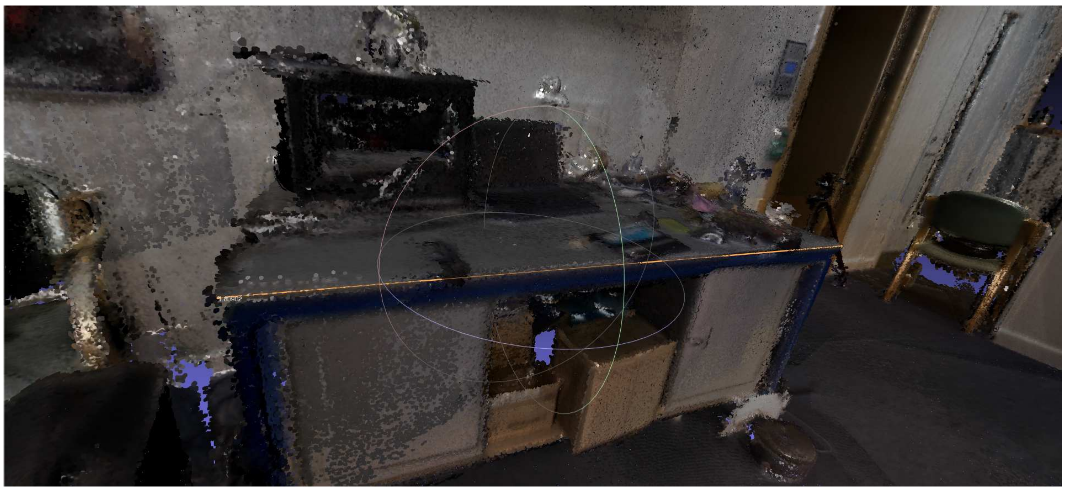
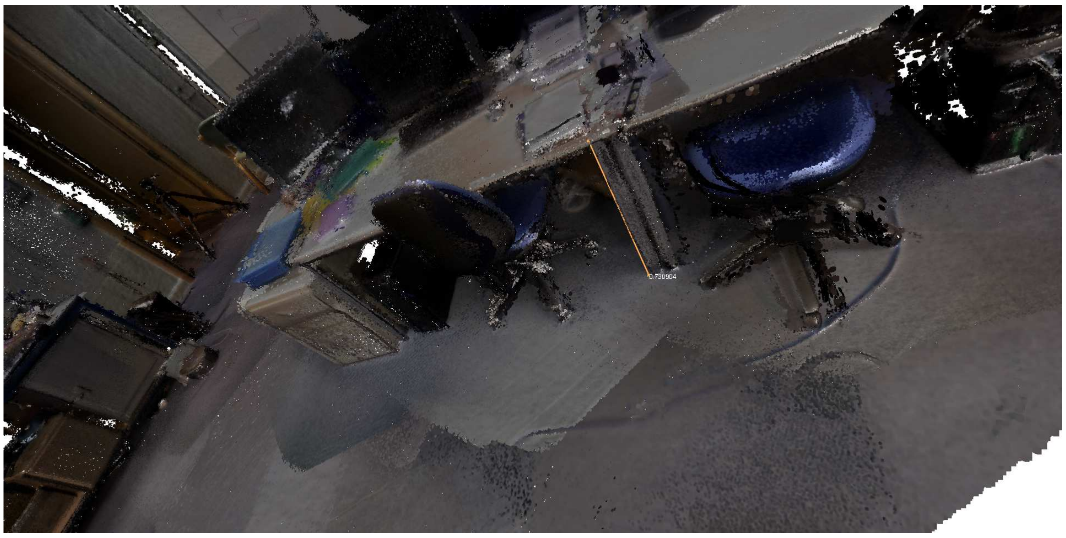

(c) Omnidirectional depth map generated from the 3D model.

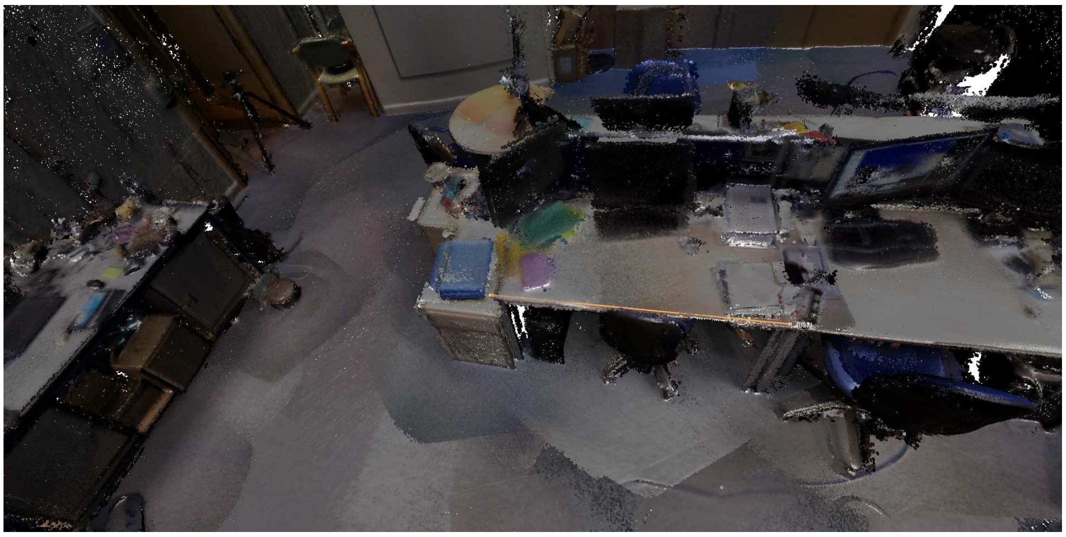
Figure 2.11: Results of the Virtual Camera ground truth.



(a) Verification, axis 1.



(b) Verification, axis 2.



(c) Verification, axis 3.

Figure 2.12: Verification of the absolute metric accuracy of our 3D model.

Chapter 3

Camera Models and Calibration

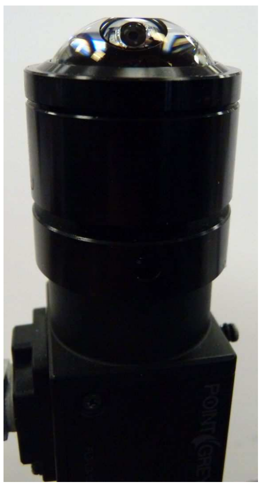
Contents

3.1 Sensors, Lenses and Optical Path 33
3.2 Perspective Camera Model 36
3.3 Generic Axial Camera Model 37
3.4 Fisheye Model 38
3.5 Calibration 39
3.6 Spherical Panoramic Model 41
3.7 Conclusion 43

In this chapter we describe camera models, especially those suitable for wide angle optics. A geometric camera model describes the relationship between the image and the corresponding rays of light that contributed to that particular part of the image. Models are simplifications of the optical path to various extents, trying to approximate imperfections in the lens and sensor construction. Later in this chapter we will focus on calibration — that is estimating the parameters of the model. This step is critical, as in this work we are focusing mostly on geometric reconstruction. Improper calibration would result in significant errors in 3D reconstruction and motion estimation.

3.1 Sensors, Lenses and Optical Path

Before we go through various geometric camera models it is worth discussing the peculiarities associated with wide angle lenses.


(a) Lens used throughout this work.

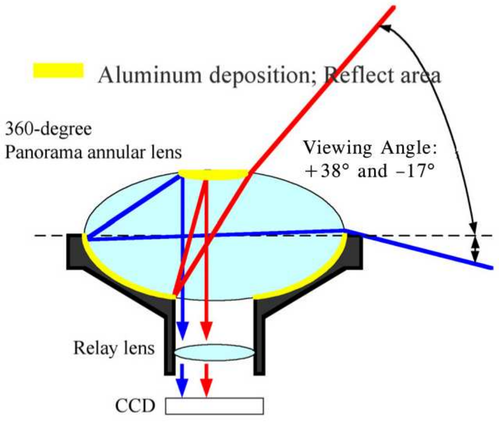

Aluminum deposition; Reflect area
360-degree Panorama annular lens
Viewing Angle: +38° and −17°
Relay lens
CCD
(b) Diagram of the lens.

Figure 3.1: Miniaturized catadioptric omnidirectional lens.

In this thesis we mostly focus on the miniaturized catadioptric lens presented in Figure 3.1a. This lens comes from the Sony RPU-C2512/RPU-C3522 series of panoramic CCTV cameras. The older designs of catadioptric lenses, more frequent in the literature of the subject, use a large conical shaped mirror positioned directly above a classical lens and camera setup. In our case this is integrated into the optical assembly of the lens with the mirrors being aluminium depositions on the glass element as seen in Figure 3.1b. This type of lens is similar to the one Dyson 360 Eye robot one and has the advantage of low height.

There are some compromises to be made when one decides to use wide angle lenses. As the optical path is more complex (see Figure 3.1b) and often involve more glass elements one might expect higher attenuation than in classical perspective lenses. This will contribute to the image noise, all other things being equal, as the sensor has to increase gain. Another issue contributing to lower output light levels is the low aperture of the wide angle lenses. This gives a very wide depth of field, allowing good focus, but at the same time decreases the amount of light reaching the sensor.

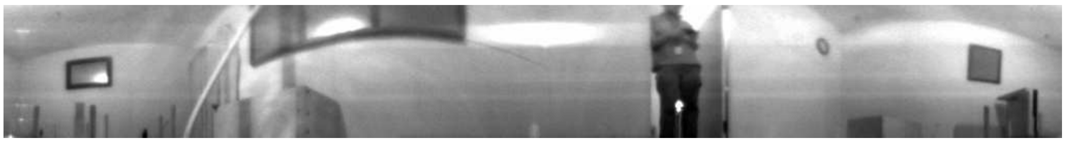


Figure 3.2: Optical artifacts from a low-cost omnidirectional lens.

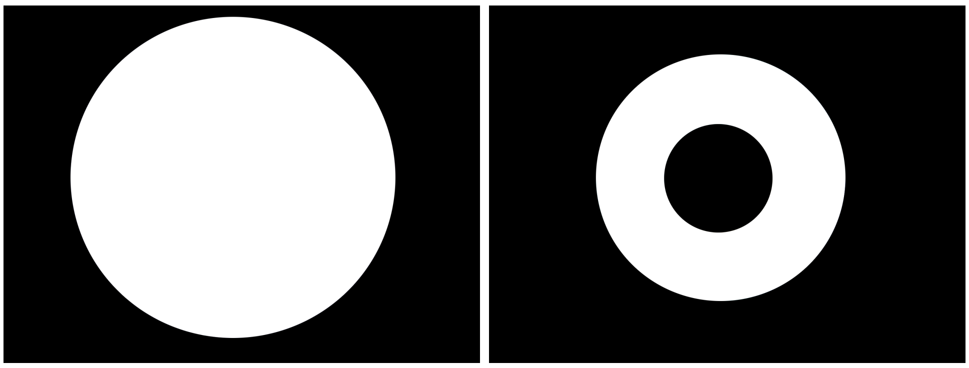

(a) Fisheye lens active area. (b) Catadioptric lens active area.

Figure 3.3: Active pixels on the sensor (e2v EV76C5706F 1/1.8”).

With wide angle lenses, composed of multiple glass elements, one also notices more pronounced vignetting (especially at extreme viewing angles), problems with chief ray angle and often caustic effects inside the lens. One rather extreme example is presented in Figure 3.2. The white specular reflection in the middle does not originate from the environment (as there is a white matt wall there), but from inside the lens. Simliarly, the horizontal stripes are due to lathe turning of the lens plastic. All of these effects make it difficult to rely on photometry. Such issues are rarely a problem when using high end lenses (e.g. Nikkor 6mm f/2.8s), but such lenses are often not manufactured any more and even second hand ones are extremely expensive.

The last issue worth mentioning is the active image area on the sensor. As we want to capture all the available viewing angles and the optics is circular, we expect that the projection will too have circular shape on the sensor. An example of this effect, for fisheye lens, can be seen in Figure 3.3a. In this case only roughly 50 % of the image area is active. This problem is more pronounced in the catadioptric case (Figure 3.3b) with the blind spot in the middle and in effect only 23 % of the pixels are active. While this is a waste of the sensor’s silicon real eastate and the link bandwith to transmit all the inactive pixel data, the more important issue here is the camera firmware. Camera or even sensor manufacturers integrate automatic exposure/shutter/gain and similar functionality either on the sensor chip or in the FPGA that is processing the data between the sensor and the data interface (e.g. FireWire or USB 3.0). All this built in funcionality assumes that the whole area is exposed and calculates pixel statistics based on this area. Sometimes setting ROI for readout might be helpful, or setting separate ROI for metering functions (available in Basler cameras). While this sometimes helps, the ROI shape is rectangular and thus still imperfect for wide angle lenses.

3.2 Perspective Camera Model

Before introducing more advanced camera models it is worth briefly describing the most widespread planar perspective camera model. This model assumes that the three dimensional environment is projected onto a plane, positioned at the focal distance of the lens, known as the image plane (see Figure 3.4).

Camera model parameters $\mathbf{p} = [f_1, f_2, u_0, v_0, s]^T$ are as follows:

f_1, f_2	Focal lengths in X and Y directions.
u_0, v_0	Principal point – centre of the optical axis.
s	Skew, often zero for modern cameras and digital video.

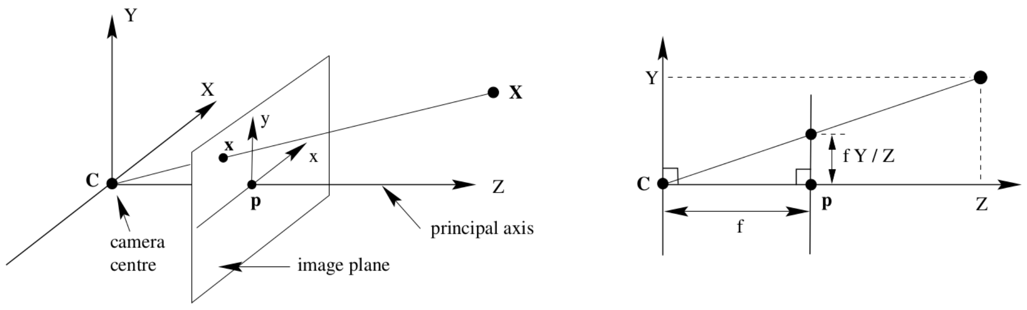


Figure 3.4: Pinhole camera model.

The perspective projection of a point ${}_C\mathbf{x} = [x, y, z]^T$ to the camera coordinate frame is as follows:

$$a = \frac{x}{z}, b = \frac{y}{z}. \tag{3.1}$$

Then to bring the point from the image plane to the sensor plane we need to scale it to the pixel coordinates $\mathbf{u}$:

$$\mathbf{u} = \begin{bmatrix} f_1 & f_1 s & u_0 \\ 0 & \gamma_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} [a, b, 1]^T. \tag{3.2}$$

Therefore the full projection equation from point ${}_C\mathbf{x}$ to pixel $\mathbf{u}$ is:

$$\mathbf{u} = \pi\left(\mathbf{p}, {}_C\mathbf{x}\right) = \begin{bmatrix} f_1 & f_1 s & u_0 \\ 0 & \gamma_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\frac{x}{z}, \frac{y}{z}, 1\right]^T . \tag{3.3}$$

3.3 Generic Axial Camera Model

In our system we employ the Geyer & Barreto ([Gey03, Bar04]) catadioptric camera model, depicted in Figure 3.5, which largely resembles the pinhole model with the addition of an extra parameter determining the curvature of the mirror.

The model has the following parameters $\mathbf{p}$:

ϵ	Mirror shape parameter.
$k_1, k_2, \gamma_1, \gamma_2$	Radial and tangential distortion coefficients.
f_1, f_2, u_0, v_0, s	Pinhole camera intrinsics as in 3.2.

First a point in the camera coordinate frame ${}_C\mathbf{x}$ is projected onto the unit sphere:

$${}_S\mathbf{x} = \frac{{}_C\mathbf{x}}{\|{}_C\mathbf{x}\|} =: (x, y, z)^\top . \tag{3.4}$$

The next step is to perform the perspective projection:

$$\mathbf{u}_u = \left(\frac{x}{z+\epsilon}, \frac{y}{z+\epsilon}\right)^\top =: (u_u, v_u)^\top , \tag{3.5}$$

where $\mathbf{u}_u$ is the undistorted image plane coordinate.

The ϵ parameter alters the projection, as it depends on the mirror shape and defines therefore the camera type, e.g. $\epsilon = 0$ is a classical perspective camera, $\epsilon = 1$ is a spherical mirror catadioptric camera.

From here the model follows the classical pinhole perspective camera, with radial and tangential distortions:

$$\begin{aligned} \mathbf{u}_d &= \mathbf{u}_u + d(\mathbf{u}_u, \mathbf{p}), \\ d(\mathbf{u}_d, \mathbf{p}) &= \begin{bmatrix} u_u(k_1\rho_u^2 + k_2\rho_u^4) + 2\gamma_1 u_u v_u + \gamma_2(\rho_u^2 + 2u_u^2) \\ v_u(k_1\rho_u^2 + k_2\rho_u^4) + 2\gamma_2 u_u v_u + \gamma_1(\rho_u^2 + 2v_u^2) \end{bmatrix}, \\ \rho_u &= \sqrt{u_u^2 + v_u^2}, \end{aligned} \tag{3.6}$$

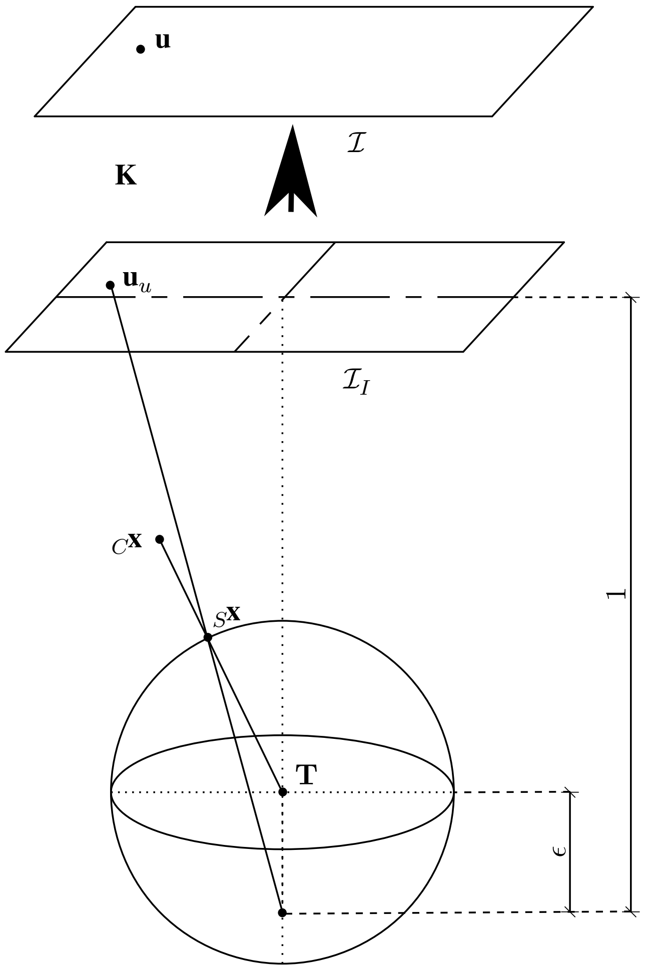


Figure 3.5: Catadioptric camera model (distortions part omitted).

transforming undistorted image plane location $\mathbf{u}_u$ into its distorted counterpart $\mathbf{u}_d$. This is followed by the multiplication of the pinhole camera intrinsic matrix:

$$\mathbf{u} = \mathbf{K}(\mathbf{p})\,\mathbf{u}_d = \begin{bmatrix} f_1 & f_1 s & u_0 \\ 0 & \gamma_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u}_d, \tag{3.7}$$

to compute the final pixel location $\mathbf{u}$ from $\mathbf{u}_d$. We denote the overall projection as $\mathbf{u} = \pi\left(\mathbf{p}, {}_C\mathbf{x}\right)$.

3.4 Fisheye Model

Although in this thesis we focus on catadioptric wide angle lenses, with the model described in 3.3 it is worth briefly mentioning a model that is better suited for more common fisheye lenses. This model is based on [KB06] with slight modifications. The implementation can be found in the OpenCV 3 library release.

The model has the following parameters $\mathbf{p}$:

k_1, k_2, k_3, k_4	Radially symmetric coefficients approximating projection curve.
f_1, f_2, u_0, v_0, s	Pinhole camera intrinsics as in 3.2.

The projection function starts with a classical perspective transform, transforming the point ${}_C\mathbf{x} = [x, y, z]^T$ to the image plane:

$$a = \frac{x}{z}, b = \frac{y}{z}. \tag{3.8}$$

Then the radial distance r from the optical axis is calculated, and the corresponding ray angle θ:

$$\begin{aligned} r &= \sqrt{a^2 + b^2}, \\ \theta &= \arctan r. \end{aligned} \tag{3.9}$$

Now the ray angle is used as input to the projection curve model:

$$\theta_m = F(\theta) = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9. \tag{3.10}$$

This modifies the original point on the image plane:

$$\mathbf{u}_d = \left[a\frac{\theta_m}{r}, b\frac{\theta_m}{r}\right]^T. \tag{3.11}$$

This point then is scaled with the camera intrinsic matrix as before, to bring it onto the sensor plane:

$$\mathbf{u} = \mathbf{K}(\mathbf{p})\,\mathbf{u}_d = \begin{bmatrix} f_1 & f_1 s & u_0 \\ 0 & \gamma_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u}_d. \tag{3.12}$$

Figure 3.6 illustrates the principle of operation of this model. One might notice that point $\mathbf{p}'$ is transformed to $\mathbf{p}$ and this originates from Equation 3.10.

3.5 Calibration

Camera calibration was performed with a chessboard (Figure 3.7a) or asymmetric circular blob pattern (Figure 3.7b) displayed on an LCD monitor, with advantages

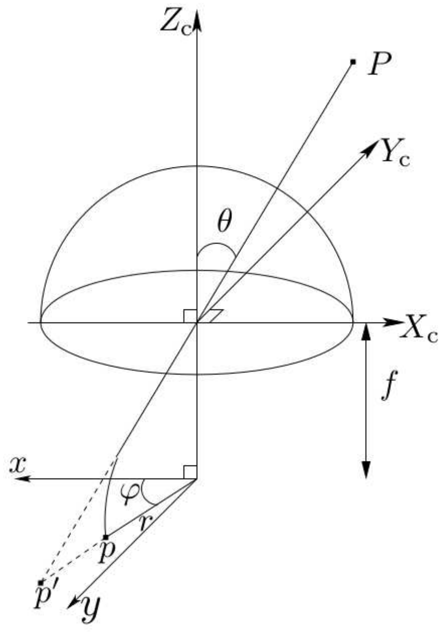
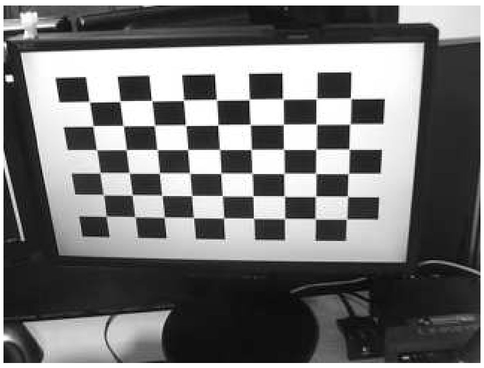
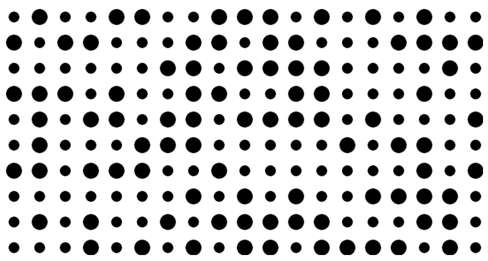


Figure 3.6: Fisheye camera model.



(a) Chessboard calibration pattern.



(b) Circular calibration pattern.

Figure 3.7: Various calibration patterns.

over a typical printout due to precision (pixel pitch, registration accuracy of 1 µm [dB05]), flatness and high contrast.

Ten images were taken by moving the camera around to cover the entire field of view. We employ Levenberg-Marquardt optimisation to minimise the following energy function with respect to the camera model parameters **p** and the camera pose

$\boldsymbol{T}_{WC}$:

$$E(\mathbf{p}, \boldsymbol{T}_{WC}) = \frac{1}{2} \sum_{i=1}^{m} \left\| \pi\left(\mathbf{p}, \boldsymbol{T}_{WC\,W}\mathbf{x}_i\right) - \mathbf{u}_i \right\|_2^2, \tag{3.13}$$

where m is the number of point correspondences, $\boldsymbol{T}_{WC} \in SE(3)$ is the pose of the camera, ${}_W\mathbf{x}_i$ is the location of a corner on the calibration pattern in Euclidean world coordinates and $\mathbf{u}_i$ is the corresponding image measurement of the same point on the image. By using this optimisation (with auto-differentiation) we routinely achieve average reprojection error around 0.1 pix or below.

3.6 Spherical Panoramic Model

A raw omnidirectional image makes subsequent steps like point feature matching and dense stereo complicated, so we perform a Look-Up-Table (LUT) unwrapping to a spherically mapped panoramic image for subsequent processing. This involves sub-pixel interpolation, but has the advantages that image dimensions fit well with GPU memory and that each pixel represents a uniform area on the surface of a sphere.

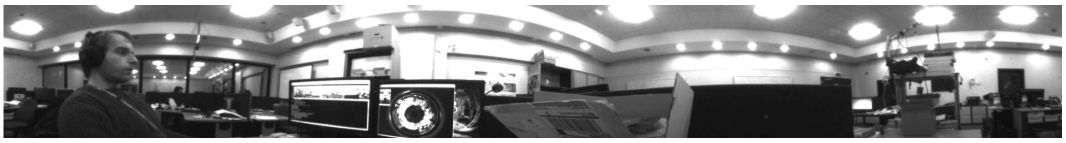
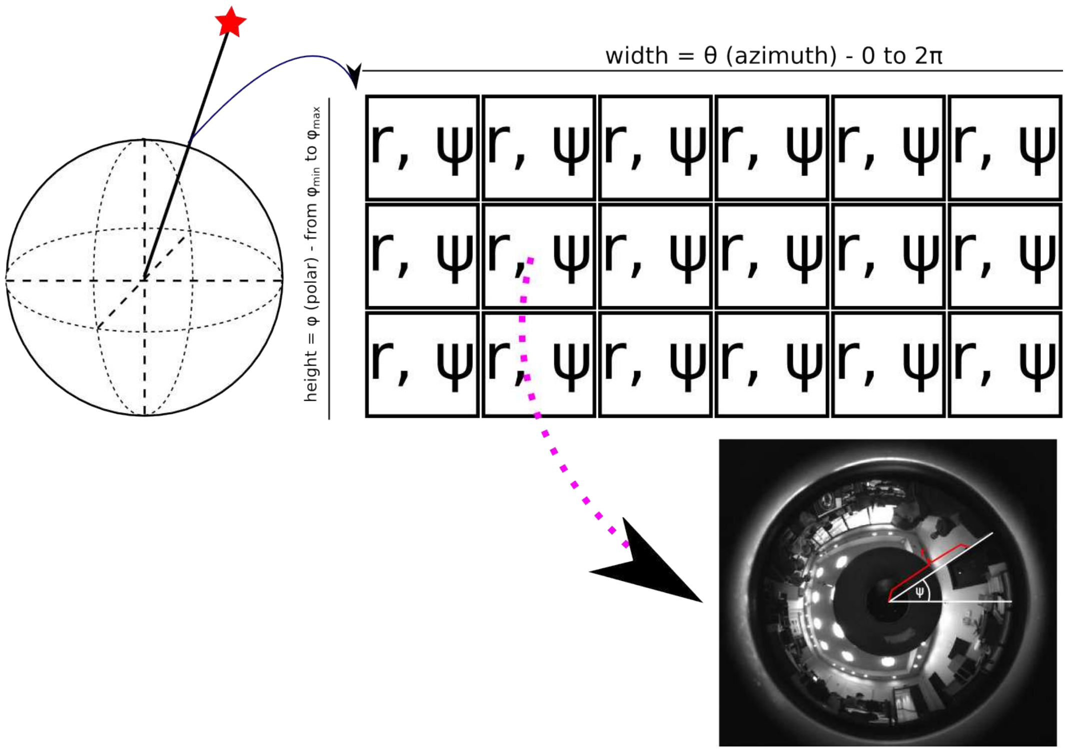


Figure 3.8: Panoramic unwrapping with LUT perfomed by the GPU

From each pixel on the spherical panorama (Figure 3.9) there is a mapping via the LUT onto the original image; however this mapping ends in polar coordinates (normalized r and θ) on the original image, as we have discovered that the centre of image projection varies with lens temperature. Therefore, to have better accuracy, instead of doing full calibration every time we just re-estimate the centre of projection and two radii and use that with the relative polar coordinates from the look-up-table to calculate sub-pixel image coordinates on the sensor.

Spherical camera model parameters $\mathbf{p}$:

w, h	Width and height of the target image (and LUT).
a_m, a_r	Vertical minimum viewing angle and angle range.



width = θ (azimuth) - 0 to 2π
height = φ (polar) - from φmin to φmax
r, ψ
ψ

Figure 3.9: Look-Up-Table principle of operation

The final projection model for the spherical panorama is as follows (${}_C\mathbf{x} = [x, y, z]^T$).

$$\mathbf{u} = \pi\left(\mathbf{p}, {}_C\mathbf{x}\right) = \left(w - (\arctan\frac{y}{x})\frac{w}{2\pi}, (\arccos\left(\frac{z}{\|{}_C\mathbf{x}\|}\right) - a_m)\frac{h}{a_r}\right)^\top, \tag{3.14}$$

where w and h denote the panorama image width and height in pixels, and a_m, a_r are the minimal vertical viewing angle and range respectively. These are estimated from the original camera model at the image boundaries.

Point ${}_C\mathbf{x}$ in the camera coordinate frame is converted to spherical coordinates and both spherical angles are transformed to panoramic image space yielding pixel coordinate $\mathbf{u} = (u, v)$ as the output.

The pseudo-inverse of the camera model transforms input pixel coordinate $\mathbf{u}$ back to a Euclidean normal vector ${}_C\mathbf{x}_n$:

$$\begin{aligned} \varphi &= \pi - \frac{u}{w}2\pi \\ \theta &= a_m + \frac{v}{h}a_r \\ {}_C\mathbf{x}_n &= \pi^{-1}\left(\mathbf{p}, \mathbf{u}\right) = \left[\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta\right]^T. \end{aligned} \tag{3.15}$$

3.7 Conclusion

While there is a large variety of non-classical camera models, (for example [Sca08] or [Sch14]), we have settled for the model described in the Section 3.3 due to its high resemblance to the perspective camera model and versality that the only custom parameter (ϵ) brings. However, for the omnidirectional camera case we convert the model and the input images with the spherical unwrapping described in the Section 3.6, as it is more memory efficient (no invalid image regions) and potentially enables the use of patch based computer vision methods (like feature detectors/descriptors described in the Chapter 4) as the spherical image is uniform with respect to the viewing angles.

Chapter 4

Omni Sparse SLAM

Contents

4.1 Motivation 45
4.2 BRISK Feature Detector and Descriptor 46
4.3 Match Filtering 48
4.4 Bundle Adjustment 48
4.5 Additional Pose Graph Constraints 50
4.6 Camera to Laser Scanner Calibration 51
4.7 Camera to Ground Truth Tracker Calibration 52
4.8 Experimental Results 54
4.9 Conclusion 56

This chapter describes our sparse SLAM pipeline for wide angle vision. Here sparse means that we only focus on specific parts of the input image, known as keypoints. Our main intention was to estimate camera pose for each frame with high accuracy, as this is required for the methods described in the following chapters.

4.1 Motivation

Our experimental robot has odometry which is not synchronized with image capture and for dense stereo reconstruction we need highly accurate camera pose estimates so we have implemented a keypoint-based omnidirectional structure from motion and bundle adjustment system to globally register all of the image frames in each experiment (even those in large rooms with multiple scan circles). Wheel odometry measurements are included in the problem to define the overall metric scale.

In order to provide a good guess for the optimiser, we use the robot's odometry as the initialisation for the camera $SE(3)$ poses.

4.2 BRISK Feature Detector and Descriptor

Sparse SLAM systems revolve around keypoints (features) — carefully chosen parts of the image with the property of saliency. This enables the algorithm to track and match such part of the image (therefore part of the 3D scene) across multiple frames. This procedure is composed of two parts. First, the image is processed to detect keypoints. Usually some threshold value can be set to adjust the number of discovered keypoints. The second part is the feature descriptor which can be thought of as a hash function for images. It operates on the patch around the keypoint and reduces the dimensionality of the data while preserving essential information. It is the feature descriptors that allow us to match keypoints between images. A good feture detector should return the most promising keypoints to track and good feature descriptor should aim at good discrimination, otherwise it might lead to mismatches, while being robust to scale, rotation or illumination changes.

In this work we have chosen the BRISK [LCS11] suite of detector and descriptor. BRISK feature detection relies on AGAST ([MHB+10]) which is in turn an extension to the widely used FAST ([RD06]) feature detector. BRISK detects features at multiple levels of an image pyramid and this brings scale invariance. Detection is based on evaluating each pixel against the pixels in a circular neighbourhood around the pixel of interest. If 9 consecutive pixels in the 16 pixel circle are significantly brighter or darker than the pixel of interest then the pixel is considered a good keypoint. Additionally, BRISK estimates the rotation of the keypoint, which will be essential in the descriptor part.

The BRISK descriptor follows the idea presented in BRIEF ([CLSF10]), namely to build a binary descriptor from a series of binary intensity comparisons from a pattern around the keypoint. Contrary to BRIEF random sampling pattern the BRISK pattern has a well defined structure, as shown in Figure 4.1.

The comparisons are made after accounting for the rotation estimated earlier. Also the image undergoes Gaussian smoothing proportional to the distance from the center. The result is a binary string of length 512. Such binary strings can be easily compared and evaluated using Hamming distance — the number of positions

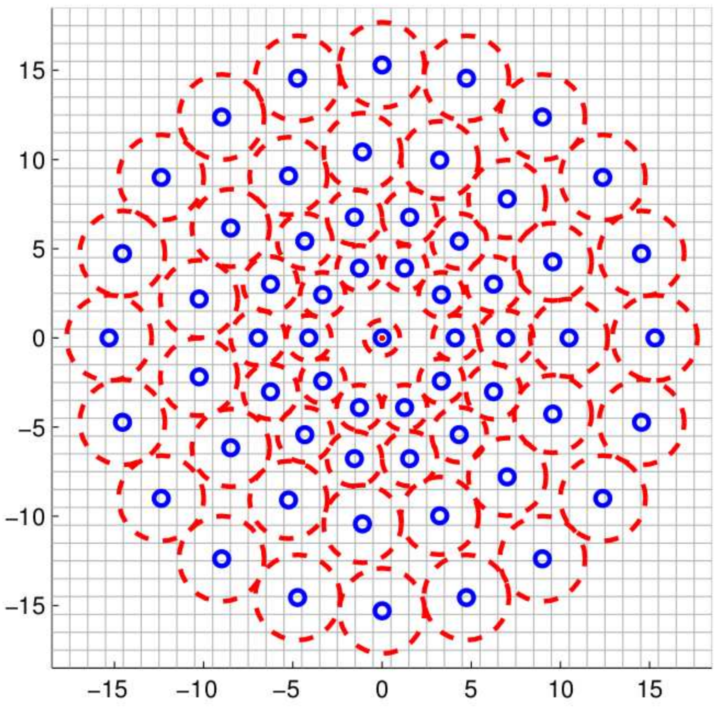


Figure 4.1: BRISK descriptor sampling pattern.

where the elements of the strings are different. Computingwise such operation is equal to the population count of an XOR of the two descriptors, a very fast operation on modern CPUs, even on 512 bit long strings. This descriptor and the associated comparison function enables matching of the features in the images.

The unanswered question here is how well common feature detectors and descriptors operate on omnidirectional images. The raw image on the sensor is circular in nature, with fewer pixels per solid angle at the inner radius and more on the outer. Readily available feature detectors and descriptors operate on planar square patches of the image, which we think might lead to suboptimal results (loosing descriptive power) when used on wide angle vision systems. This is one of the reasons we have decided to perform panoramic unwrapping in spherical coordinates, where pixels are uniformly related to the horizontal and vertical viewing angles (see Section 3.6). While this unwrapping alters the data due to interpolation it produces a rectangular image for which feature detectors and descriptors are suited to work.

4.3 Match Filtering

Our feature-based motion estimation method runs with separate tracking and mapping similarly to the PTAM [KM07] algorithm. For each new live frame, after BRISK detection and description, the tracker part is run, which matches either the entire map of landmarks or the landmarks visible in the last N keyframes to the keypoints found in the live image. Landmarks are matched based on reprojection error in the image plane and BRISK descriptor distance. With the matches ready the tracker estimates the $SE(3)$ pose for the live image. The live frame pose is initialized from the previous value or a linear motion model is applied.

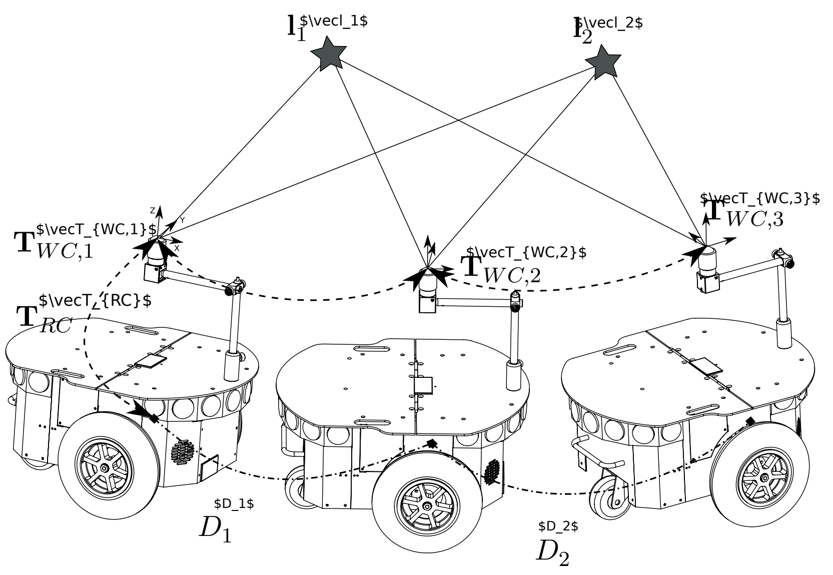
If the current pose fullfils the criteria below, we create a keyframe out of the current frame and add it to the map. In our system the criteria are:

- tracker matches fewer than 45 landmarks successfully,
- translational distance from the previous keyframe is above 25 mm,
- rotational distance from the previous keyframe is above 3°.

When the current tracked live frame is converted into a keyframe, the process of generating new landmarks can begin. This is based on the bucketing technique — we divide the image into uniform grid, either planar or radial (for example in the case of omni/fisheye image) and for each of the cells we take the M best keypoints. Now from each of these keypoints, except the ones that were already matched by the tracker in the previous step, we generate new landmarks by lifting the unit vector from the pixel location of the keypoint at the distance of 7.5 m. If a putative landmark gets one or more matches later it will be bundle adjusted to the correct depth. Bucketing ensures a good distribution of the best keypoints over the entire are of the image.

4.4 Bundle Adjustment

Bundle Adjustment uses non-linear optimisation to jointly estimate the locations of the landmarks (Figure 4.2 $\mathbf{l}_1$ and $\mathbf{l}_2$) and camera poses ($\mathbf{T}_{WC}$). The method is widely regarded a "Gold Standard" in sparse structure from motion, and offers higher accuracy per computational effort when compared to EKF filtering as described in [SMD12].



$\mathbf{l}_1$ $\vecl_1$
$\mathbf{l}_2$ $\vecl_2$
$\mathbf{T}_{WC,1}$ $\vecT_{WC,1}$
$\mathbf{T}_{WC,2}$ $\vecT_{WC,2}$
$\mathbf{T}_{WC,3}$ $\vecT_{WC,3}$
$\mathbf{T}_{RC}$ $\vecT_{RC}$
D_1 D_1
D_2 D_2

Figure 4.2: Geometry of feature-based motion estimation, showing both visual and odometry constraints.

The objective is to minimise the following energy function and find accurate estimates for camera poses $\boldsymbol{T}_{WC_j}$ and landmarks positions $\mathbf{l}_i$:

$$\min_{\boldsymbol{T}_{WC_j},\mathbf{l}_i} = \sum_{i=1}^{n}\sum_{j=1}^{m} b_{ij} \left\| \begin{bmatrix} \frac{1}{\sigma_u} & 0 \\ 0 & \frac{1}{\sigma_v} \end{bmatrix} \left(\pi \left(\boldsymbol{T}_{WC_j}\, \mathbf{l}_i \right) - \mathbf{u}_{ij} \right) \right\|_H . \tag{4.1}$$

This part imposes constraints between camera poses $\boldsymbol{T}_{WC_j}$, landmark poses $\mathbf{l}_i$ and their respective matched keypoint detections $\mathbf{u}_{ij}$. The variable b_{ij} is a binary control that reflects matched features (i.e. feature i seen in frame j or not).

This results in an error metric in pixel space and is then multiplied by the precision matrix (inverse covariance) as a means of weighting before being normalised with Huber norm (marked $\|\|_H$). The standard deviations σ_u and σ_v in the precision matrix are set to 0.5 pixels. In a purely visual formulation of the Bundle Adjustment problem the inverse covariance matrix weighting is not necessary, however we will be extending our pose graph with additional constraints, as described in Section 4.5, therefore, correct balancing of various types of error terms is necessary.

The solution is obtained by the iterative Levenberg-Marquardt algorithm [AKO] with the Jacobians calculated by means of auto-differentiation.

4.5 Additional Pose Graph Constraints

Purely visually constrained Bundle Adjustment from a monocular camera has scale ambiguity. As we operate on a robot and often have access to additional sensors we can input this information to improve our solution. In our case we use wheel odometry as an additional constraint to provide metric scale.

This is done by tying two relative camera pose estimates (tied already with landmarks with the visual reprojection errors) with a relative pose between corresponding odometry readings. In our case wheel odometry is $SE(2)$, so we promote it to $SE(3)$ to keep transforms consistent. In our diagram (Figure 4.2) these relative odometry changes are marked as D_1 and D_2. There is a pose change between the wheel odometry and the omnidirectional camera, assumed identical for every frame, marked $\boldsymbol{T}_{RC}$ on the diagram.

Therefore to calibrate both the robot to camera frame transform and keep the visual part in scale we need to minimize the $\boldsymbol{T}_{ERR}$ term:

$$\begin{aligned}
\boldsymbol{T}_{C_iC_j} &= \boldsymbol{T}_{WC_j} \otimes \boldsymbol{T}_{WC_i}^{-1}, \\
\boldsymbol{T}_{D_iD_j} &= \left(\boldsymbol{T}_{WD_j} \otimes \boldsymbol{T}_{RC}\right) \otimes \left(\boldsymbol{T}_{WD_i} \otimes \boldsymbol{T}_{RC}\right)^{-1}, \\
\boldsymbol{T}_{ERR_ij} &= \boldsymbol{T}_{D_iD_j} \otimes \boldsymbol{T}_{C_iC_j}^{-1}.
\end{aligned} \tag{4.2}$$

Simply minimizing $\boldsymbol{T}_{ERR}$ to zeros would lead to incorrect results, as the tangent space of $SE(3)$ is composed of both rotational and translational parts, each with different meaning, units and scale. Therefore, to correct for this we weight this error residual by the inverse covariance matrix:

$$\Sigma^{-1} = \begin{bmatrix}
\frac{1}{\sigma_t^2} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{\sigma_t^2} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{\sigma_t^2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{\sigma_r^2} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{\sigma_r^2} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{\sigma_r^2}
\end{bmatrix} \tag{4.3}$$

The standard deviation for this type of constraint σ_t is proportional to the square root of distance travelled and equal to 14.42 mm per 100 mm travelled. Similarly,

σ_r is set to 8.90° per 180° of rotation. These estimates were obtained by comparing the incremental odometry readings against the VICON ground truth tracker.

4.6 Camera to Laser Scanner Calibration

For evaluating free space mapping we have equipped our robotic platform with a SICK LMS500 laser scanner, as seen in Figure 4.3. Such laser scanners are widely used by the robotics research community, but rarely seen in commercial products due to prohibitive cost. The scanner however is a suitable source of ground truth, as it measures over 190° range, with less than 1° step and sub-centimere error guaranteed by the manufacturer.

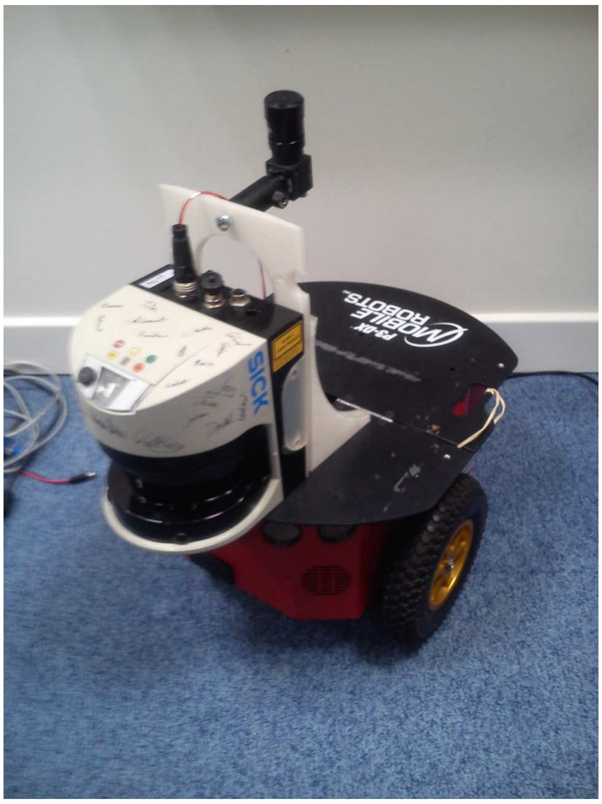


Figure 4.3: Robot equipped with a planar laser scanner.

The spatial relationship between the camera and the laser scanner had to be established to use the laser as ground truth. As the laser scanner is a 2D sensor we could only estimate an $SE(2)$ frame change, while the missing dimension Z was measured with a ruler.

The inputs to the calibration procedure are a sequence of relative pose changes. One set is the result of the Bundle Adjustment Visual Tracking procedure described above (converted to $SE(2)$) and the second one is obtained from an Iterative Closest Point algorithm run on the consecutive laser scans.

Iterative Closest Point algorithm is a classical way to align 2D or 3D point clouds. In our case the parameter of interest is the relative pose change between two 2D laser scans.

The method works iteratively as follows (also see Figure 4.4):

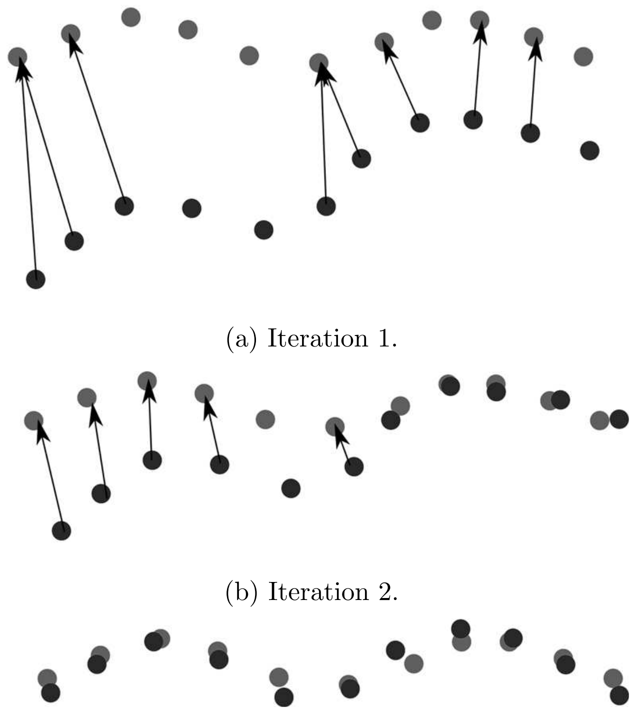
1. For each point in scan A, transformed with current $\boldsymbol{T}_{AB}$, find the closest point in scan B (this can be efficiently performed with K-D trees).
2. Formulate the least squares problem $E(\boldsymbol{T}_{AB}) = \sum_{i=1}^{N} \|{}_{A}\mathbf{x}_i - \boldsymbol{T}_{AB}\, {}_{B}\mathbf{x}_i\|^2$.
3. Solve the problem and update the current $\boldsymbol{T}_{AB}$.
4. Repeat until covergence or maximum number of iterations.

This procedure gives us relative pose changes between consecutive laser scans. By using an approach similar to the one described in 4.5, but with $SE(2)$ input data, we are able to estimate the $\boldsymbol{T}_{LC}$ pose change between the laser scanner and the omnidirectional camera.

4.7 Camera to Ground Truth Tracker Calibration

To use the semi-synthetic data generated by the Virtual Camera software described in Section 2.5 we must also calibrate the pose change between a set of VICON tracker markers on the robot and the omnidirectional camera on the robot. With this transform estimated we then can generate novel semi-synthetic views and depth maps exactly from the viewpoint where the original image was captured.

This calibration has two parts. To provide the initial guess we rely on a specially made calibration pattern (Figure 4.5), featuring both VICON reflective markers and classical chessboard pattern. Therefore from the ground truth tracker we know at the same time the pose of the robot and the pose of the patten in the tracker frame of reference. Precise laser manufacturing of the pattern, with a known geometrical


(a) Iteration 1.

(b) Iteration 2.

(c) Iteration 3.

Figure 4.4: Iterative Closest Point algorithm principle of operation.

relationship between the markers and chessboard pattern, allows us to establish the coordinates of the chessboard corners in the tracker frame of reference (within 0.1 mm–0.2 mm). With that we only need to optimize for $\boldsymbol{T}_{VC}$, the pose between the camera sensor and the set of markers on the robot.

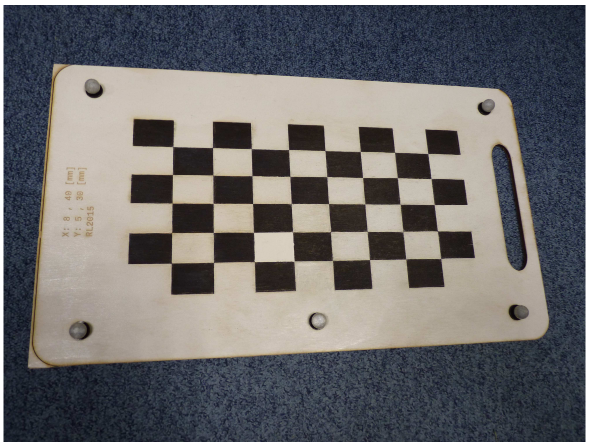


Figure 4.5: Combined vision and ground truth tracker calibration pattern.

This procedure alone provides adequate accuracy, but to refine it even more we add this pose constraint to every keyframe in our sparse SLAM system. As the ground truth tracker does not exhibit drift, as is the case for wheel odometry, we can use not just relative pose changes as in 4.5, but pose constraints between the global camera pose against the global ground truth measurement of the markers:

$$\boldsymbol{T}_{ERR} = \left(\boldsymbol{T}_{V_i} \otimes \boldsymbol{T}_{VC}\right) \otimes \boldsymbol{T}_{WC_i}^{-1}, \tag{4.4}$$

where $\boldsymbol{T}_{V_i}$ is the measurement of the marker set on the robot from the ground truth tracker, $\boldsymbol{T}_{VC}$ is the pose change between markers and the omnidirectional camera (to be refined). This transformation, similarly to the approach in Section 4.5, is assumed constant and initialized with the special calibration pattern method described above and $\boldsymbol{T}_{WC_i}$ is the pose of the i-th camera.

4.8 Experimental Results

To evaluate our omnidirectional sparse SLAM method, we have captured a dataset in a room equipped with a VICON tracker. The results of the evaluation of our omni sparse SLAM are presented in Figure 4.6 and Tables 4.1, 4.2 and 4.3. The robot performed a trajectory consisting of 5 circles of varying radius. The pose between the infrared reflective markers on the robot and the camera was estimated as described in Section 4.7.

We have followed the established methodology for the evaluation, that is comparing the error between incremental poses, as described in [KSD+09].

The VICON tracker can provide submilimeter accurate poses at 250 Hz rate. This is however valid for a static object in a well covered (by the tracker cameras) volume of the room, soon after system's calibration. In the case of a moving object, with sometimes partial visibility of the markers, one cannot expect the same level of accuracy. Another point worth mentioning is that the system slowly drifts out of calibration, especially with a diurnal cycle, as the cameras move due to the contraction and expansion of the walls. Other contributing factors are slippage of the camera mounts and vibration. Ambient light (particularly sunlight) can also influence the accuracy.

The manufacturer guarantees 1 mm/1° error with 1 mm/1° standard deviation with 1000 samples captured in a setup with 4 cameras at 3 m distance, observing a 1 m^3 volume of space. This is often not the case in our setup, as while the robot moves it does not stop to collect 1000 samples for every frame and our camera coverage is lower than recommended (10 cameras for approximately 90 m^3).

While manufacturer's guarantees tend to be on the conservative side, it is important to understand the limitations of the ground truth system used and evaluate its performance.

The Table 4.1 provides more detail on the circular test trajectories, along with overall travel error.

Circle	Frame count	Approx. radius $[mm]$	Total travel BA $[mm]$	Total travel ground truth $[mm]$	Travel error [%]
1	53	55	328.7	324.4	1.30
2	59	114	719.3	730.7	1.58
3	66	183	1184.8	1193.9	0.76
4	85	237	1542.4	1557.1	0.95
5	230	478	3136.9	3199.8	2.00

Table 4.1: Test trajectories characteristics.

Translational and rotational errors of the test trajectories are presented in the Tables 4.2 and 4.3.

Circle	Mean translation error $[mm]$	Translation error std. deviation $[mm]$
1	1.107	0.844
2	1.591	0.992
3	1.570	1.082
4	1.876	1.245
5	1.410	0.864

Table 4.2: Incremental pose error analysis (translation-magnitude).

As we can see in Tables 4.2 and 4.3 the overall errors are minimal, especially the more important ones, on the rotation. At these low levels ($[mm]$ and sub-degree) it is impossible to name the primary source of the error. The unknown part of the error comes from the feature keypoint mismatches in the SLAM system. Minor errors might be caused by manual calibration of the timestamp offsets, as sensors' clocks are not synchronized. Some errors might originate from the $\boldsymbol{T}_{VC}$ pose between VICON markers and the omnidirectional sensor (see Section 4.7). The calibration

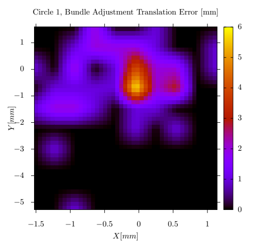
Circle	Mean rotation error [°]	Rotation error std. deviation [°]
1	0.015	0.544
2	0.005	0.558
3	0.028	0.175
4	0.007	0.158
5	0.002	0.119

Table 4.3: Incremental pose error analysis (rotation).

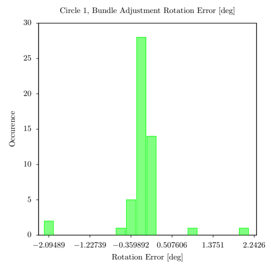
of the omnidirectional camera itself can drift by small amounts due to temperature changes (see section 3.6). In the end there are the ultimate limitations of our ground truth system as explained above.

4.9 Conclusion

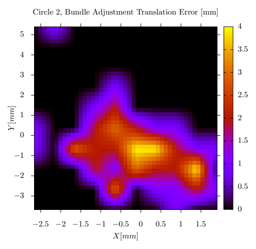
The system presented in this chapter is not a fully featured SLAM pipeline, with advanced feature matching, loop closures or keyframe management (as for example [MAMT15]), as this would be beyond the scope of this thesis. Our goal was to estimate accurate camera poses with correct scale and we conclude that the accuracy of our omni sparse SLAM, with only wheel relative odometry constraints, achieves a correct position within milimeters and single degree levels over many metre long trajectories, with overall scaling errors around 1 %–2 %. Thus it can be used in the field to estimate camera poses for further analysis, 3D reconstruction and room understanding tasks.



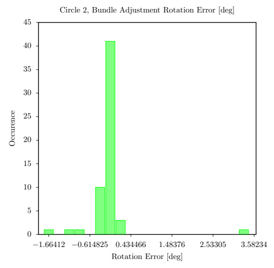
(a) Circle 1, Translation Error



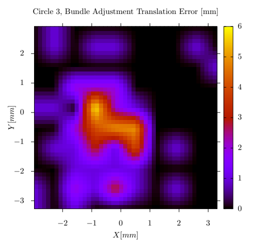
(b) Circle 1, Rotation Error



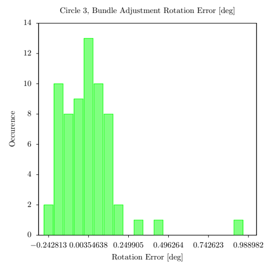
(c) Circle 2, Translation Error



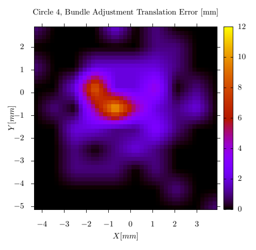
(d) Circle 2, Rotation Error



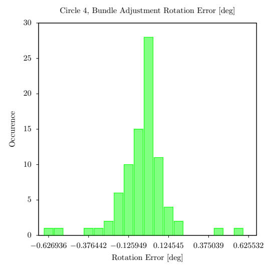
(e) Circle 3, Translation Error



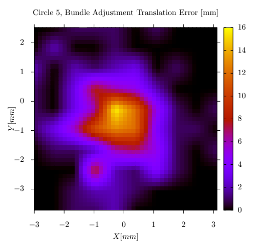
(f) Circle 3, Rotation Error



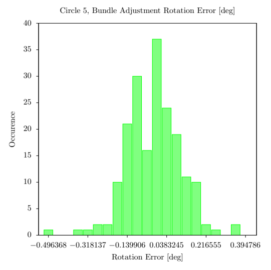
(g) Circle 4, Translation Error



(h) Circle 4, Rotation Error



(i) Circle 5, Translation Error



(j) Circle 5, Rotation Error

Figure 4.6: Omni sparse SLAM evaluation (blue), using Vicon tracker as the ground truth (red).

Chapter 5

Omni Dense Depth Map Estimation

Contents

5.1 Dense Representation **59**

5.2 Cost Volume and Plane Sweep **60**

5.3 TV-L1 Regularization **60**

5.4 Depth Standard Deviation Estimation **62**

5.4.1 Bilateral Filter 63

5.5 Experimental Results **63**

5.5.1 Synthetic Ground Truth 63

5.5.2 Semi-synthetic Ground Truth 64

5.5.3 Evaluation of the Filtering Techniques 64

5.5.4 The Impact of Trajectory on Depth Map Quality 71

5.6 Conclusion **74**

In this chapter we describe an omnidirectional dense depth map estimation technique. The ability to obtain dense depth maps is crucial in the following chapters which focus on extracting higher level information from these sets of depth estimates.

5.1 Dense Representation

It is important to first specify the rationale behind dense estimation techniques. While the localization performance of sparse method described in Chapter 4 is often adequate, the mapping part is of limited usability. Even hundreds of sparse landmarks poorly describe the surrounding environment and therefore prohibit higher level understanding.

Dense methods estimate depth for every pixel of the image. This is possible thanks to a photometric constancy assumption — i.e. the pixel values from multiple values are similar for the same point in the 3D world. The assumption on its own might seem naive and often is invalid (on specular or textureless surfaces where it is impossible to find a match) thus we have employed robust norms and developed filtering approaches that let us ignore uncertain estimates above some threshold. This leads to a semi-dense depth map and while some measurements are discarded the overall amount of depth data is vastly higher than in the sparse SLAM case.

5.2 Cost Volume and Plane Sweep

Our 3D reconstruction method, based on [NLD11], similarly to the other MVS techniques using large numbers of images, relies on the use of a “cost volume” which is a volumetric representation of space where each voxel accumulates photometric error between images (robustified by the Huber norm). This method is also known in the literature under the name “plane sweep”. The cost volume element C_r value from reference image $\mathbf{I}_r$, for pixel $\mathbf{u}$ and depth d over the set of images $\mathcal{I}(r)$ is defined as:

$$C_r\left(\mathbf{u}, d\right) = \frac{1}{c} \sum_{m \in \mathcal{I}(r)} \left\| \rho_r \left(\mathbf{I}_m, \mathbf{u}, d\right) \right\|_H , \tag{5.1}$$

with c being the number of successful reprojections. The two view photometric error is defined as:

$$\rho_r\left(\mathbf{I}_m, \mathbf{u}, d\right) = \mathbf{I}_r\left(\mathbf{u}\right) - \mathbf{I}_m\left(\pi\left(\boldsymbol{T}_{WC,m} \otimes \left(\boldsymbol{T}_{WC,r}^{-1}\, \pi^{-1}\left(\mathbf{u}, d\right)\right)\right)\right) . \tag{5.2}$$

Each row of photometric errors at pixel $\mathbf{u}$ can be then searched to find the minimal photometric error and therefore its depth. Alas, such depth estimates, even when incorporating a subpixel interpolation step, tend to be very noisy.

5.3 TV-L1 Regularization

As in the original paper [NLD11], on which we have based our method, we have also implemented TV-L1 depth map regularization. This method is based around

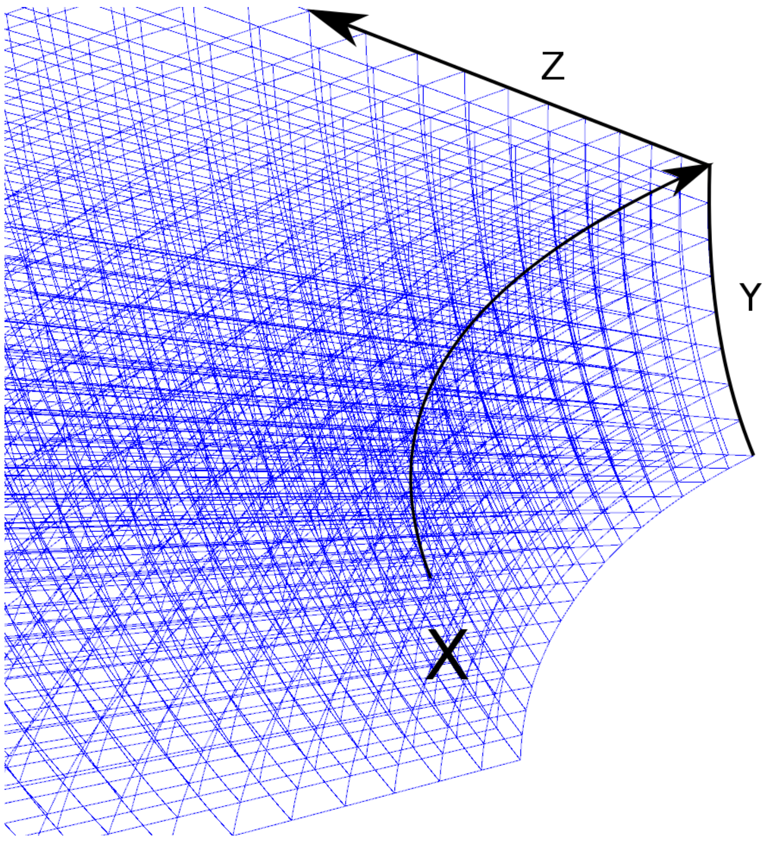


Figure 5.1: Cross section of the omnidirectional cost volume space discretization.

the Primal-Dual optimization technique, well described in [HNAD11] and the works of Chambolle and Pock [CP11].

$$g(\mathbf{u}) = e^{-\alpha \|\nabla \mathbf{I}_r(\mathbf{u})\|_2^{\beta}} \tag{5.3}$$

$$E_{\xi,\alpha} = \int_{\Omega} \left\{ g(\mathbf{u}) \left\| \nabla \xi(\mathbf{u}) \right\|_{\epsilon} + \frac{1}{2\theta} \left(\xi(\mathbf{u}) - \alpha(\mathbf{u}) \right)^2 + \lambda C \left(\mathbf{u}, \alpha(\mathbf{u}) \right) \right\} \tag{5.4}$$

The method has a considerable number of parameters, therefore a lot of time is spent in tuning. In our application, 3D reconstruction on a mobile robot, we tuned the regularizer to absolutely remove highly erroneous measurements, either protruding into the free space or extending into the maximum defined depth range. Such estimates could cripple the operation of the robot, if these measurements dictate the planning behaviour (e.g. it would assume there is no free space to move at all). Usually this regularizer is used to remove salt & pepper type of noise and provide asthetically pleasing 3D reconstruction, instead of a practical one. Conservative tuning in our setup leads to extreme oversmoothing of the resulting depth map and we believe this is of limited use, as it distorts the surrounding environment and while the robot could navigate such an environment it would often bump into obstacles.

That led us to develop alternative approaches to filtering the raw depth maps, as described in Section 5.4.

Example result of TV-L1 filtering is presented in the Figure 5.8d.

5.4 Depth Standard Deviation Estimation

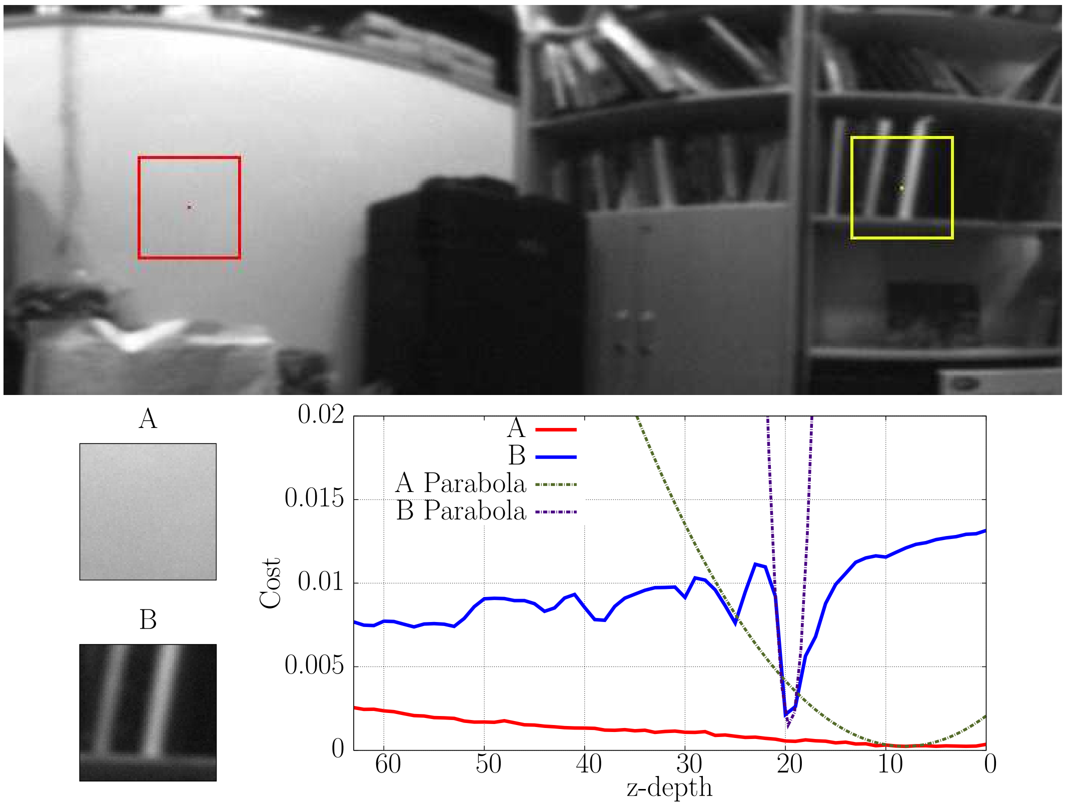


Figure 5.2: Depth standard deviation estimation. Patch (A) has low texture and therefore poor depth estimate quality, with patch (B) the opposite. Fitting a parabola provides us with a subpixel depth estimate as well as depth standard deviation.

As a substantial proportion of the 3D reconstructed area has poor depth estimates we need a way to decide which measurements are trustworthy. Our method relies on estimating depth standard deviation, as illustrated in Figure 5.2. We show two extreme cases; (A): a low texture area in the middle of a wall, and (B): a high texture area, with contrasting energy responses in their respective cost volume depth sampling. By fitting a parabola to the minimum we can estimate the standard deviation of the depth estimate. In addition, fitting a parabola provides subpixel depth resolution. In Figure 5.2 parabolae for (A) and (B) have vastly different a parameters which determine the standard deviation in the inverse depth domain: $\sigma_{\varepsilon}(\mathbf{u}) = \frac{1}{\sqrt{2a}}$. This is then converted into standard deviation in the depth domain as follows:

$$\sigma_d(\mathbf{u}) = \frac{\sigma_{\varepsilon}(\mathbf{u})}{d(\mathbf{u})^2}, \tag{5.5}$$

where $d(\mathbf{u})$ is the subpixel depth estimate for pixel $\mathbf{u}$.

Thresholding in the depth standard deviation domain is superior to other heuristics, as demonstrated in Section 5.5.1. The resultant depth measurements are not now fully dense, but much more reliable for free space inference.

In our implementation the depths are represented in inverse form and the depth range is sampled into 64 bins.

5.4.1 Bilateral Filter

We found the regularization method presented in [NLD11] suboptimal for mobile robot tasks. Our thresholding method (Section 5.4) is effective against outliers, but does not provide any kind of smoothness prior. Therefore, we employ cost volume filtering presented in [YK05], which is essentially a bilateral filter applied to every slice of the cost volume. Other depth map filtering methods are available, for example Guided Filter [HST10] that has better performance than Bilateral Filter.

5.5 Experimental Results

We have evaluated our method with both synthetic data and semi-synthetic ground truth data.

5.5.1 Synthetic Ground Truth

We have performed experiments using synthetic omnidirectional image data to investigate the performance of omnidirectional dense and semi-dense reconstruction using circular motions as in our real experiments. We have generated photorealistic data using the open source PoVRay ray-tracing software by re-rendering the data of Handa *et al.* [HNAD12] using an omnidirectional camera model, and a circular camera trajectory with 30 evenly sampled poses. We construct depth estimates from a multi-view stereo cost volume generated using all 30 images and ground truth camera poses.

To show the properties of both dense mapping algorithms in various scenes, as well as the contribution of the thresholding method described in Section 5.4, we have created two datasets, one more realistic — called textureless, due to plan white walls, and one extremely textured with vivid wallpaper on the walls.

5.5.2 Semi-synthetic Ground Truth

In addition to fully synthetic datasets obtained from a ray tracer, we have prepared a semi-synthetic dataset, based on our Virtual Camera and VICON tracker system, as described in Section 2.5. With a metrically accurate full 3D room model we can generate images and, more importantly, depth maps from the exact locations from which the real omni directional camera captured the images along robot's trajectory. While this is not fully realistic, as the room was first captured into a 3D model and then rendered and warped into another camera model image/depth, it brings us much closer to reality. Having accurate depth maps is key in this section.

5.5.3 Evaluation of the Filtering Techniques

In these evaluations we analyze full depth maps, without removing the floor or ceiling areas or over/under-exposed areas. All possible filtering combinations were tested, such as:

- Raw depth map (per pixel max. photo-consistency depth),
- Bilateral Filtering of the cost volume (see Section 5.4.1),
- TV-L1 depth map filtering (see Section 5.3),
- Bilateral Filtering of the cost volume followed by TV-L1 depth map filtering,
- Raw depth map followed by thresholding to leave 75 % best estimates (see Section 5.4),
- Bilateral Filtering of the cost volume followed by thresholding to leave 75 % best estimates,

- TV-L1 depth map filtering followed by thresholding to leave 75 % best estimates,
- Bilateral Filtering of the cost volume followed by TV-L1 depth map filtering followed by thresholding to leave 75 % best estimates.

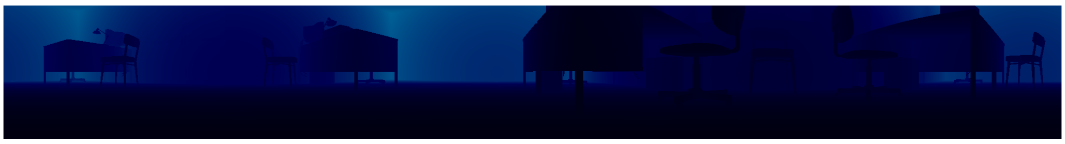
Synthetic, Textureless Dataset

Figure 5.3: Depth map quality with various filtering techniques. Synthetic, textureless dataset.

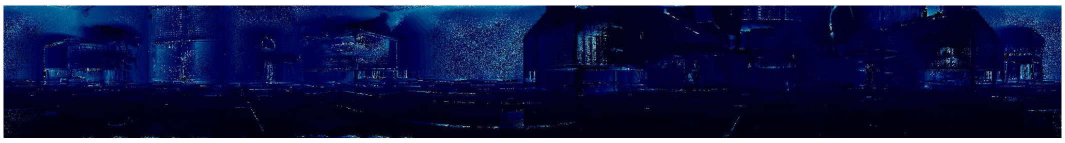
This dataset, while clearly showing the value of our filtering method, favours the raw depth maps, both on the depth error distribution and in our metric. Alas, simple per pixel depth errors do not describe the smoothness we sought. We think the strange results in this and 5.5.3 datasets are caused by the purely synthetic nature of the images. Raytracer does not add any kind of noise and might introduce staircasing effects. It is also important to note that the dimensions of these room are 8×5m, which is on the extreme end of the size spectrum for our applications.

Circle	Frame count	Valid area [%]	Area with abs. error under 200[mm] over valid area [%]
Raw	29	100.0	50.4
BF	29	100.0	49.3
TVL1	29	100.0	42.7
BF+TVL1	29	100.0	41.0
Raw+75%	29	75.0	60.3
BF+75%	29	75.0	59.3
TVL1+75%	29	75.0	52.2
BF+TVL1+75%	29	75.0	50.7

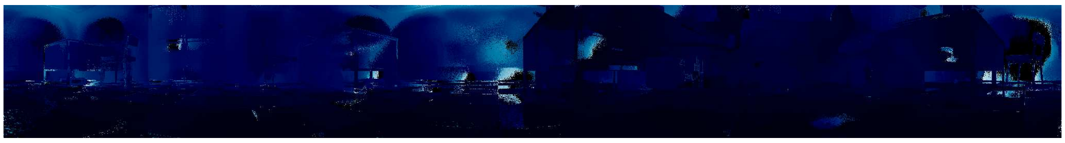
Table 5.1: Depth map quality with various filtering techniques. Synthetic, textureless dataset.



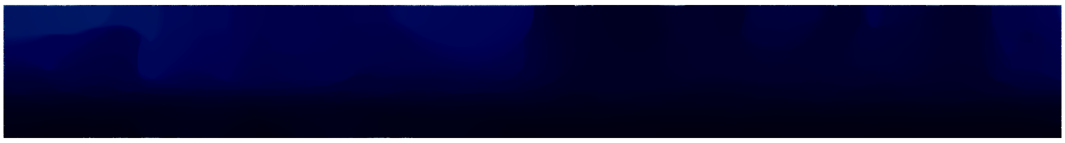
(a) Ground Truth



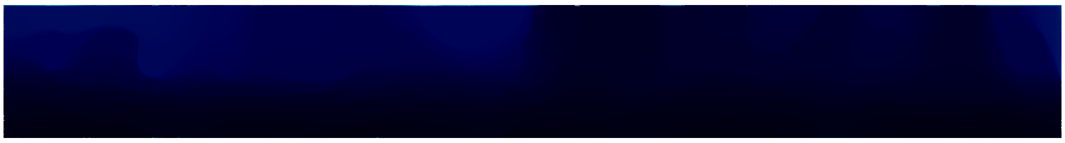
(b) Raw



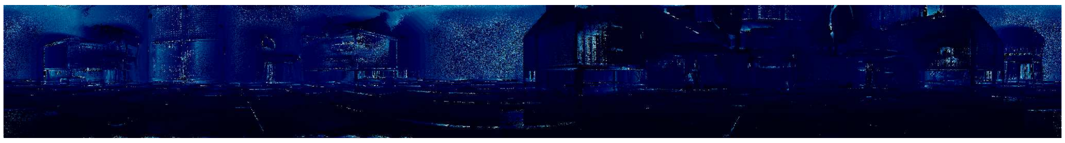
(c) BF



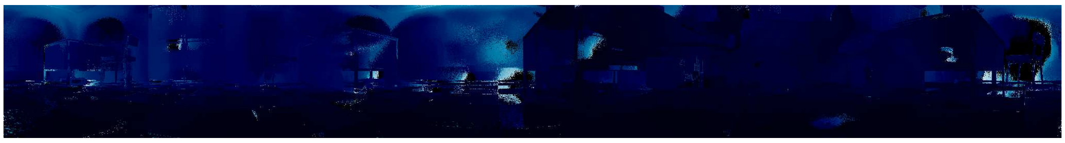
(d) TVL1



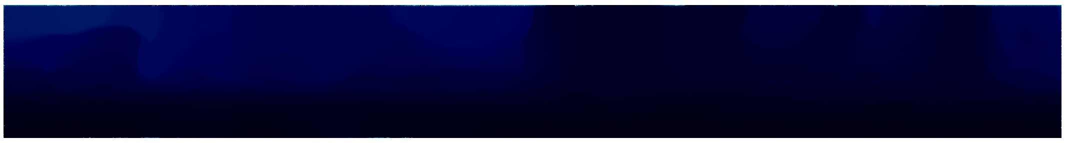
(e) BF+TVL1



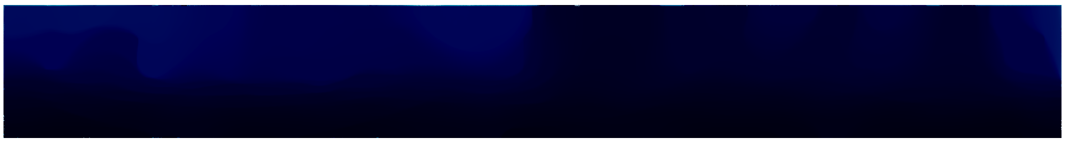
(f) Raw+75%



(g) BF+75%



(h) TVL1+75%



(i) BF+TVL1+75%

Figure 5.4: Depth maps obtained with various filtering techniques. Synthetic, textureless dataset.

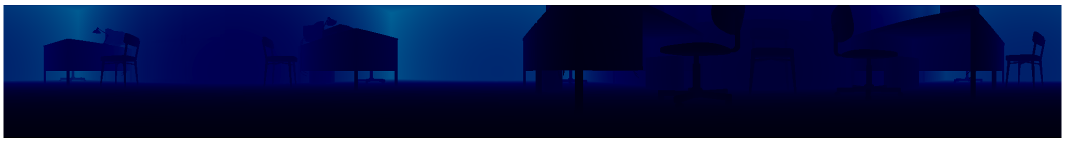
Synthetic, Textured Dataset

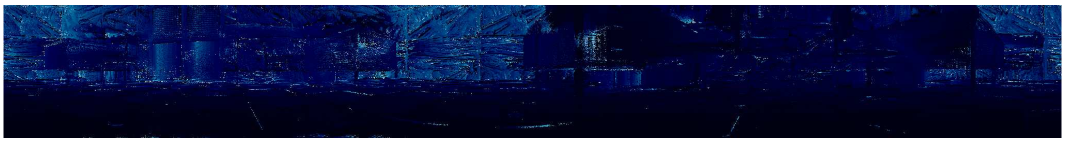
Figure 5.5: Depth map quality with various filtering techniques. Synthetic, textured dataset.

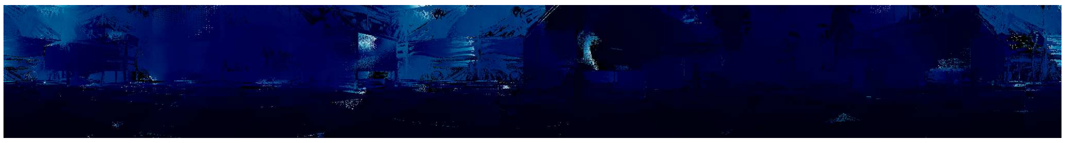
Simlilarly to the dataset 5.5.3, this one as well produced rather unexpected results, but it being more textured the overall depth map quality is slightly higher.

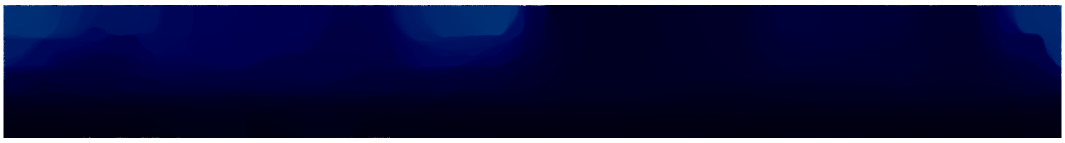
Circle	Frame count	Valid area [%]	Area with abs. error under 200[mm] over valid area [%]
Raw	29	100.0	54.9
BF	29	100.0	57.0
TVL1	29	100.0	44.1
BF+TVL1	29	100.0	41.2
Raw+75%	29	75.0	63.6
BF+75%	29	75.0	64.8
TVL1+75%	29	75.0	52.2
BF+TVL1+75%	29	75.0	50.1

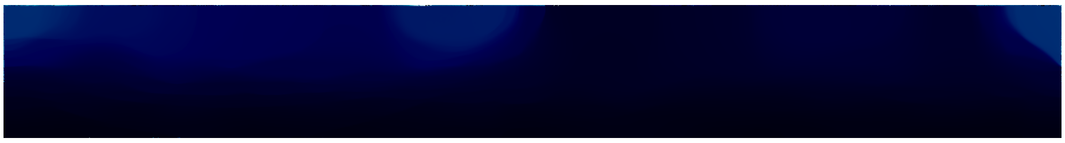
Table 5.2: Depth map quality with various filtering techniques. Synthetic, textured dataset.

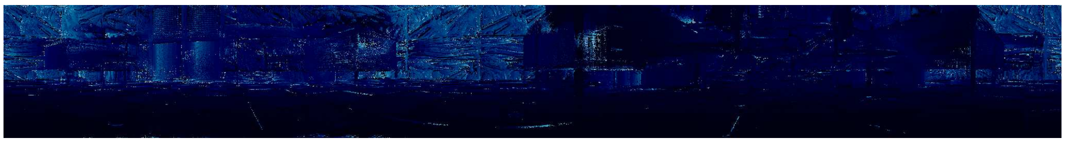

(a) Ground Truth

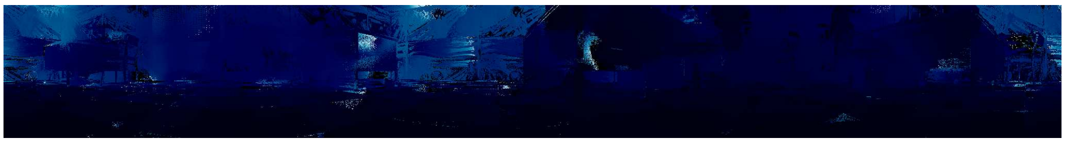

(b) Raw

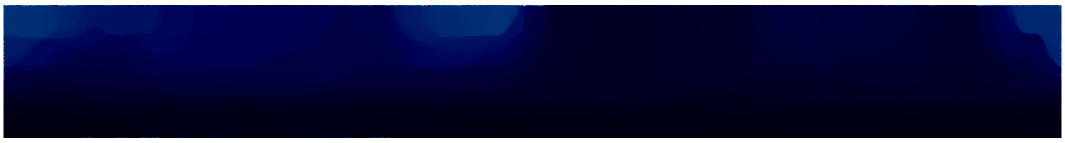

(c) BF


(d) TVL1


(e) BF+TVL1


(f) Raw+75%


(g) BF+75%


(h) TVL1+75%

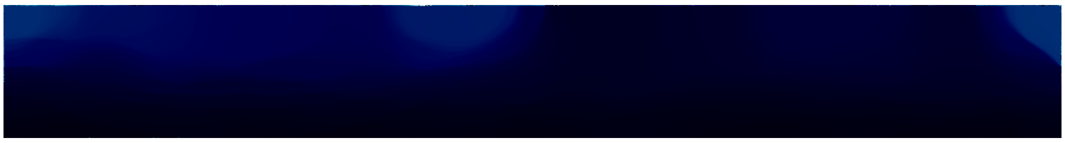

(i) BF+TVL1+75%

Figure 5.6: Depth maps obtained with various filtering techniques. Synthetic, textured dataset.

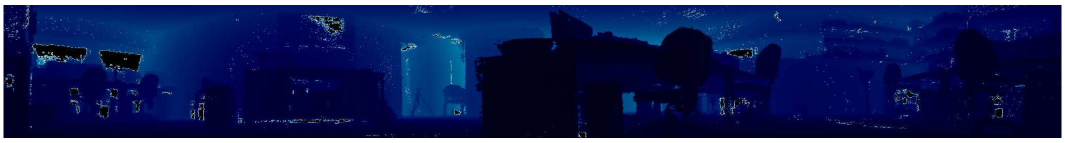
Semi-Synthetic Dataset

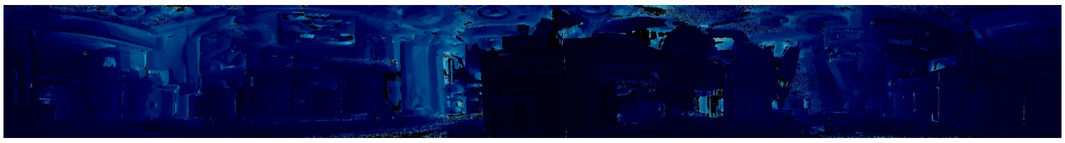
Figure 5.7: Depth map quality with various filtering techniques. Semi-Synthetic dataset.

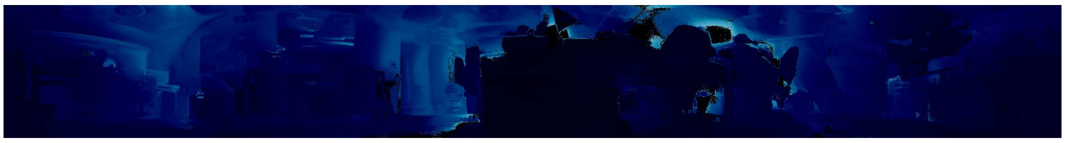
Here, contrary to the previous datasets from a ray tracer, we have a semi-synthetic dataset created from real world data, with all inherent noise and more reasonable scale of the environment. Here we can clearly see the benefit of bilateral filtering, especially when combined with our filtering method. Here again the numbers unfortunately do not reflect the information on the smoothness of the estimate.

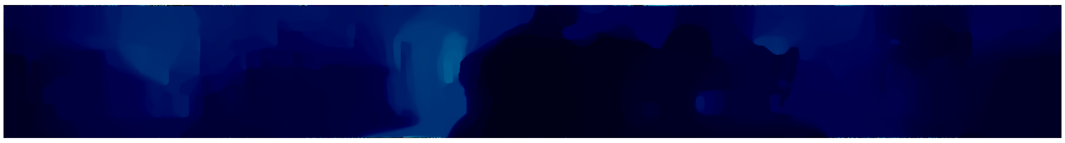
Circle	Frame count	Valid area [%]	Area with abs. error under 200[mm] over valid area [%]
Raw	65	100.0	35.8
BF	65	100.0	39.3
TVL1	65	100.0	38.6
BF+TVL1	65	100.0	39.5
Raw+75%	29	75.0	40.9
BF+75%	65	75.0	44.25
TVL1+75%	65	75.0	43.9
BF+TVL1+75%	65	75.0	46.9

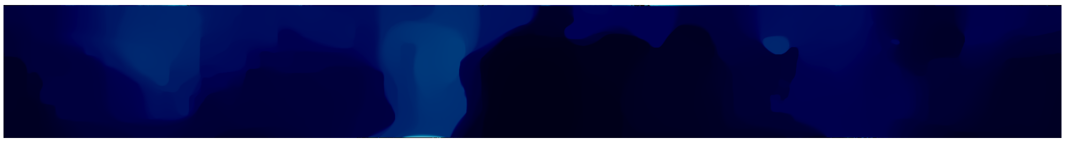
Table 5.3: Depth map quality with various filtering techniques. Semi-Synthetic dataset.

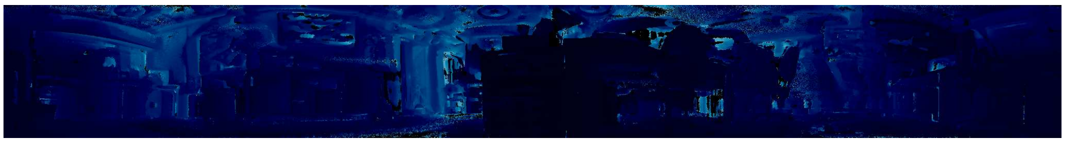

(a) Ground Truth

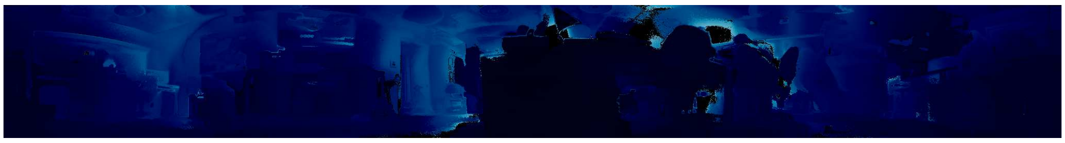

(b) Raw

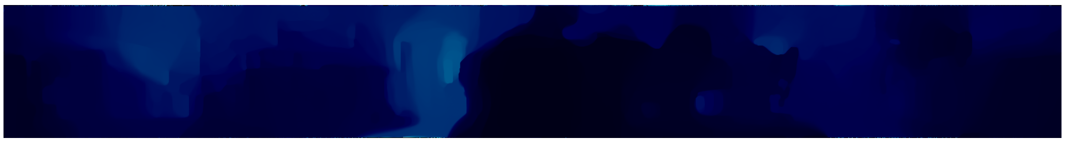

(c) BF


(d) TVL1


(e) BF+TVL1


(f) Raw+75%


(g) BF+75%


(h) TVL1+75%

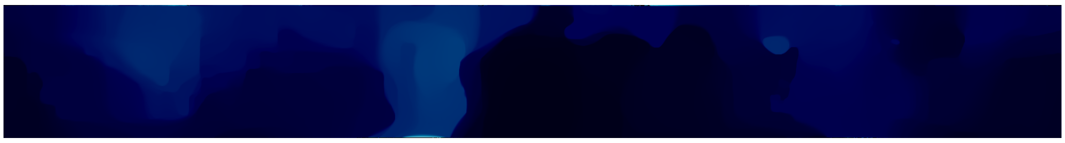

(i) BF+TVL1+75%

Figure 5.8: Depth maps obtained with various filtering techniques. Semi-Synthetic dataset.

5.5.4 The Impact of Trajectory on Depth Map Quality

In addition to the experimental evaluation of the filtering techniques, we have performed experiments to analyze depth map quality depending on the motion undergone by the robot. Our main test scenarios are:

- circular movement of varying diameter,
- circular movement, fixed dimeter, variable number of frames integrated into the cost volume,
- linear movement, fixed distance, variable number of frames integrated into the cost volume.

Circles of Varying Diameter

In this test the robot drove along 5 circles of varying diameter. More information can be found in Table 4.1. In each case the reference frame was set to point in an approximately similar direction. After filling the cost volume the bilateral filter was applied, without the thresholding step described in 5.4. Areas below floor and ceiling planes, as well as underexposed or overexposed areas, were masked to be removed from evaluation.

The depth map error histograms use the absolute depth error, defined as:

$$\mathbf{d}_{err}\left(x,y\right) = \left|\mathbf{d}_{es}\left(x,y\right) - \mathbf{d}_{gt}\left(x,y\right)\right|, \tag{5.6}$$

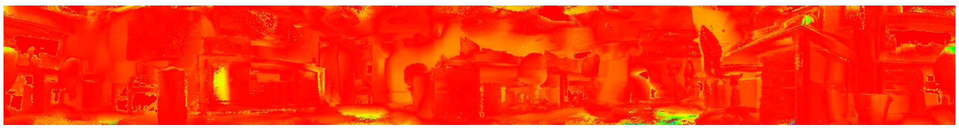
where $\mathbf{d}_{es}$ is the estimated depth map, $\mathbf{d}_{gt}$ is the ground truth depth map and $\mathbf{d}_{err}$ is the resulting absolute error.

While the metric in the Table 5.4 is inconclusive, the histogram plots in Figure 5.9 show that larger circular motions are beneficial, due to more significant baselines, aiding stereo reconstruction. In our case even the biggest circle (1 m diameter) is probably not big enough, compared to the scale of the room (6×6m) for the occlusion problems to manifest themselves. There are, however, practical advantages to the circles of low diameter. Faster to execute, unlikely to crash into obstacles and no occlusions.

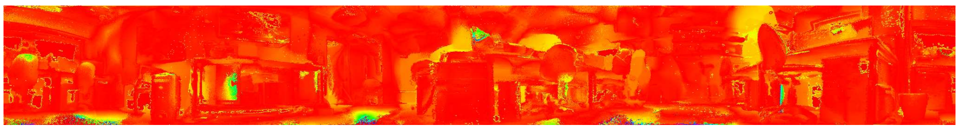
Figure 5.9: Circles.

Circle	Frame count	Approx. radius $[mm]$	Valid area [%]	Area with abs. error under 200$[mm]$ over valid area [%]
1	52	55	100.0	39.1
2	57	114	100.0	43.4
3	61	183	100.0	30.8
4	56	237	100.0	42.4
5	65	478	100.0	39.4

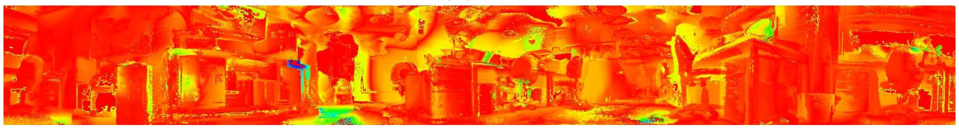
Table 5.4: Depth map quality of circular trajectories.



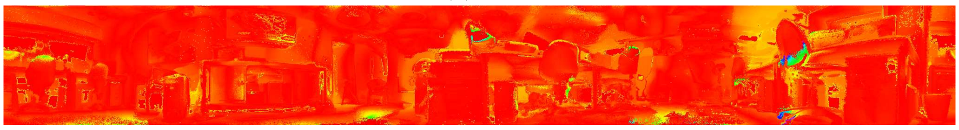
(a) C1



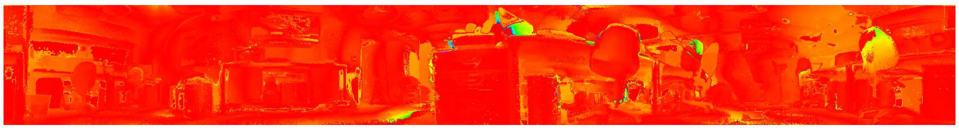
(b) C2



(c) C3



(d) C4



(e) C5

Figure 5.10: Depth map errors for circular trajectories of varying radius.

Varying Number of Frames over Circular Motion

In this test we employ the trajectory used for circle 5 (see Table 5.4), but vary the number of frames by skipping every 1, 2, 3 or 4 frames. Other parameters are as in Section 5.5.4, and the depth map error histograms are computed as in 5.6.

Figure 5.11: Circular motion with frames skipped.

Circle	Frame count	Approx. radius $[mm]$	Valid area [%]	Area with abs. error under $200[mm]$ over valid area [%]
5/0	65	478	100.0	39.4
5/1	32	478	100.0	39.4
5/2	21	478	100.0	39.5
5/3	16	478	100.0	39.1
5/4	14	478	100.0	39.7
5/5	10	478	100.0	39.3
5/6	10	478	100.0	39.1
5/7	9	478	100.0	39.5
5/8	8	478	100.0	39.3
5/9	7	478	100.0	38.8
5/10	6	478	100.0	37.1
5/16	4	478	100.0	37.9
5/32	2	478	100.0	35.3

Table 5.5: Depth map quality of variable number of frames over a circular motion.

Interestingly, we see that there is almost no relationship between the number of integrated frames on a given trajectory. This might be caused by multiple factors. First and foremost, the robot moves on a plane, thus there is no stereo baseline along the vertical axis. Therefore one cannot expect good reconstruction of horizontal features of the scene, regardless of how many views will be integrated into the cost volume. The second reason might be the limited resolution of the camera (with respect to the extreme viewing angles) combined with a large scale scene. At some distance from the camera the increments along the epipolar curve become subpixel and infinitesimal and may be a limiting factor for depth map quality, again, regardless of the number of frames integrated.

Varying Number of Frames over Linear Motion

In this test we employ a linear trajectory and vary the number of frames by skipping every 1, 2, 3 or 4 frames. Other parameters are as in Section 5.5.4.

Figure 5.12: Linear motion with frames skipped.

Line	Frame count	Approx. distance $[mm]$	Valid area [%]	Area with abs. error under 200$[mm]$ [%]
L/0	67	3250.91	100.0	51.9
L/1	33	3250.91	100.0	51.4
L/2	23	3250.91	100.0	51.5
L/3	16	3250.91	100.0	50.5
L/4	14	3250.91	100.0	51.5
L/5	12	3250.91	100.0	50.5
L/6	10	3250.91	100.0	50.5
L/7	8	3250.91	100.0	49.66
L/8	8	3250.91	100.0	49.9
L/9	7	3250.91	100.0	48.6
L/10	7	3250.91	100.0	48.9
L/15	4	3250.91	100.0	43.7
L/20	3	3250.91	100.0	36.6

Table 5.6: Depth map quality of variable number of frames over a linear motion.

In this case the depth map quality also seems relatively independent of the number of frames integrated into the cost volume, but the effect is not as small as in the circular case (Table 5.5).

5.6 Conclusion

The dense mapping pipeline was a big milestone for this thesis, as it enables the higher level methods presented in the following chapter and, more importantly, visibly shows the limitations imposed by the camera type and the setup. The most important contribution of this chapter is a very comprehensive evaluation of the method, missing from the original paper [NLD11]. The experimental data reveals some very interesting traits of the method in our particular application. Larger baselines improve the reconstruction quality (see Section 5.5.4), which is to be expected. However, the conventional wisdom states the more data the better, so the reader might find the results in Sections 5.5.4 and 5.5.4 surprising - skipping even a significant number of frames does not degrade the reconstruction quality. The depth map quality is constrained by the optical performance of the lens and camera, with its limited resolution and dynamic range, as well as the type of application we are aiming at. A mobile robot, moving largely on a plane, through relatively large scale, often textureless or specular, environments presents an unique challenge to the 3D reconstruction tasks. Therefore, the information is already lost in the optics/sensor and thus integrating more frames does not improve the estimation. This outcome is very promising. As our dense multi-view stereo is quite computationally expensive (exhaustively sampling over the entire cost volume), we can save noticeable computing expense by integrating 10 instead of 60 frames for example. This is of vital importance for mobile low power platforms.

Chapter 6

Room Understanding

Contents

6.1 Introduction **77**

6.2 Free Space Mapping **78**

6.2.1 Occupancy Grid Map Representation 78

6.2.2 Free Space Inference 78

6.2.3 Experimental Results 80

6.3 Parameterized Model Fitting **108**

6.3.1 Room Model 108

6.3.2 Differentiable Renderer 109

6.3.3 Incremental Room Layout Recovery 110

6.3.4 Experimental Results 111

6.4 Conclusions **134**

6.1 Introduction

The focus of this chapter is on room understanding, moving away from SLAM only methods described in the previous chapters (sparse and dense). The most important input to the methods described below is the omnidirectional depth map, obtained with the method described in Chapter 5. This depth map alone is of limited use for higher level understanding - more abstract representations are required. In Section 6.2 we present a method of estimating occupancy grid maps from omnidirectional depth data. While it is not very high level representation, it is widely used and extremely useful for possible subsequent robotic tasks, like path planning and active exploration. Section 6.3 presents a truly high level representation - cuboid of a room. This is the first step of room understanding and may enable the user to extract even more information (see Section 6.3.3), especially if the method is to be coupled with active exploration.

6.2 Free Space Mapping

Occupancy grids are one of the most successful and established map representations in mobile robotics. In this chapter we will focus on this application of omnidirectional depth maps estimated as described in Chapter 5. From the omnidirectional data we will be able to generate an occupancy grid representation and update it incrementally as the robot explores the environment. This chapter concludes with evaluation of the method against laser scanner and semi-synthetic ground truth.

6.2.1 Occupancy Grid Map Representation

The occupancy grid representation was first introduced by [Elf89] as a means to integrate uncertain sensor readings into a common spatial and probabilistic representation and then infer a more robust model of the world from the map. While in this thesis we do not focus on the path planning or decision making that occupancy grids enable, we understand the importance of the occupancy grid representation for sensor integration (in our case depth maps) and the adequateness of the representation for an indoor mobile robot moving on a plane. Most robots that are equipped with adequate sensing use occupancy grids in one form or another. Dyson 360 Eye is no exception. However, currently, the omnidirectional vision provides only tracking, so to build the occupancy grid map the robot has to rely on short range infrared sensors and actually explore the environment. The work presented in this chapter is aimed at significantly improving the state of affairs in this respect, without adding any additional sensors. The idea is to rapidly uncover many square meters of free space that would enable high level planning or room recognition, not necessarily very detailed reconstruction of the free space.

6.2.2 Free Space Inference

With a semi-dense depth map we can now infer the amount of free space area around the reference frame camera pose. Here having spherical panoramic unwrapping is very convenient, as each column of the depth map represents a different viewing angle around the 360° field of view. Thus, for each column we need to find the closest valid depth measurement. Since we are trying to find drivable free space for a small robot, we start by looking for depth measurements in a column below the horizon row (to deal with the case where there might be free space under a table or other overhanging furniture). If no valid measurements are found then we examine the rest of the column (above the horizon), on the assumption that some vertical walls might be blank at camera height but have useful texture higher up. If no depth estimates survive standard deviation thresholding in an entire column then we assume that for this particular viewing direction the free space boundary is at ε_{max} (usually 50 cm), a fail safe measure. This area is not truly measured but the safe passage of the width of the robot plus some margin usually provided by local sensors means that we can remove it from our occupancy maps.

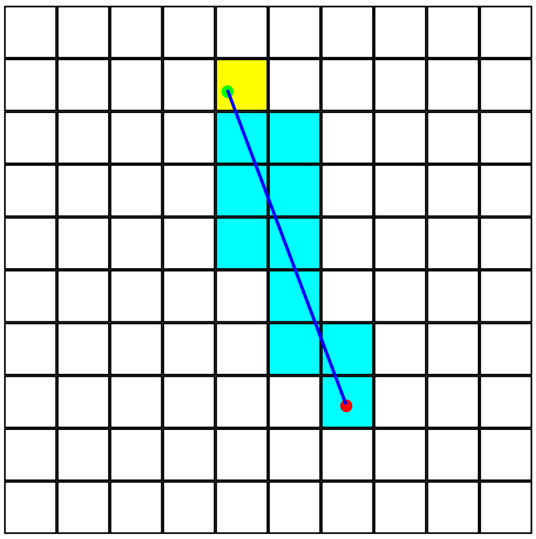


Figure 6.1: Free space recovery. Principle of operation.

The closest depth estimate from each viewing direction is considered a boundary point for the free space region (green point in Figure 6.1). Any cell marked as such boundary (yellow in Figure 6.1) takes precedence over whatever free space evidence might appear there due to noise. From each viewing direction, a line of sight ray is cast, from the reference frame pose (red point in Figure 6.1) to the boundary point and the Liang—Barsky [LB84] intersection test is performed against all the cells in the occupancy grid. If the test is positive the free space counter for such cell is incremented (blue cells in Figure 6.1). Integrating hundreds of free space boundary points from a single depth map and later integrating multiple depth maps, spaced across the room and processed in the same manner, generates a global free space map in the occupancy grid. We could employ the more orthodox Bayesian approach as presented in [TBF05], as we have measurement uncertainties obtained from the method described in the Section 5.4, but we found even the simple brute force approach gives satisfactory results (even without any particular thresholding, treating every non-zero free space cell as free space). This is mainly due to the post-processing and filtering steps in the dense reconstruction pipeline that result in higher quality depth maps. We believe that it is more beneficial to focus on the depth map quality, as it can be used in other methods and applications, than to filter the occupancy grid representation afterwards.

We use an occupancy map cell size of 10 cm in all experiments. This size is based on common sense, as our robotic platforms are roughly 40 cm in diameter, so the resolution should only be good enough to be able to tell if the robot can drive through a passage or not.

6.2.3 Experimental Results

Laser Scanner Ground Truth

In the three cases presented below our robot was equipped with a SICK LMS500 laser scanner, with its extrinsics calibrated as described in Section 4.6. While this is a superior source of ground truth with manufacturer's guaranteed high precision and accuracy, we found it suboptimal for evaluating our approach. The laser scanner shines a narrow laser beam via a rotating mirror, therefore it reaches through chairs, under tables and other similar obstacles, while our visual method is much more coarse (and passive). In effect the laser scanner ground truth occupancy grid does not represent truly the drivable area of the room.

Cluttered Office This is a dataset in an office room, with the elementary motion of a 0.5 m diameter circle. The robot captured 126 frames regularly spaced around the circle which were bundle adjusted to estimate accurate poses. From 3 reference poses the depth and depth standard deviation estimation procedures were performed to create 3 dense depth maps with their respective standard deviation maps; Figure 6.2 shows one depth map result.

After thresholding the depth standard deviation maps and estimating the free space boundary from the semi-dense map, each boundary was integrated into the

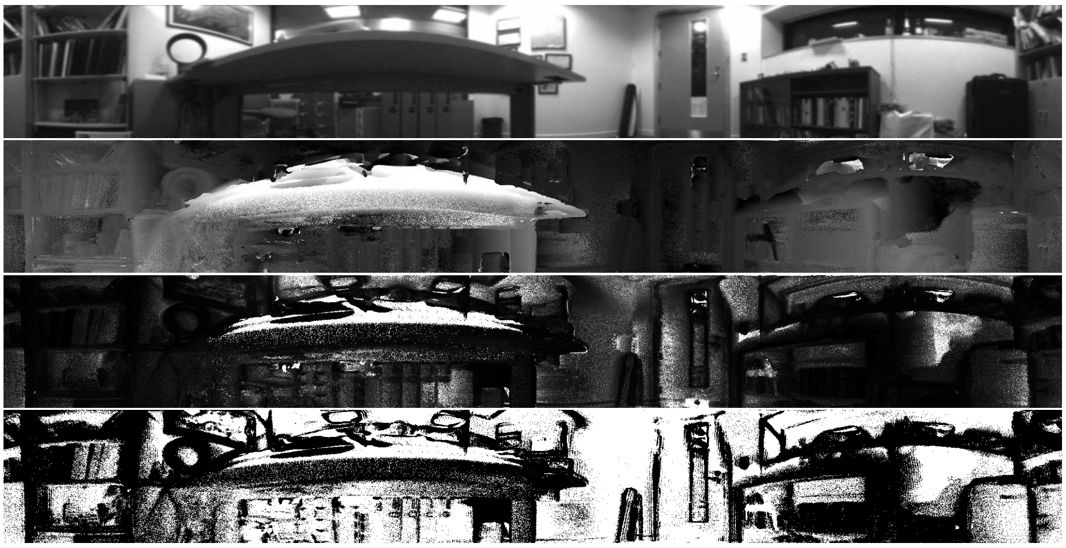
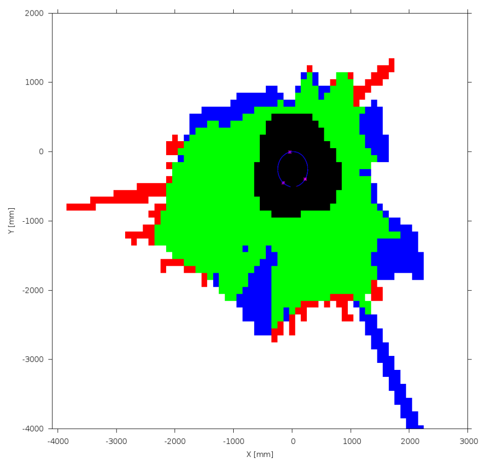


Figure 6.2: Dataset “Cluttered Office”, from top to bottom: reference frame, estimated depth Map, estimated depth standard deviation map, depth threshold mask.

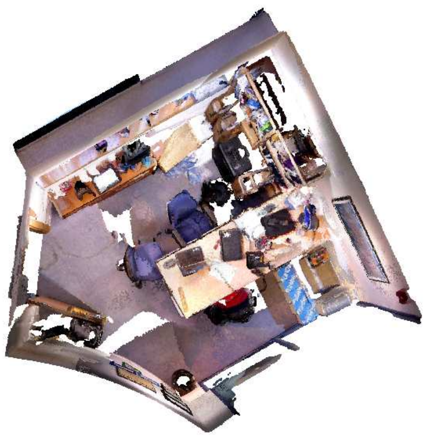
occupancy grid shown in Figure 6.3a (to be compared with the visualisation in 6.3b). We highlight the green area, where the free space recovered from omnidirectional vision agrees with LADAR. The single circle motion, without any exploration and taking only a few seconds, uncovers over 80 % of the room’s floor area.

Empty Office The second example we present is from an empty and newly painted office. This room is interesting because it clearly demonstrates the weaknesses of a passive stereo method in challenging, highly untextured scenes. As before, the robot drove in a 0.5 m diameter circle and 3 depth map and free space measurements were estimated. Each one contributed to the occupancy grid, as seen in Figure 6.4.

As before, we draw attention to the green area in Figure 6.5a, which is the free space area mapped by omnidirectional vision which agrees with ladar. In this scenario the comparison between ground truth laser sensor measurements and our vision system are more meaningful, as there is almost no furniture and both sensors observe similar obstacles. However, due to the low texture conditions there are larger blue areas where meaningful depth measurements have not been made. Still, even in such unfavourable conditions for the vision system we are able to uncover 50 % of the free space in the room with a rapid circular motion.



(a) "Cluttered Office": occupancy grid from one circular motion and three reference images, comparing our vision-based estimate with LADAR mapping. Green: vision and laser agree; blue: found by laser but not by vision; red: found by vision but not by laser; black: area close to robot known to be empty.



(b) "Cluttered Office": top-down view of the room for comparison (screenshot from an RGBD 3D SLAM system).

Figure 6.3

Incremental Mapping in a Large Office Environment This dataset is of a much larger scale than the ones described in the previous subsections. It illustrates how a full free space map of a room can be built incrementally. The robot moves in a 1 m diameter circle, performs all the steps of the method (i.e. estimates three depth maps and integrates free space estimates into a global occupancy grid) three times, it then traverses a short distance to another location and makes another circle. The whole trajectory is globally bundle adjusted for globally consistent poses. This means that the free space information recovered at each circle location can be fused incrementally to eventually cover almost the entire room. This is repeated 4 times across the room and each and every step uncovers more of the free space area, finally reaching almost the entire room, as can be seen in Figure 6.6f.

In this example it might appear that only 50 % of the free space area was discovered (as compared to the laser), but as mentioned in the Section 6.2.3 the laser scanner, due to its sensing characteristics and placement on the robot with respect to the camera, shines through chairs, under the desks and sometimes through the doorframe, heavily distorting actual drive-able free space area. Rather, we would like to

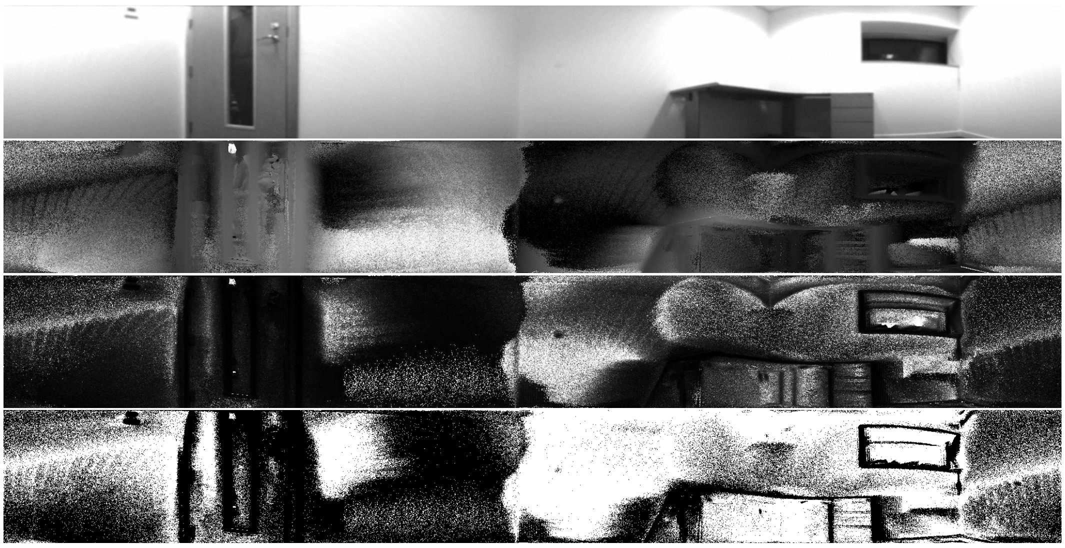
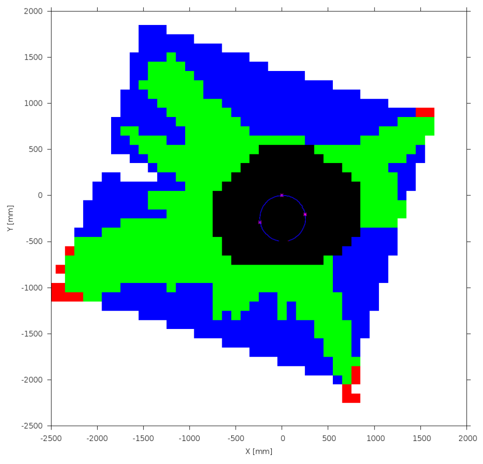
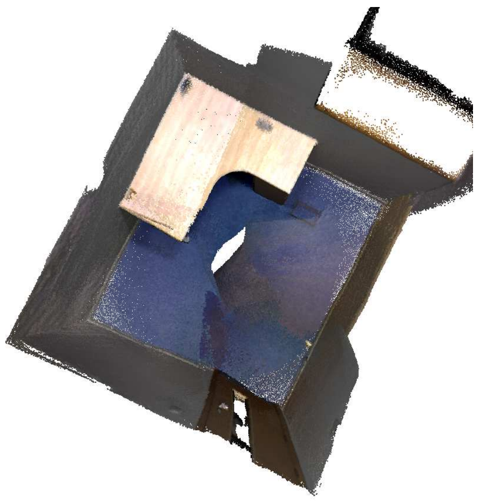


Figure 6.4: Dataset “Empty Office”, from top to bottom: reference frame, estimated depth Map, estimated depth standard deviation map, depth threshold mask.



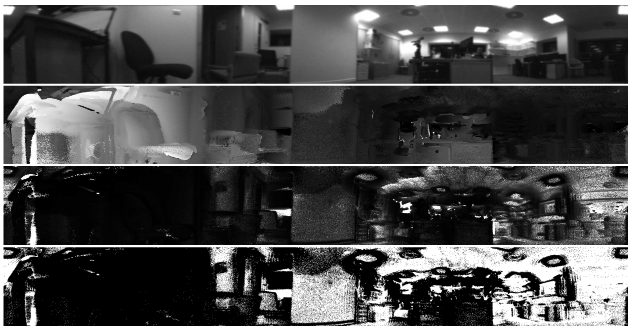
(a) Dataset “Empty Office”: Occupancy grid

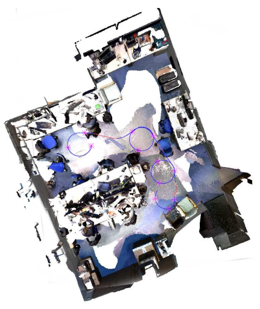


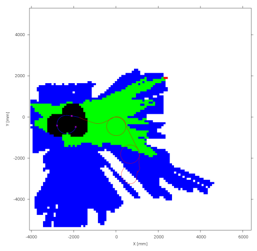
(b) Dataset “Empty Office”: top-down view of the room for comparison (screenshot from an RGBD 3D SLAM system).

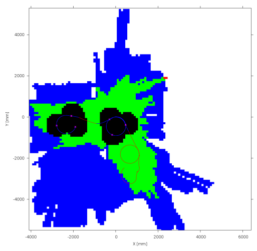
Figure 6.5

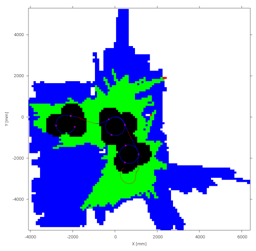
highlight how such a large area of free space has been reliably found using a sensor which is not an obvious one for such a job, and in difficult real-world conditions. This gives great promise for getting more from the cameras already installed in many low-cost robots without the need for hardware changes.


(a) Dataset "Large Office", one of the measurements, from top to bottom: reference frame, estimated depth map, estimated depth standard deviation map, depth threshold mask.


(b) Dataset "Large Office": top-down view of the room for comparison (screen-shot from an RGBD 3D SLAM system).


(c) Dataset "Large Office": Occupancy grid, step 1


(d) Dataset "Large Office": Occupancy grid, steps 1-2


(e) Dataset "Large Office": Occupancy grid, steps 1-3

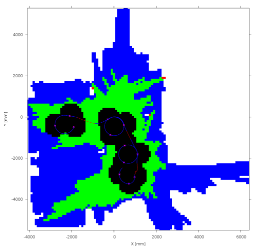

(f) Dataset "Large Office": Occupancy grid, steps 1-4

Figure 6.6

ElasticFusion Ground Truth

To prove the value of our approach for low cost mobile robots equipped with an omni-directional camera, we have performed extensive testing in real world environments, spanning multiple houses and flats with multiple rooms, from the tiniest bedrooms, through difficult for computer vision bathrooms (textureless and specular) to large scale living rooms.

The real world data is difficult to analyze quantitatively due to lack of precise ground truth. In most scenarios it would be impossible to install a VICON tracking system without disturbing the structure of the room / free space (with tripods etc). We also did not have precise 3D scanners, such as FARO Focus 3D or Velodyne. Our laser scanner setup, used in the Section 6.2.3, could not be used as for the real world tests we have moved from Pionner based mobile platform (see Figure 2.2a) to a more portable TurtleBot (see Figure 2.2b).

In the end we have decided to reuse our Virtual Camera infrastrucure (see Section 2.5). In contrast, for example, to the ground truth from the Section 5.5.2 here VICON system was unavailable, therefore the ElasticFusion system [WLSM+15] operated with built-in tracking, often resulting in rather sparse reconstructions to keep the system stable and the reconstructed shapes correct (e.q. square etc.). The resulting models were remeshed using Poisson Surface Reconstruction to create denser, watertight models. Additionally, without the VICON tracker, there is no common coordinate frame origin, so the resulting 3D models had to be manually aligned to the frame of reference of the Bundle Adjusted robot trajectories.

With such constraints and limitations we cannot definitely say how accurate such ground truth is, certainly not as accurate as the VICON enabled ones presented in the Section 2.5.1. However, as the freespace mapping is a rather coarse representation (100 mm grid), we think that such ground truth should be acceptable and has a good balance of quality against convenience and cost. The experiments below populated the ground truth occupancy grid by applying our method (Section 6.2.2) on the ground truth Virtual Camera generated depth maps. Larger datasets employ incremental reconstruction and occupancy grid population, as described in the Section 6.2.3.

As in the previous datasets, the meaning of the occupancy grid plots is as follows:

- Green: vision and the Ground Truth agree;
- Blue: found by the Ground Truth but not by vision;
- Red: found by vision but not by the Ground Truth;
- Black: area close to robot and its motion, assumed to be empty.

Unless otherwise noted, all datasets were processed using the same dense reconstruction settings:

- bilateral filter enabled,
- no TV-L1 regularization,
- depth map thresholded to leave 80 % of the best estimates,

- over/under exposed or below floor level depth map areas removed from consideration.

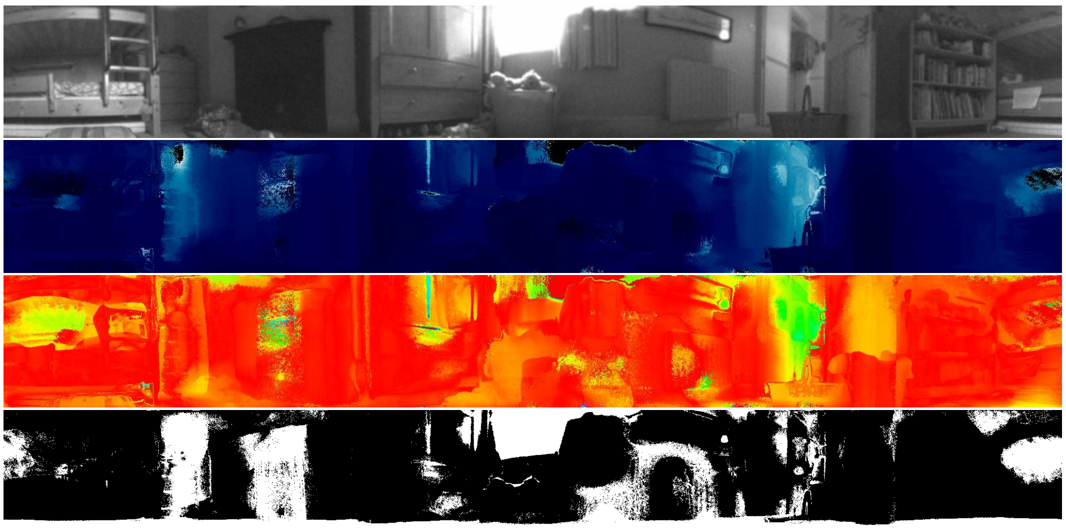


Figure 6.7: Dataset "Ealing Bedroom 1", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

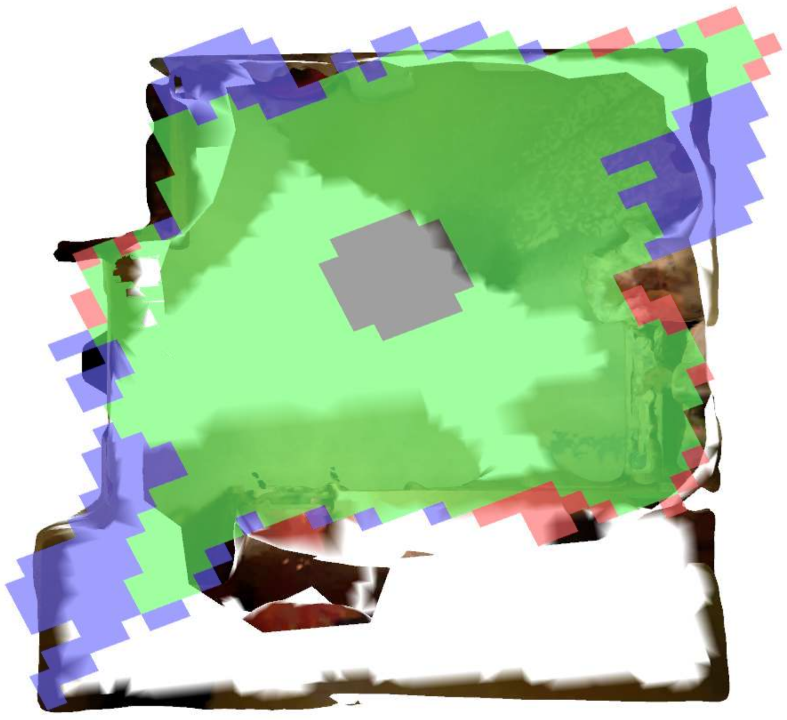


Figure 6.8: Dataset "Ealing Bedroom 1", 3D model with occupancy grid overlay.

Dataset “Ealing Bedroom 1” This dataset is a fairly typical scenario for our system: simple rectangular room, without obstacles in the middle and plenty of structure and texture on the walls aids reconstruction quality and therefore improves free space mapping. Desirably, the algorithm uncovered at least 80 % of the free space, probably more than metrics in the Table 6.1 might indicate, as the 3D scan was performed at a different point in time and, in addition to small objects being moved, the door was closed during 3D scanning, but not in the robot’s dataset (see Figure 6.8).

Robot own motion space $[m^2]$	0.32
Vision estimated free space $[m^2]$	5.9
Ground Truth free space $[m^2]$	7.0
Vision to Ground Truth overlap $[m^2]$	5.6
Estimation error area $[m^2]$	1.67
Vision / Ground Truth ratio $[\%]$	84.14

Table 6.1: Dataset “Ealing Bedroom 1”, metrics.

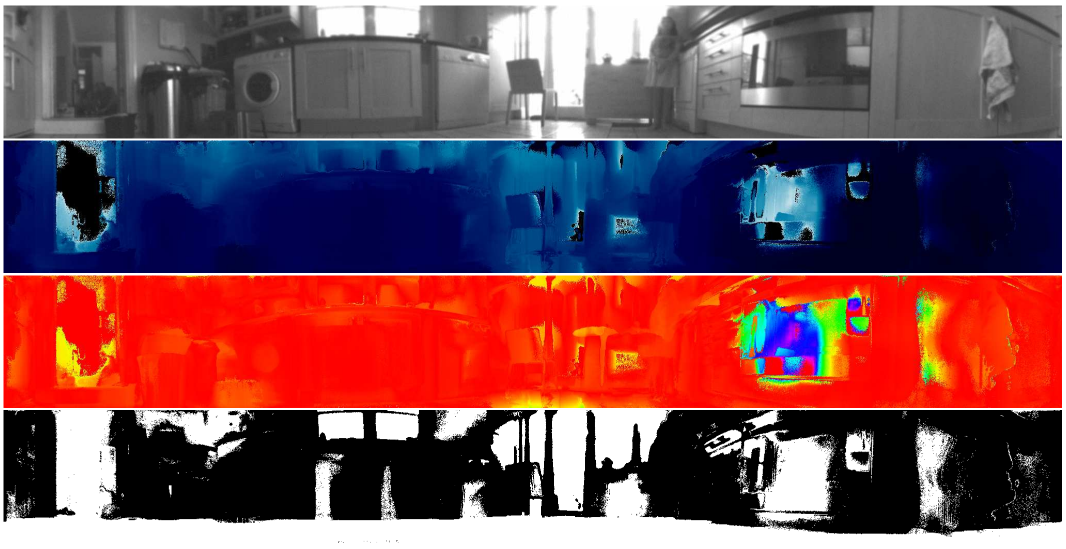


Figure 6.9: Dataset “Ealing Kitchen”, from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

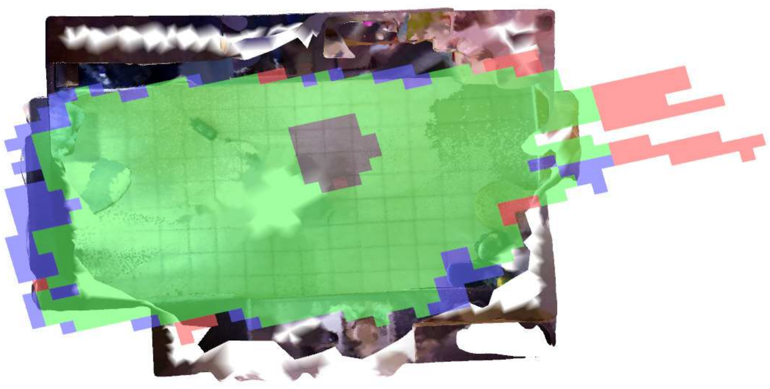


Figure 6.10: Dataset “Ealing Kitchen”, 3D model with occupancy grid overlay.

Dataset “Ealing Kitchen” This is a somewhat more challenging dataset than the previous one, as the room is larger in size and one side is mostly garden door, overexposing the image and causing noise in the depth estimates. Otherwise the overlap between our estimates and ground truth is very high, approaching 90 %.

Dataset “Ealing Bedroom 2” This dataset is similar in characteristics to the one presented in Section 6.2.3, yielding similarly good results of over 90 % of the area correctly classified.

Dataset “Ealing Livingroom” This dataset is more complex than the ones presented so far. It is a large scale oddly shaped living room, where the robot

Robot own motion space $[m^2]$	0.3
Vision estimated free space $[m^2]$	7.2
Ground Truth free space $[m^2]$	7.4
Vision to Ground Truth overlap $[m^2]$	6.6
Estimation error area $[m^2]$	1.5
Vision / Ground Truth ratio [%]	97.4

Table 6.2: Dataset "Ealing Kitchen", metrics.

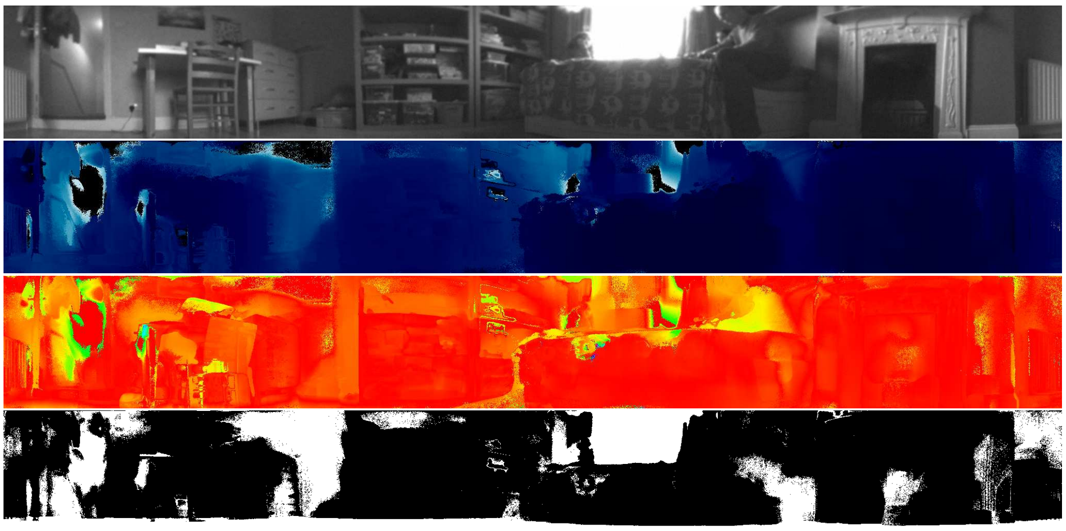


Figure 6.11: Dataset "Ealing Bedroom 2", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

Robot own motion space $[m^2]$	0.32
Vision estimated free space $[m^2]$	7.32
Ground Truth free space $[m^2]$	7.54
Vision to Ground Truth overlap $[m^2]$	6.91
Estimation error area $[m^2]$	1.04
Vision / Ground Truth ratio [%]	97.08

Table 6.3: Dataset "Ealing Bedroom 2", metrics.

performed a long trajectory with 3 circles, from each of which a depth map was estimated. The majority of errors (see in red in the top left corner of the occupancy grid 6.14) can be actually attributed to poor ground truth, as the noise in the 3D model, after Poisson Surface Reconstruction, manifested itself as a hump on the floor that limited the ground truth free space to the edge of the table. The dataset

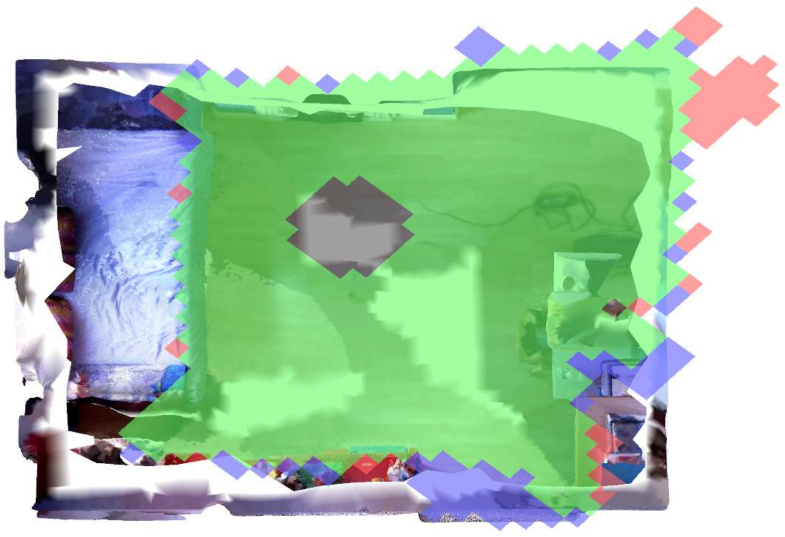


Figure 6.12: Dataset "Ealing Bedroom 2", 3D model with occupancy grid overlay.

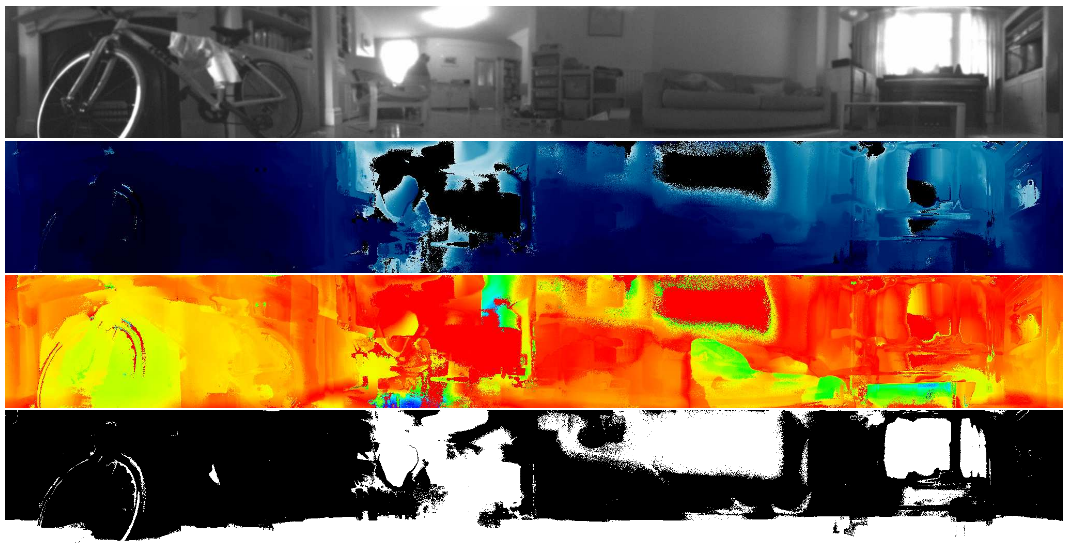


Figure 6.13: Dataset "Ealing Livingroom", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

was captured far apart in time from the 3D model, which may have contributed to minor errors.

Dataset "Elephant & Castle Livingroom" Similarly to 6.2.3, this dataset exhibits a challenging environment, being a large scale room with a long trajectory with 5 circles that the robot performed. Here the error sources are twofold. Partially it is the ground truth quality. This room was scanned with a Microsoft Kinect V2

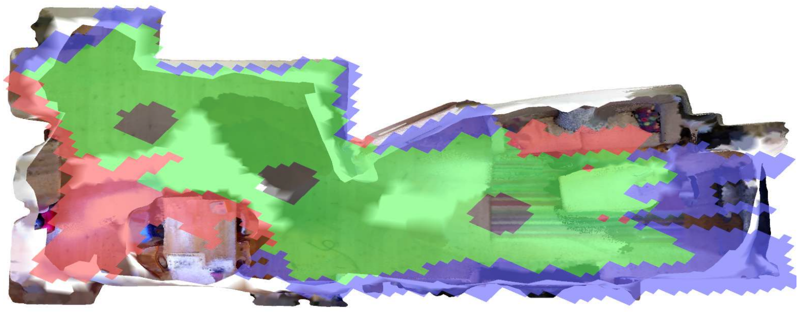


Figure 6.14: Dataset "Ealing Livingroom", 3D model with occupancy grid overlay.

Robot own motion space $[m^2]$	0.91
Vision estimated free space $[m^2]$	16.55
Ground Truth free space $[m^2]$	18.55
Vision to Ground Truth overlap $[m^2]$	12.97
Estimation error area $[m^2]$	9.46
Vision / Ground Truth ratio [%]	87.79

Table 6.4: Dataset "Ealing Livingroom", metrics.

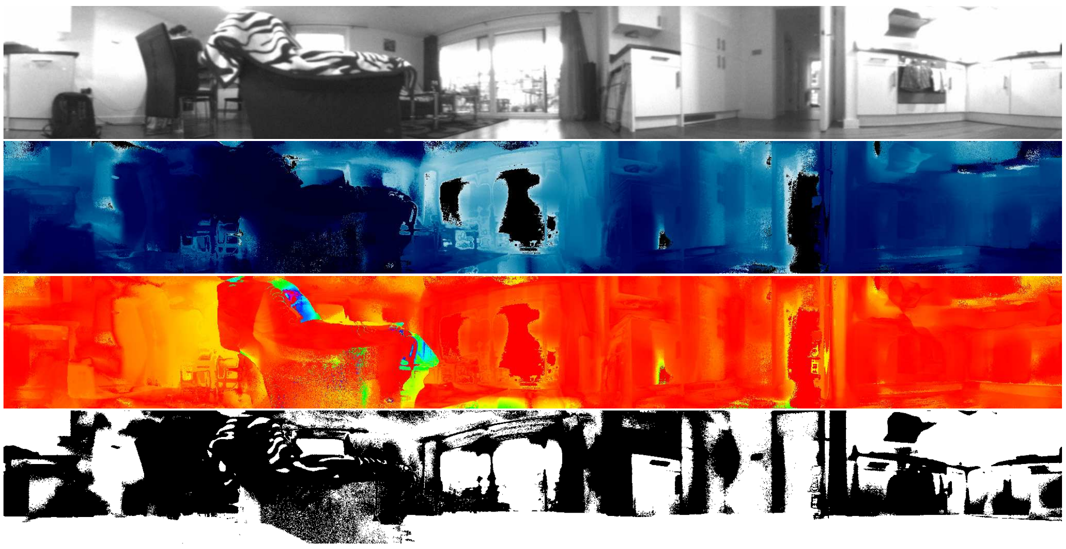


Figure 6.15: Dataset "Elephant & Castle Livingroom", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

RGBD time-of-flight camera that operates in the IR band that is often absorbed by black objects. Table tops in this room are made of glass, which is impossible to reconstruct even with active sensors. Hence there are missing tables in the ground truth model. For 3D scanning the door was locked and window blinds shut (to

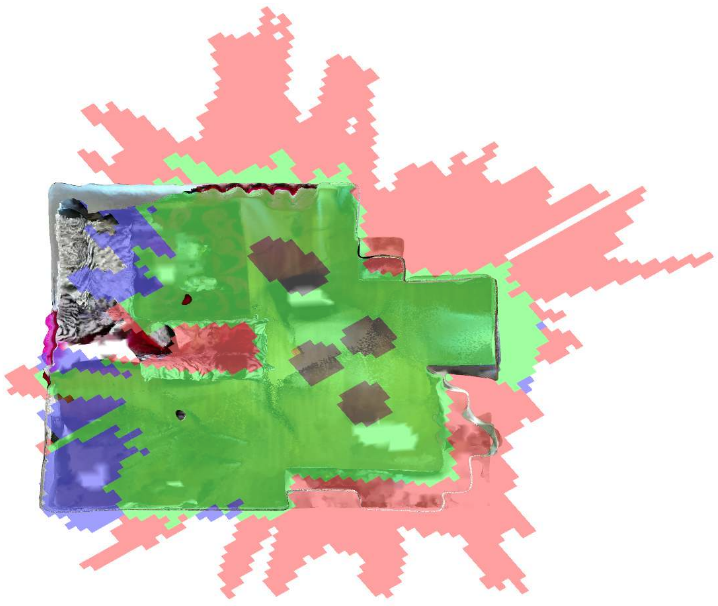


Figure 6.16: Dataset “Elephant & Castle Livingroom”, 3D model with occupancy grid overlay.

obstruct infrared ratiation from the sunlight). From the method side the scale of the room combined with rather textureless plain white furniture contributed greatly to the resulting noise.

Robot own motion space $[m^2]$	1.46
Vision estimated free space $[m^2]$	34.61
Ground Truth free space $[m^2]$	20.64
Vision to Ground Truth overlap $[m^2]$	17.6
Estimation error area $[m^2]$	20.05
Vision / Ground Truth ratio [%]	167.68

Table 6.5: Dataset “Elephant & Castle Livingroom”, metrics.

Dataset “Elephant & Castle Bathroom” Bathrooms are probably one of the most difficult environments to reconstruct. While the scale is not overwhelming, the other aspects are. Bathrooms are largely textureless, with mirrors and other specular surfaces that break lambertian photoconsistency assumptions. Despite these adversities our algorithm estimated almost 70 % of the free space area correctly.

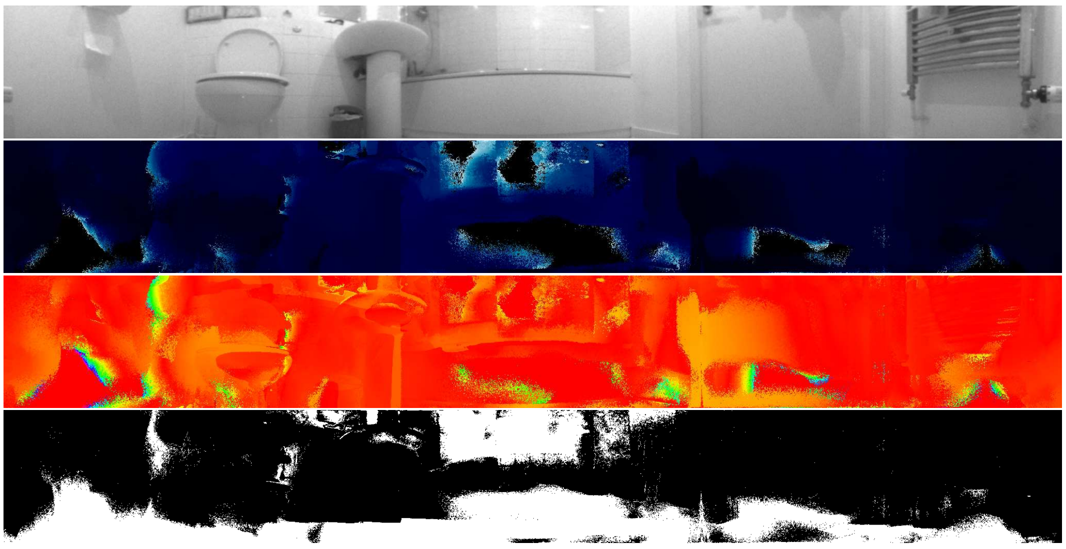


Figure 6.17: Dataset “Elephant & Castle Bathroom”, from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

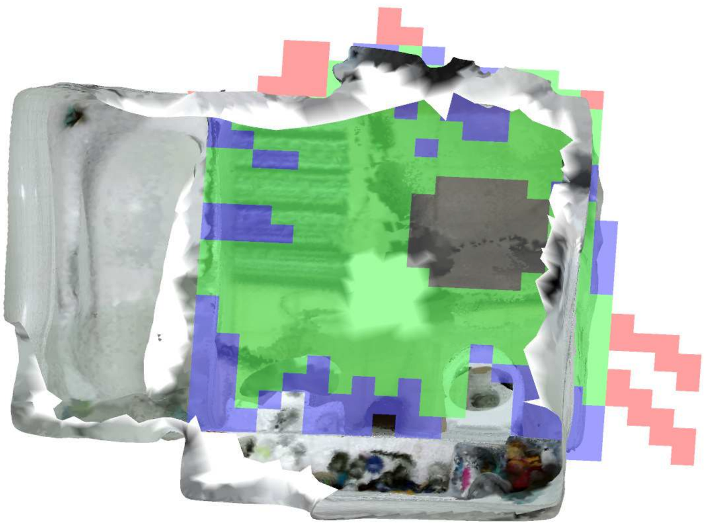


Figure 6.18: Dataset “Elephant & Castle Bathroom”, 3D model with occupancy grid overlay.

Dataset “Elephant & Castle Bedroom 1” Bedrooms, as in the previous datasets, are relatively easy to analyze. Here 75 % of the free space area was estimated correctly, and the major part of the error (bottom left corner of the occupancy map) was caused by sunlight entering the room from the windows on the left. When build-

Robot own motion space $[m^2]$	0.32
Vision estimated free space $[m^2]$	2.43
Ground Truth free space $[m^2]$	3.10
Vision to Ground Truth overlap $[m^2]$	2.16
Estimation error area $[m^2]$	1.21
Vision / Ground Truth ratio [%]	78.38

Table 6.6: Dataset "Elephant & Castle Bathroom", metrics.

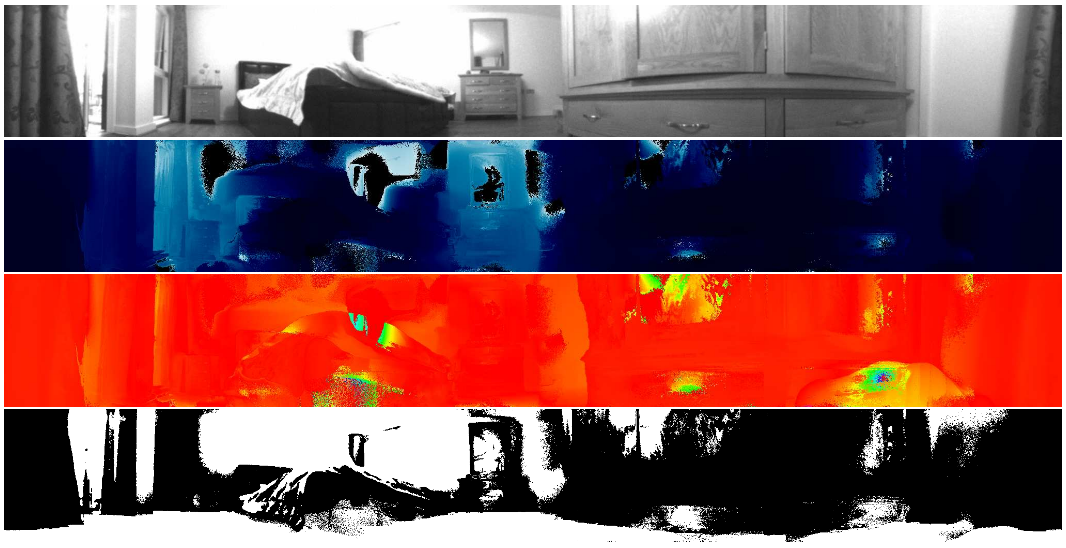


Figure 6.19: Dataset "Elephant & Castle Bedroom 1", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

ing the 3D model of the room the blinds were shut. It is worth noting the peculiar shape of the free space and the correct result that would allow the robot to travel through passages to cover the entire room.

Robot own motion space $[m^2]$	0.6
Vision estimated free space $[m^2]$	7.87
Ground Truth free space $[m^2]$	8.63
Vision to Ground Truth overlap $[m^2]$	6.47
Estimation error area $[m^2]$	3.56
Vision / Ground Truth ratio [%]	91.19

Table 6.7: Dataset "Elephant & Castle Bedroom 1", metrics.

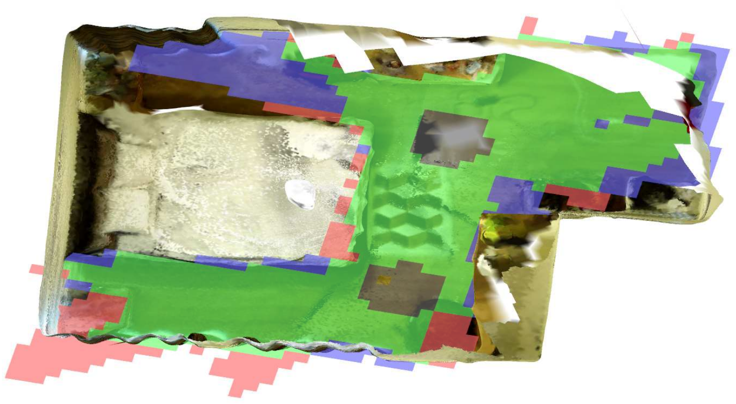


Figure 6.20: Dataset "Elephant & Castle Bedroom 1", 3D model with occupancy grid overlay.

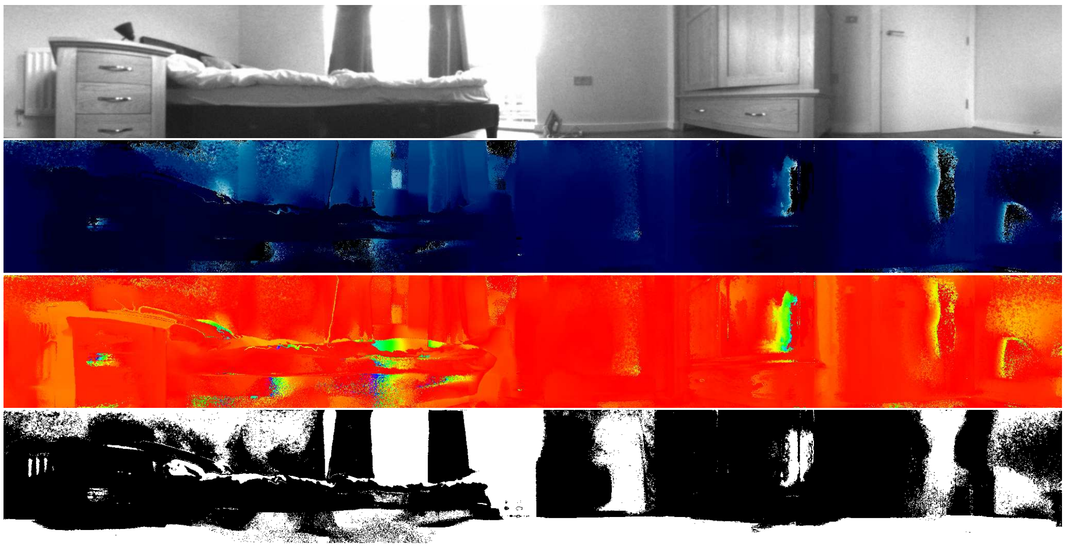


Figure 6.21: Dataset "Elephant & Castle Bedroom 2", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

Dataset "Elephant & Castle Bedroom 2" In this dataset not only is the number of false positive labellings minimal, but the overall coverage of the ground truth area is superb, approaching 80 %.

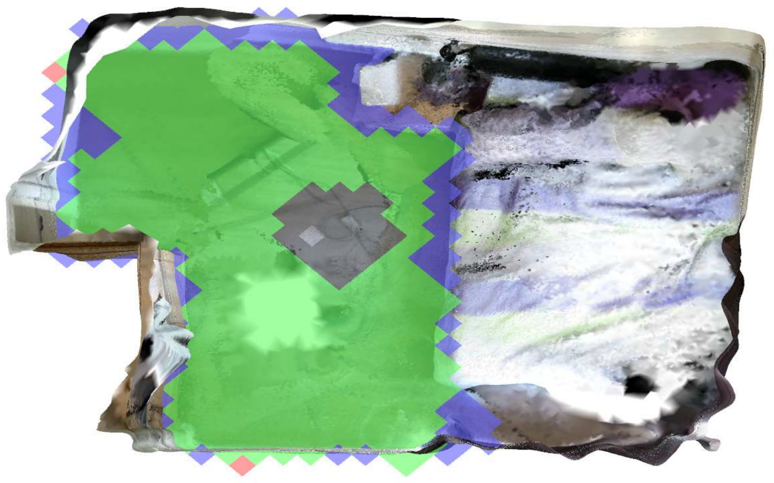


Figure 6.22: Dataset “Elephant & Castle Bedroom 2”, 3D model with occupancy grid overlay.

Robot own motion space $[m^2]$	0.3
Vision estimated free space $[m^2]$	3.73
Ground Truth free space $[m^2]$	4.76
Vision to Ground Truth overlap $[m^2]$	3.71
Estimation error area $[m^2]$	1.07
Vision / Ground Truth ratio $[\%]$	78.36

Table 6.8: Dataset “Elephant & Castle Bedroom 2”, metrics.

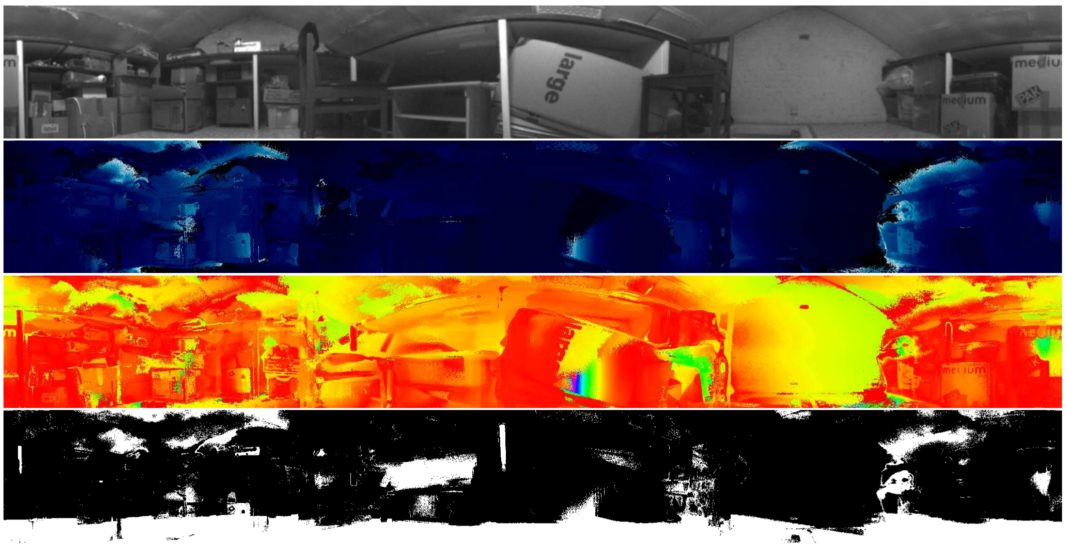


Figure 6.23: Dataset "Wandsworth Attic", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

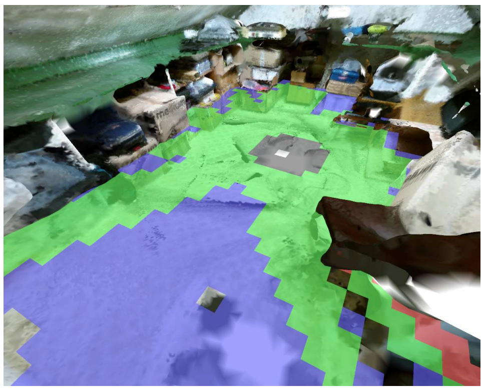


Figure 6.24: Dataset "Wandsworth Attic", 3D model with occupancy grid overlay.

Dataset "Wandsworth Attic" This room is perhaps a rather peculiar choice of environment for a domestic robot, but is included here nonetheless to test even the edge cases. Here the attic area is equal to the footprint of the house, thus is on the large side. An angled ceiling is what differentiates this dataset from the others, but for this method it should not make a difference. The coverage is acceptable and in this case some errors were caused by the ground truth. Boxes and other stored items on both sides have gaps in between or are rather hollow by design (spare chairs, coffee tables). This was not picked up by our 3D model — not after Poisson Surface Reconstruction, but was present in passive omnidirectional reconstruction. The discrepancy caused some of the outlier errors as seen in Figure 6.24.

Robot own motion space $[m^2]$	0.32
Vision estimated free space $[m^2]$	5.08
Ground Truth free space $[m^2]$	7.1
Vision to Ground Truth overlap $[m^2]$	4.5
Estimation error area $[m^2]$	3.18
Vision / Ground Truth ratio $[\%]$	71.54

Table 6.9: Dataset "Wandsworth Attic", metrics.

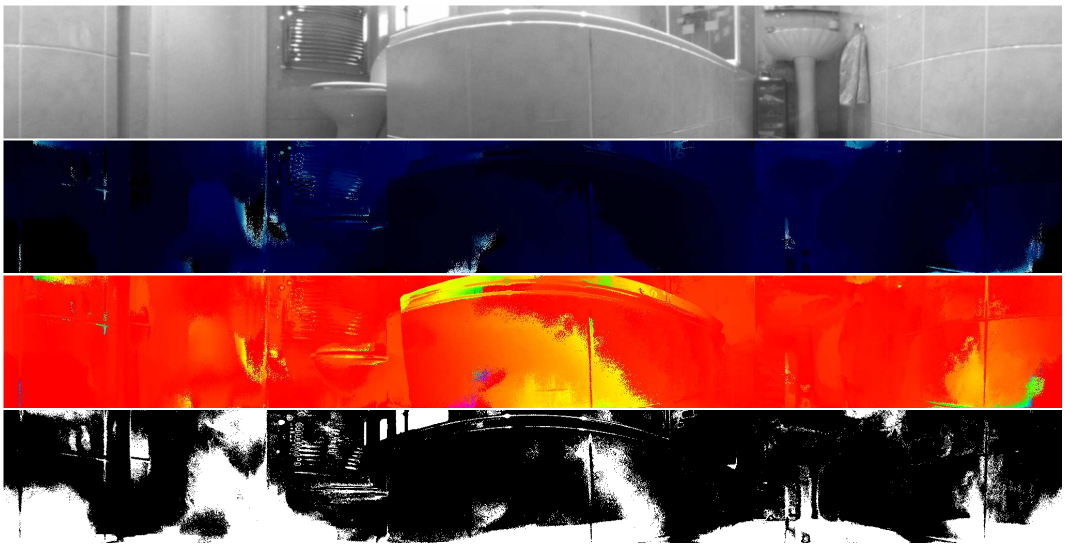


Figure 6.25: Dataset "Wandsworth Bathroom", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

Dataset "Wandsworth Bathroom" Similarly to dataset 6.2.3, the results are substandard due to the extreme nature of the environment. Additionally, in this case the robot underwent purely linear motion, which cannot provide a baseline along the direction of motion, rendering the depth estimates along this axis completely unreliable.

Robot own motion space $[m^2]$	0.32
Vision estimated free space $[m^2]$	1.7
Ground Truth free space $[m^2]$	1.81
Vision to Ground Truth overlap $[m^2]$	1.09
Estimation error area $[m^2]$	1.33
Vision / Ground Truth ratio $[\%]$	93.92

Table 6.10: Dataset "Wandsworth Bathroom", metrics.

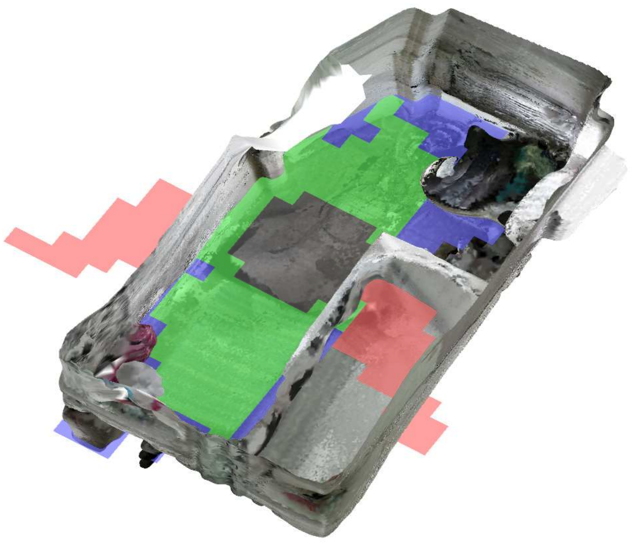


Figure 6.26: Dataset “Wandsworth Bathroom”, 3D model with occupancy grid overlay.

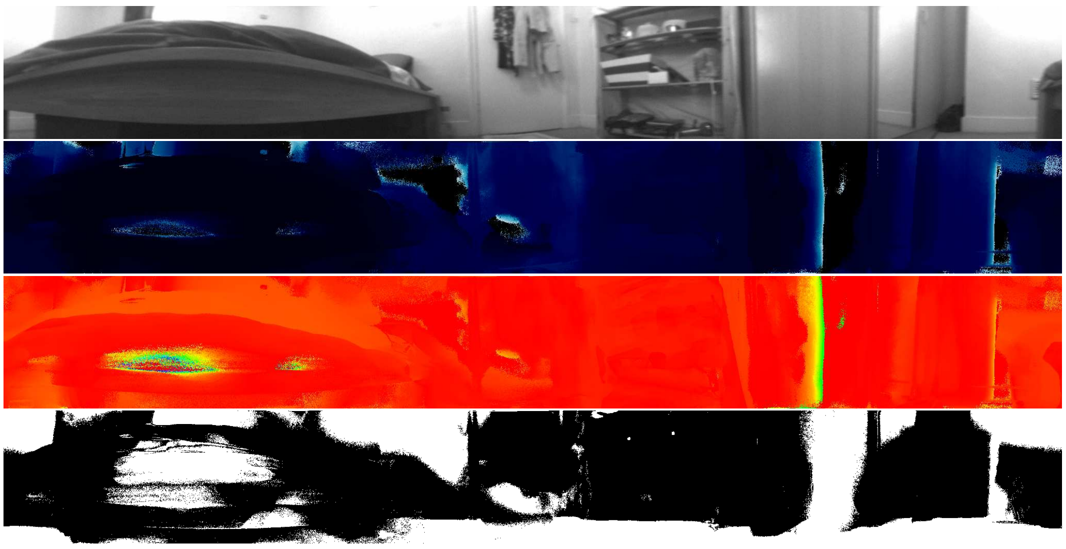


Figure 6.27: Dataset “Wandsworth Bedroom 1”, from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

Dataset “Wandsworth Bedroom 1” While our method on this dataset estimated most of the free space correctly, there are some issues to discuss. This was a 2 part trajectory. First one was a circle closer to the door, then a straight linear movement along the bed. While the first part did not contribute much error there is

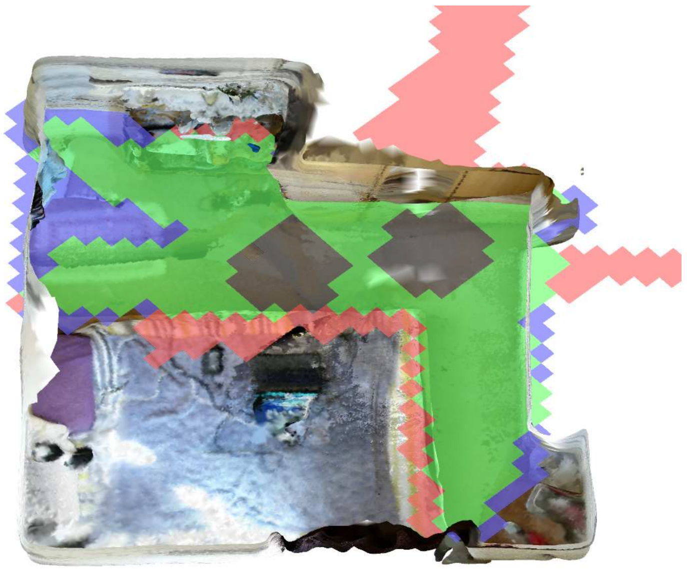


Figure 6.28: Dataset “Wandsworth Bedroom 1”, 3D model with occupancy grid overlay.

an exception. One wardrobe has a mirror door, which completely distorts our depth estimates. Hence the false positive area in the top left corner of the occupancy grid. Second, linear movement, as mentioned in the Section 6.2.3 has very large uncertainties along the direction of motion, hence the error in the top right corner.

Robot own motion space $[m^2]$	0.6
Vision estimated free space $[m^2]$	5.07
Ground Truth free space $[m^2]$	3.84
Vision to Ground Truth overlap $[m^2]$	2.97
Estimation error area $[m^2]$	2.97
Vision / Ground Truth ratio $[\%]$	132.03

Table 6.11: Dataset “Wandsworth Bedroom 1”, metrics.

Dataset “Wandsworth Bedroom 2” This was a particularly challenging dataset, as this is a rather oddly shaped small bedroom occupied mostly by the bed itself, leaving only a narrow passage for the robot. This implies linear movement with its consequences (see Section 6.2.3). Additionally, it is important to remember that the robot moves only on a plane (for detailed discussion see Section 5.5.4), thus has extremely poor reconstruction of the horizontal parts of the scene (which in this

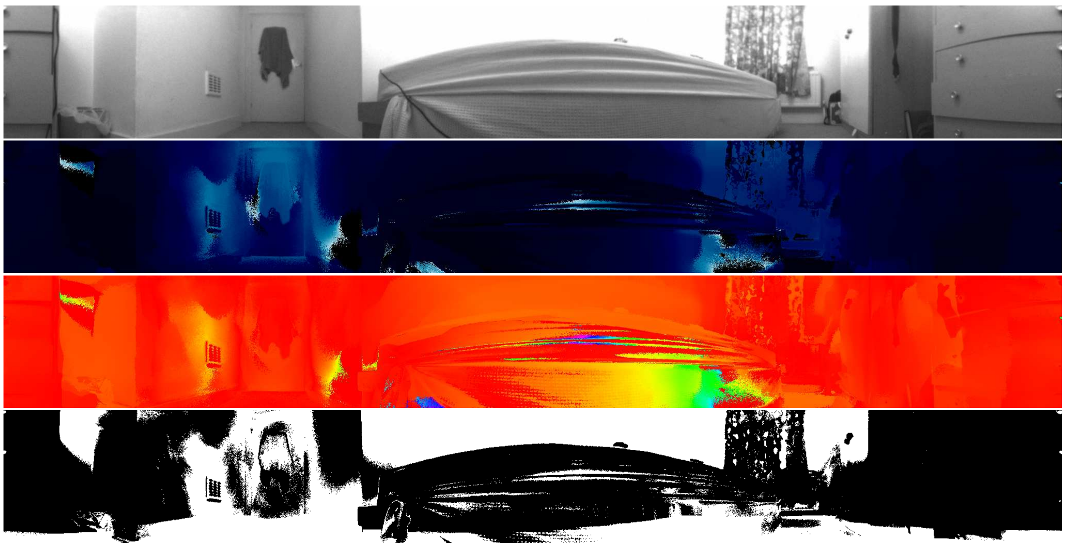


Figure 6.29: Dataset “Wandsworth Bedroom 2”, from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

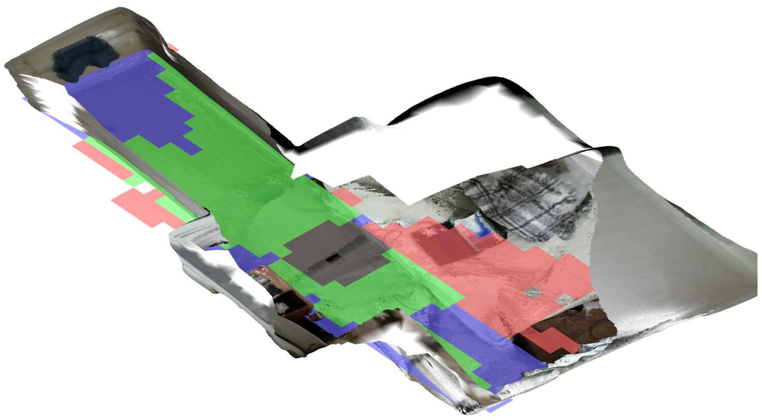


Figure 6.30: Dataset “Wandsworth Bedroom 2”, 3D model with occupancy grid overlay.

dataset is the edge of the bed). This results in a noisy depth map estimate for the bed region, thus yielding a large number of false positives in the centre bottom part of the occupancy grid.

Dataset “Wandsworth Bedroom 3” Considering the scale of this room, combined with other adverse elements (mirror door of the wardrobe, overexposed windows), one must admit that the estimation is acceptable. After a brief motion the robot uncovers almost 5 m^2 of valid free space. The erroneous areas are probably

Robot own motion space $[m^2]$	0.32
Vision estimated free space $[m^2]$	3.39
Ground Truth free space $[m^2]$	3.69
Vision to Ground Truth overlap $[m^2]$	2.23
Estimation error area $[m^2]$	2.69
Vision / Ground Truth ratio [%]	91.86

Table 6.12: Dataset “Wandsworth Bedroom 2”, metrics.

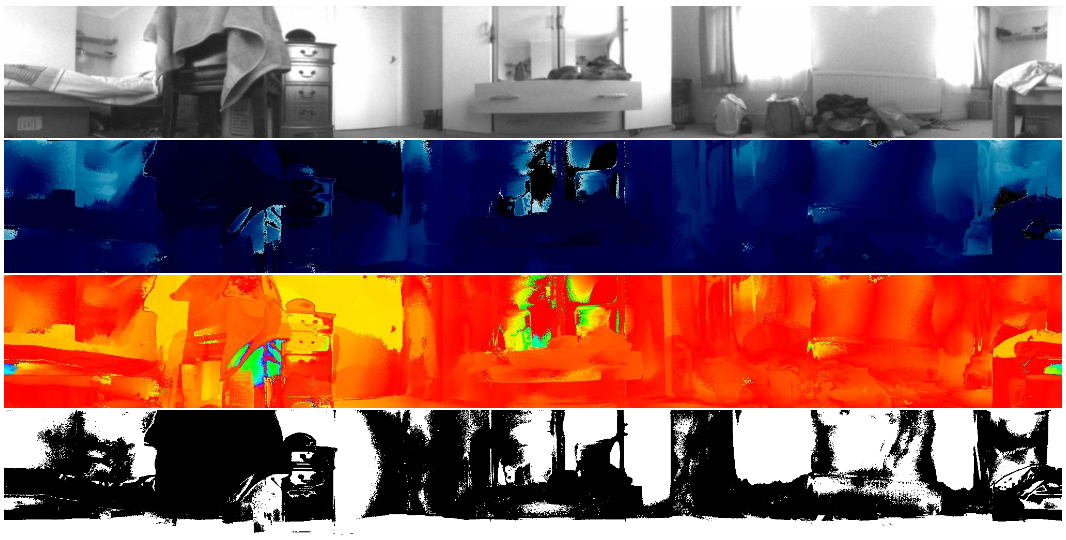


Figure 6.31: Dataset “Wandsworth Bedroom 3”, from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

mostly caused by discrepancies between passive vision estimation of the desk and chair and their 3D model representations, likely smoothed by the remeshing process.

Robot own motion space $[m^2]$	0.3
Vision estimated free space $[m^2]$	5.16
Ground Truth free space $[m^2]$	7.15
Vision to Ground Truth overlap $[m^2]$	4.76
Estimation error area $[m^2]$	2.79
Vision / Ground Truth ratio [%]	72.16

Table 6.13: Dataset “Wandsworth Bedroom 3”, metrics.

Dataset “Wandsworth Kitchen” This dataset is an example of an outlier. Here omnidirectional depth estimation exhibits significant noise (due to linear motion and

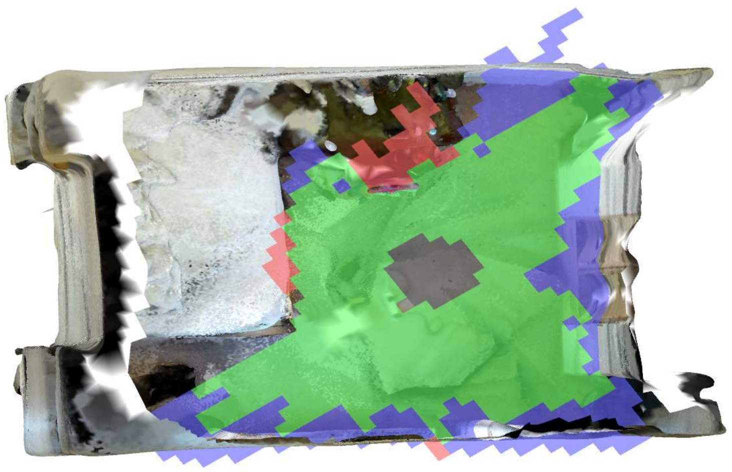


Figure 6.32: Dataset "Wandsworth Bedroom 3", 3D model with occupancy grid overlay.

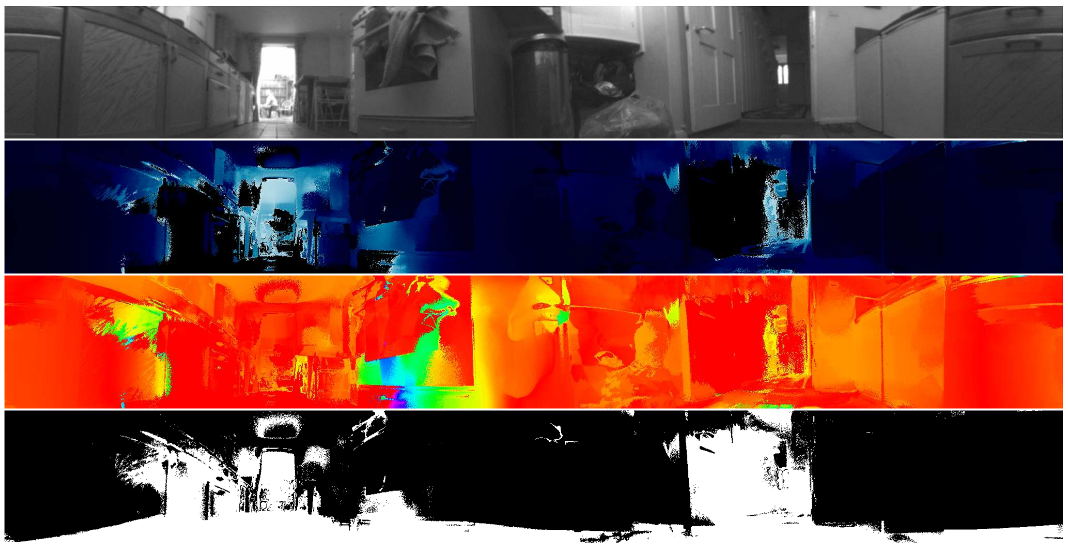


Figure 6.33: Dataset "Wandsworth Kitchen", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

varying auto-exposure), but also the ground truth model is deficient in many aspects. The inner part is reconstructed decently, but as we moved forward more depth map noise was present, to the point where the sunlight flooded the sensor and it generated no valid depth estimates.

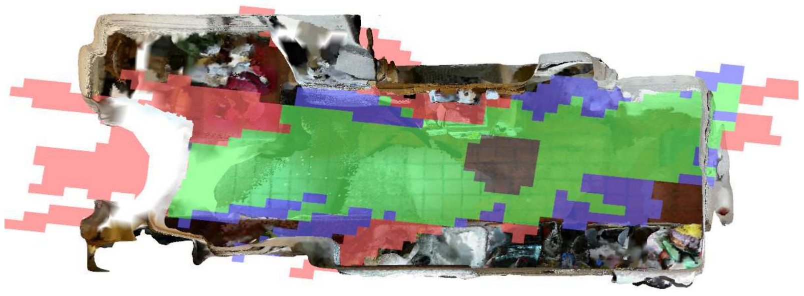


Figure 6.34: Dataset “Wandsworth Kitchen”, 3D model with occupancy grid overlay.

Robot own motion space $[m^2]$	0.31
Vision estimated free space $[m^2]$	7.33
Ground Truth free space $[m^2]$	5.57
Vision to Ground Truth overlap $[m^2]$	4.04
Estimation error area $[m^2]$	4.82
Vision / Ground Truth ratio [%]	131.6

Table 6.14: Dataset “Wandsworth Kitchen”, metrics.

Dataset “Wandsworth Livingroom” This dataset represents a typical, albeit slightly larger than some of the bedrooms, room. The occupancy grid was filled with data from multiple depth map estimates, creating a rather tight fit to the true free space. Minor errors are due to ground truth from Poisson Surface Reconstruction vs real depth maps discrepancies as well as non-lambertian objects such as a mirror, or the sunlight overexposing parts of the scene through blinds.

Dataset “Wandsworth Lab” In these dataset we, unusually, apply multiple depth estimates in a fairly small room (3 $\times$ 3m). Most of the valid free space was estimated correctly, and minor errors are caused by occlusion differences between ground truth depth maps and real depth maps and minor differences in scene clutter.

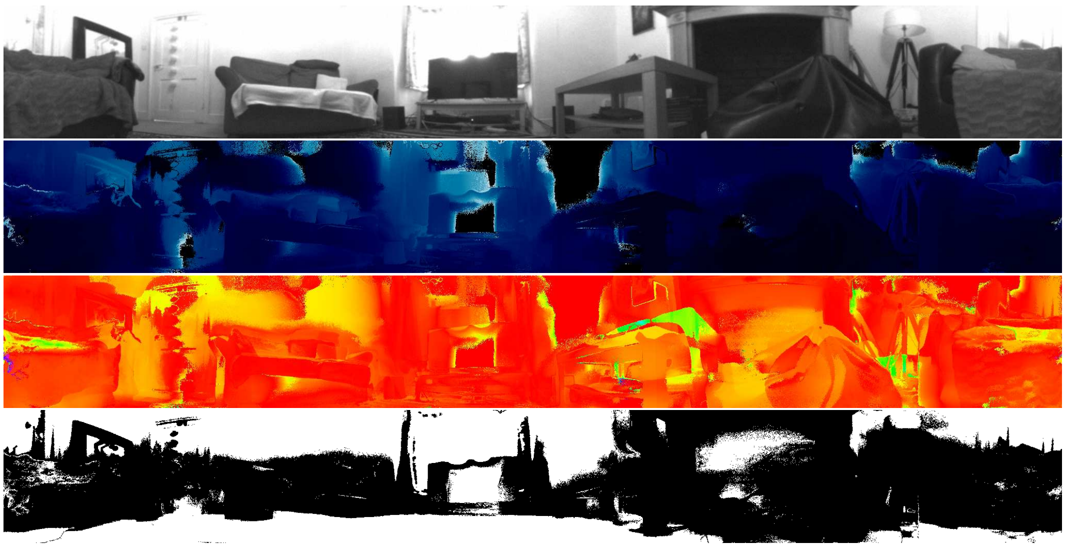


Figure 6.35: Dataset “Wandsworth Livingroom”, from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

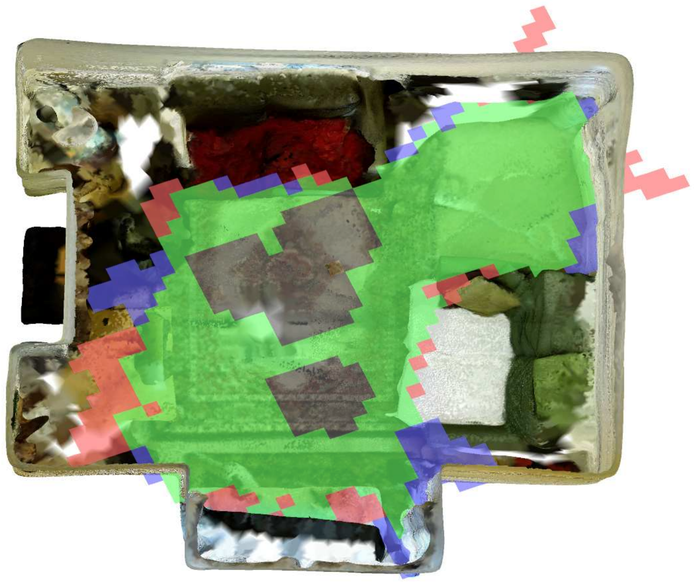


Figure 6.36: Dataset “Wandsworth Livingroom”, 3D model with occupancy grid overlay.

What could be improved is the ground truth for the real world experiments. Due to our limitations we had to resort to Kinect360/KinectOne depth cameras with the ElasticFusion system, with manual alignment, which leaves a lot to be desired. Unfortunately, proper and precise tools for such tasks are almost always prohibitively

Robot own motion space $[m^2]$	1.17
Vision estimated free space $[m^2]$	5.71
Ground Truth free space $[m^2]$	5.48
Vision to Ground Truth overlap $[m^2]$	4.66
Estimation error area $[m^2]$	1.87
Vision / Ground Truth ratio [%]	104.20

Table 6.15: Dataset "Wandsworth Livingroom", metrics.

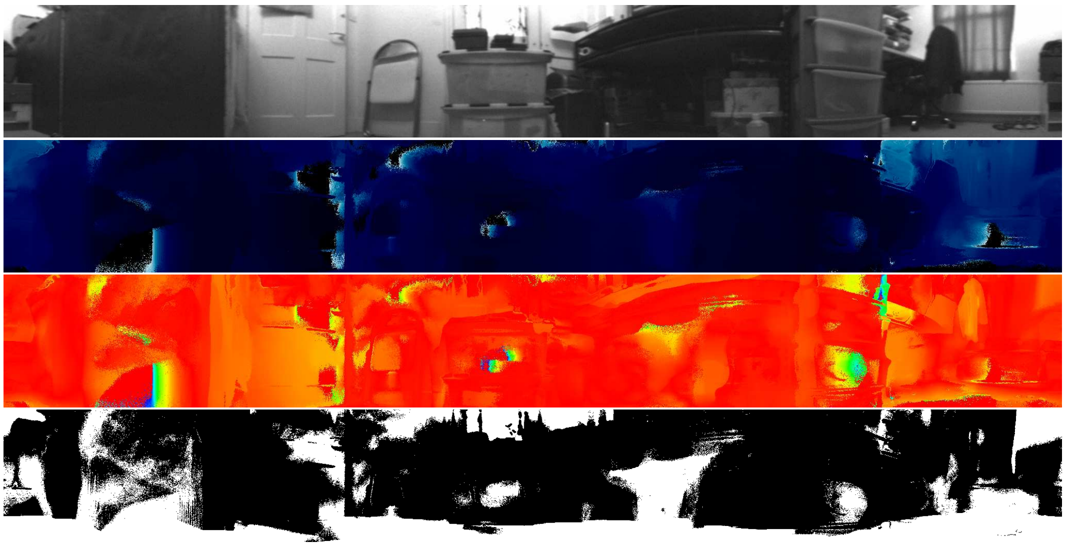


Figure 6.37: Dataset "Wandsworth Lab", from top to bottom: reference frame, estimated depth map, depth map error, depth threshold mask.

Robot own motion space $[m^2]$	0.63
Vision estimated free space $[m^2]$	4.19
Ground Truth free space $[m^2]$	4.75
Vision to Ground Truth overlap $[m^2]$	3.8
Estimation error area $[m^2]$	1.34
Vision / Ground Truth ratio [%]	88.2

Table 6.16: Dataset "Wandsworth Lab", metrics.

expensive, often heavy and bulky and may be difficult to set up in a small real rooms. Dataset collection with a robot across various properties is often an expensive and time consuming enterprise, in many labs involving many people and even hiring extra staff. Ground truth with reasonable quality is even more difficult, as we must double and triple check our measurements constantly to assure the setup is in spec.

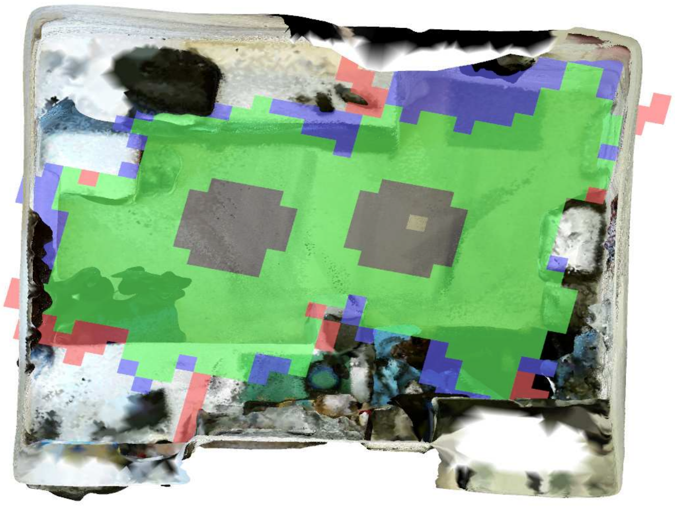


Figure 6.38: Dataset “Wandsworth Lab”, 3D model with occupancy grid overlay.

6.3 Parameterized Model Fitting

While the occupancy grid maps described in Section 6.2 are of extreme use for the robot and its path planning tasks, in this section we would like to obtain ever higher level information, possibly useful not only for robot functionality, but for the customer and the user interface as well. As before the primary inputs to the method are omnidirectional depth maps estimated as described in Chapter 5. Instead of estimating occupancy grid and free space area we focus on estimating the overall room shape and size. For the robot such knowledge might lead to room recognition (combined with other estimates/sensors) and for the user of the robot it brings human readable blueprint like floorplan maps.

6.3.1 Room Model

While we have experimented with multiple geometric models, some of them did not converge well or were not very meaningful. For example discretized sphere model (parametrized by distances from the centre to vertices) is just equal patch-like to depth map averaging. Convex polygon was more meaningful, but on many rooms it was overparametrized thus had poor convergence. We have settled on the simplest and most intuitive model of a room — a cuboid box parameterized as follows: $\mathbf{p} = [x_-, x_+, y_-, y_+, \theta]$. The box revolves around robot's reference pose (for which the depth map was calculated) with the angle θ and the four other parameters are distances to the walls. Therefore the box has width $x_- + x_+$ and length $y_- + y_+$. The height of the box is not estimated and is an arbitrary large value that ensures full coverage of the camera viewing angles. From the model coefficients described above we generate a triangular 3D model, composed of 8 triangles (two for each face) and therefore 24 vertices. A visual representation of the model is presented in figure 6.39.

Now, with the model defined, we might explain the principle of operation of the method. With any kind of geometric model of the environment we can, by ray-tracing, generate a synthetic depth map, as presented in the Figure 6.40. Now the overall idea is to optimize the model parameters, so the generated depth map fits the real omnidirectional depth map, as estimated by the method from Chapter 5.

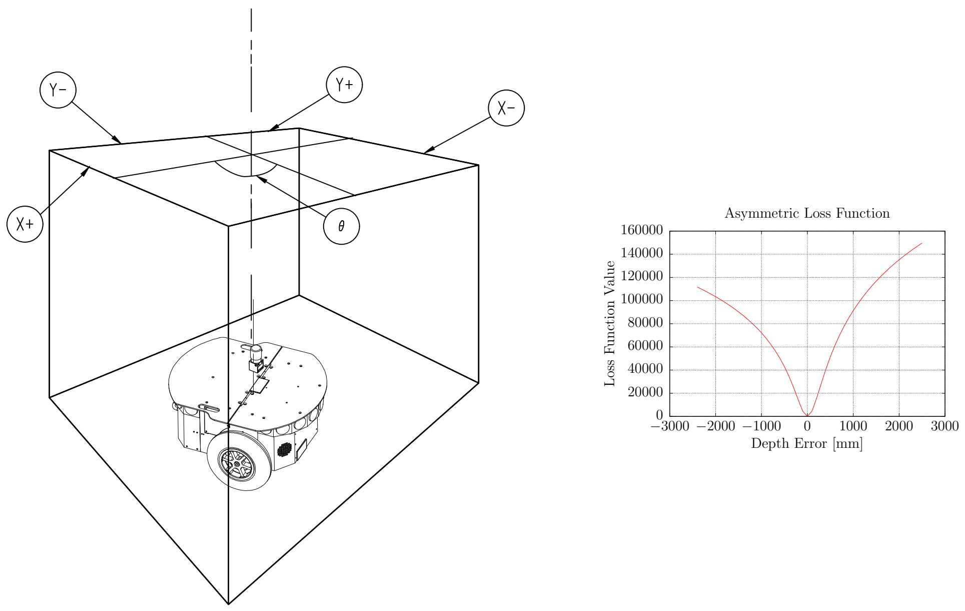


Figure 6.39: Box model and asymmetric loss function.

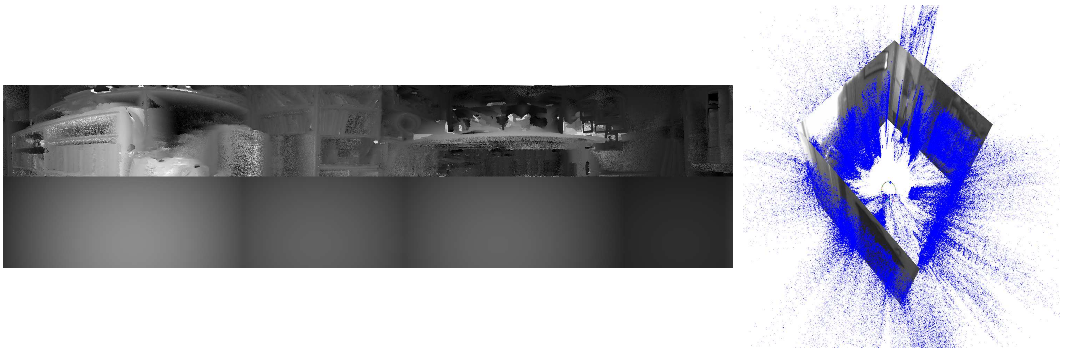


Figure 6.40: Original and model raytraced depth map and successfully estimated box.

6.3.2 Differentiable Renderer

Our method employs automatic differentiation (see Section 2.3) with forward accumulation to compute Jacobians. Each step of the calculation (triangular mesh generation, ray-triangle intersection, camera projection, residual and loss function computation), both on the CPU and the GPU, carries partial derivatives with respect to the model coefficients. After computing each per-pixel residual, the residual value and the partial derivatives are summed with a GPU reduce operation.

With the triangular 3D model of the current estimate, we perform per pixel ray-tracing. For each pixel we iterate through all the triangles of the model, check ray-triangle intersection as described in [MT97] and calculate the resulting depth $d_b(\mathbf{p}, x, y)$ at which the intersection occurs (with Z-buffer like logic in place). For each pixel we also have our measured depth value $d_m(x, y)$. These two depths create the residual of a cost function:

$$E(\mathbf{p}, x, y) = M(d_m(\mathbf{p}, x, y) - d_b(x, y), c_n, c_p), \tag{6.1}$$

where $M(e, c_n, c_p)$ is an asymmetric Cauchy loss function defined as follows:

$$M(e, c_n, c_p) = \begin{cases} \frac{c_n^2}{2} log\left(1 + (\frac{e}{c_n})^2\right), & \text{if } e < 0 \\ \frac{c_p^2}{2} log\left(1 + (\frac{e}{c_p})^2\right), & \text{otherwise .} \end{cases} \tag{6.2}$$

An example plot of this function is presented in figure 6.39. The reason to use an asymmetric cost function is that when we fit a box to an omnidirectional depth map with the aim of finding the outer shape of a room, we should pay less attention to depth data which is closer to the camera than predicted by the model (since this will often be caused by furniture or other clutter) than to depth data which is farther away then predicted by the model. We have experimented with asymmetric Huber and Tukey as well, but Cauchy had the best convergence properties.

The final energy function, after the reduction over the entire image, and therefore the optimization problem is:

$$\underset{\mathbf{p}}{\text{argmin}}\, E(\mathbf{p}) = \underset{\mathbf{p}}{\text{argmin}} \sum_{x,y} M(d_m(\mathbf{p}, x, y) - d_b(x, y), c_n, c_p) \ . \tag{6.3}$$

The resulting cost function value, together with the Jacobian, is used in a Levenberg-Marquardt optimization scheme [AKO] to estimate model parameters. We would like to mention that performing a coordinate descent approach (first estimating just box dimensions, then just box orientation θ) yields faster convergence and better final results. The output of the method is the fitted box model and, optionally, a raytraced depth map, as seen in Figure 6.40.

6.3.3 Incremental Room Layout Recovery

Here we present additional possible applications of our method, beyond single room shape estimation:

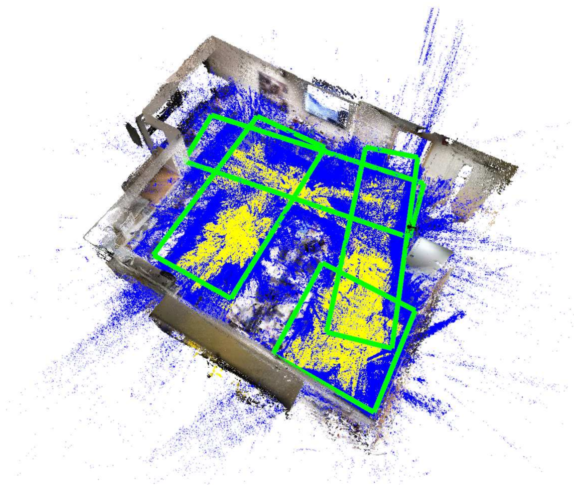


Figure 6.41: Dataset “Lab”, showing a union of multiple box estimates.

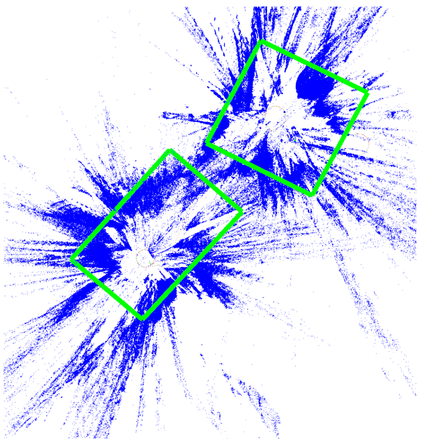


Figure 6.42: Dataset “Through Door”, showing two non-overlapping boxes, enabling inference that the robot passed between rooms.

- Large scale, more complex, room mapping can be performed in an incremental manner during active exploration by computing the boolean union of single box estimates; Figure 6.41 is a typical example of such a result,
- The method, employed in an active exploration loop, could allow for room demarcation and other higher level reasoning. Figure 6.42 shows two box estimates, each taken in a separate room. The fact that boxes don’t overlap indicates that the robot has driven through a portal (door).
- As the method is robust, the result of the method (room dimension estimates) could be fed into a Machine Learning classifier to enable room recognition,
- Swapping asymmetric cost function coefficients estimates the innermost box instead of the outermost boundary. This can be thought of as free space estimation. This representation may be beneficial when compared to occupancy grids presented in the Section 6.2.

6.3.4 Experimental Results

This section describes the results of multiple experiments where our approach was put to the test using synthetic data and real data from office rooms and real world household rooms which are our real target case. In figures we mark the estimated box with a green rectangle and the ground truth with a red one. Vertical stripes on the reference frame mark the corners of the respective box.

Since our room estimation method is purely vision based, it is worth noting that the box estimated might be different from real wall to wall measurements, if, for example, large pieces of furniture obstruct the view, as is often the case for a small mobile robot equipped with our type of lens. From the robot's point of view, furniture, wardrobes and beds are the walls of the room. Top down view, blueprint like, floor plans therefore require more than visual reasoning (or wider upward viewing angles) and do not reflect the robot's world view.

Synthetic Data

To first characterize our method, we present purely synthetic datasets, generated a modified PovRay raytracer, introduced in the Section 5.5.1.

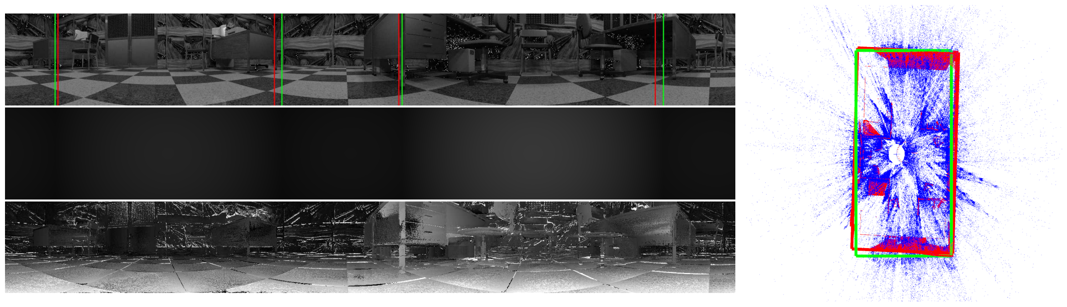


Figure 6.43: Dataset "Synthetic Textured"; from top to bottom: reference frame, raytraced depth map, raw depth map. On the right: top down view of the point cloud with the box estimate.

Synthetic — Textured This dataset is probably an almost ideal case. It is a textured scene with clutter only along the walls. The estimates listed in Table 6.17 are almost perfect. The only limiting factor here is the overall size of the room, as depth estimates at far range, with relatively small baseline, are highly unreliable.

Ground Truth Dimensions $[m]$	8.00 x 5.00
Ground Truth Area $[m^2]$	40.00
Estimated Dimensions $[m]$	8.17 x 4.72
Estimated Area $[m^2]$	38.56
Area ratio (E/G) $[\%]$	96.4

Table 6.17: Dataset "Synthetic Textured", metrics.

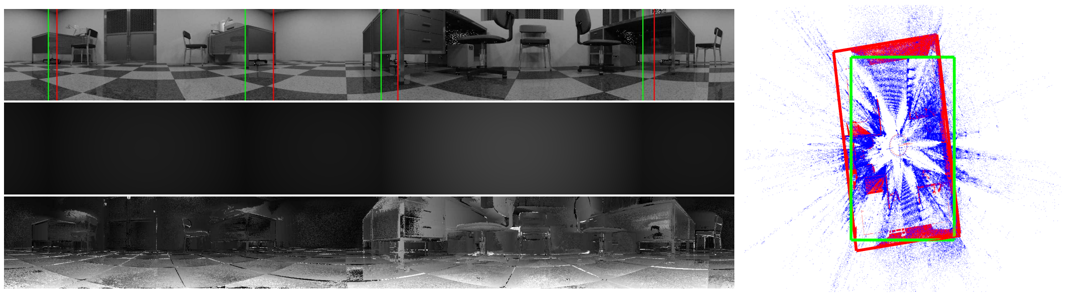


Figure 6.44: Dataset “Synthetic Textureless”, from top to bottom: reference frame, raytraced depth map, raw depth map. On the right: top down view of the point cloud with the box estimate.

Synthetic — Textureless This dataset shows more realistic performance of the method. While the overall estimated dimensions are roughly correct, the uncertainty of the depth estimates is high, therefore the method tends to align to the denser correct estimates at close range, despite the asymmetric cost function punishment for doing so. This may skew the estimate of the angle if the depth estimates that would otherwise provide vanishing line like hints are unavailable.Then the method tries the next best fit and that might not necessarily result in the correct orientation. Implementing a separate vanishing line detector could improve the performance in this regard.

Ground Truth Dimensions $[m]$	8.00 x 5.00
Ground Truth Area $[m^2]$	40.00
Estimated Dimensions $[m]$	7.28 x 4.95
Estimated Area $[m^2]$	36.036
Area ratio (E/G) [%]	90.09

Table 6.18: Dataset “Synthetic Textureless”, metrics.

The synthetic datasets above are rather extreme examples as the total area is $40\,\mathrm{m}^2$, which is unusually large for a typical household room.

Household Data

For the household datasets we have employed the ground truth from Section 6.2.3 — ElasticFusion based 3D models with postprocessing and manual alignment of the coordinate frames, with all its inherent limitations. Dense mapping settings and asymmetric Cauchy function weights were the same throughout all datasets. In these datasets we presented various ground truth approaches. On most rooms, especially small, the ground truth shape of the room was estimated by applying our method to the ground truth depth map (an apprach similar to the one presented in the Section 6.2.3). On larger rooms, especially with multiple incremental steps, we have decided to set ground truth to the actual outer wall-to-wall shape of the room, in the hope that incremental operation would estimate the major part of the ground truth.

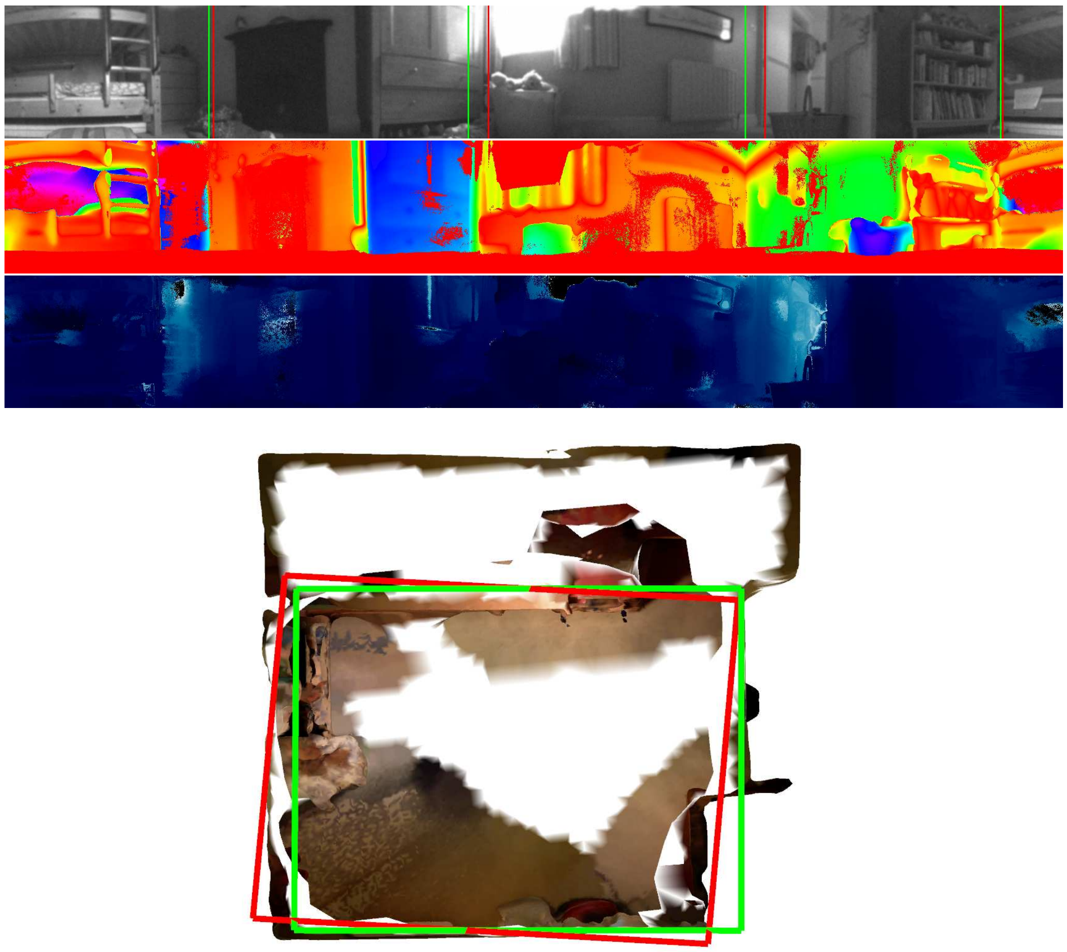


Figure 6.45: Dataset "Ealing Bedroom 1"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

Dataset "Ealing Bedroom 1" This dataset perfectly illustrates the limitation imposed by the purely visual nature of the method. In this room one of the walls is behind a bunk bed, thus for the robot the bunk bed becomes a wall and therefore the results cannot be treated as an architectural blueprint. The ground truth was estimated using a 3D model based depth map and is actually worse than our estimate, due to the poor quality of the 3D model (sparse, mostly filled and smoothed by Poisson Surface Reconstruction).

Ground Truth Dimensions $[m]$	2.63 x 2.49
Ground Truth Area $[m^2]$	6.55
Estimated Dimensions $[m]$	2.57 x 2.47
Estimated Area $[m^2]$	6.36
Area ratio (E/G) [%]	97.19
Intersection area $[m^2]$/[%]	6.03 / 92.16

Table 6.19: Dataset "Ealing Bedroom 1", metrics.

Dataset "Ealing Kitchen" This is another example, where large furniture (cupboard units) that is featured prominently in the field of view becomes the outer boundary for the shape of the room. The ground truth was established from the 3D model based depth map and both estimates fit closely, with minor errors due to some changes in the scene between the time the robot captured the dataset and the time the 3D model was created.

Ground Truth Dimensions $[m]$	1.95 x 3.73
Ground Truth Area $[m^2]$	7.28
Estimated Dimensions $[m]$	1.94 x 3.58
Estimated Area $[m^2]$	6.97
Area ratio (E/G) [%]	95.83
Intersection area $[m^2]$/[%]	6.71 / 92.21

Table 6.20: Dataset "Ealing Kitchen", metrics.

Dataset "Ealing Bedroom 2" In this dataset the ground truth was manually set to the real outer shape of the room to show the discrepancy between true room dimensions and vision based ones. It is worth noting that the raytraced depth map error (created from our box estimate) clearly shows the nature of the asymmetric cost function, as the small furniture (here table and chair) stands out in the error color map, showing how the cost function tries to apprach the "wall".

Dataset "Ealing Livingroom" Here we see the incremental operation of the method on a difficult large scale dataset. Due to the nature of our model (box/rectangle), neither some of our estimates nor the manual ground truth fit the unusual

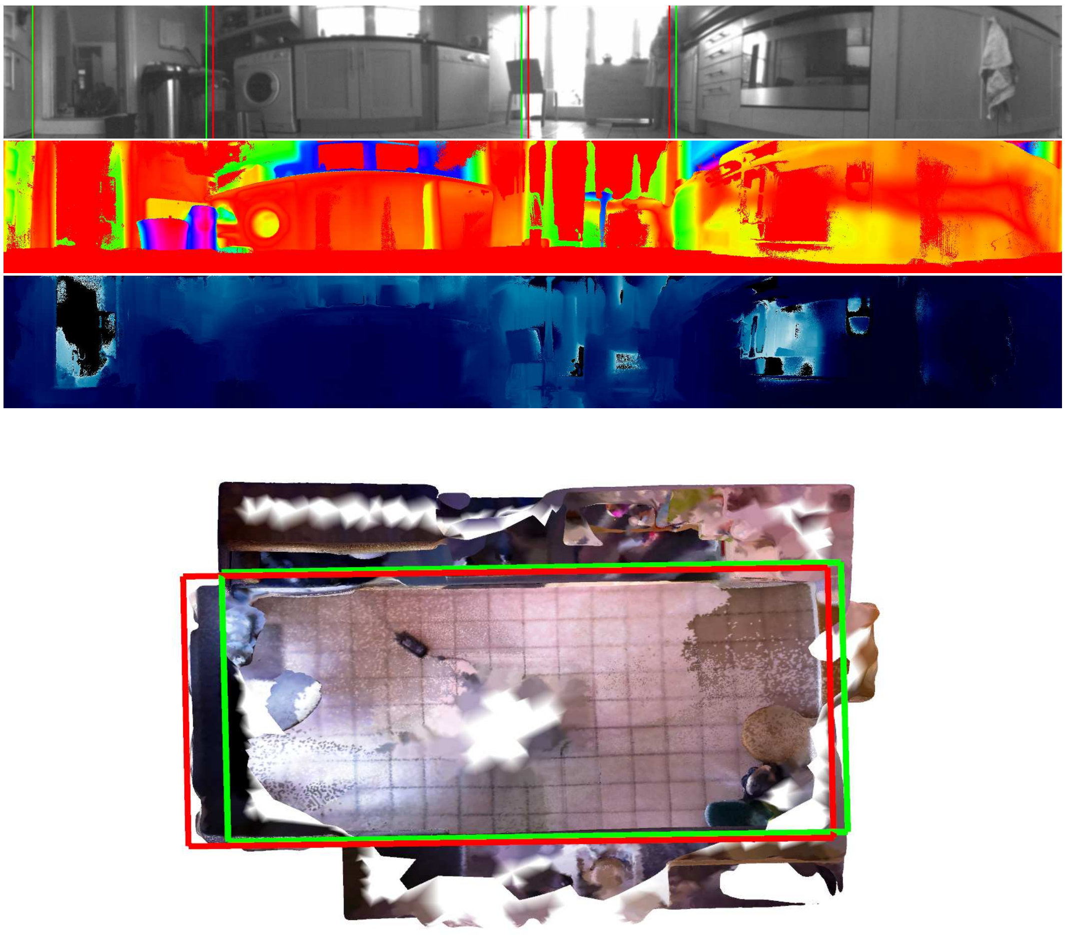


Figure 6.46: Dataset “Ealing Kitchen”; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

Ground Truth Dimensions $[m]$	2.80 x 3.42
Ground Truth Area $[m^2]$	9.60
Estimated Dimensions $[m]$	2.70 x 2.70
Estimated Area $[m^2]$	7.30
Area ratio (E/G) [%]	76.06
Intersection area $[m^2]$/[%]	7.14 / 74.35

Table 6.21: Dataset “Ealing Bedroom 2”, metrics.

shape of the room. This can be seen in the second estimate, where one dimension was following the width of the room (like the first estimate), while the other the

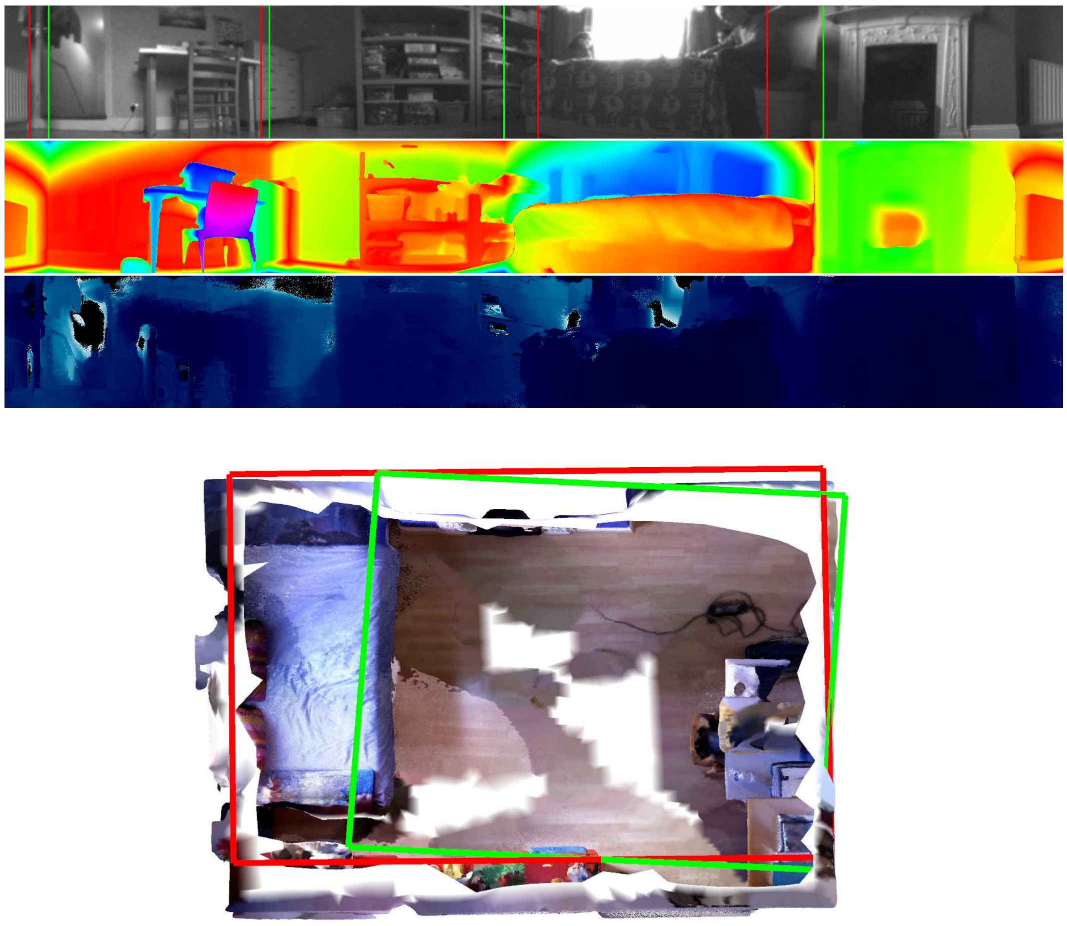


Figure 6.47: Dataset “Ealing Bedroom 2”; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

length (as in the third estimate). Combining these two causes noticeable error.

	Box 1	Box 2	Box 3
Ground Truth Dimensions $[m]$	7.74 x 3.40		
Ground Truth Area $[m^2]$	26.35		
Estimated Dimensions $[m]$	3.31 x 3.47	6.38 x 3.41	5.98 x 2.49
Estimated Area $[m^2]$	11.51	21.79	14.92
Area ratio (E/G) $[\%]$	43.70	82.72	56.62
Intersection area $[m^2]/[\%]$	10.07 / 38.25	10.10 / 38.35	11.17 / 42.40
Box union area $[m^2]/[\%]$	21.51 / 81.65		

Table 6.22: Dataset “Ealing Livingroom”, metrics.

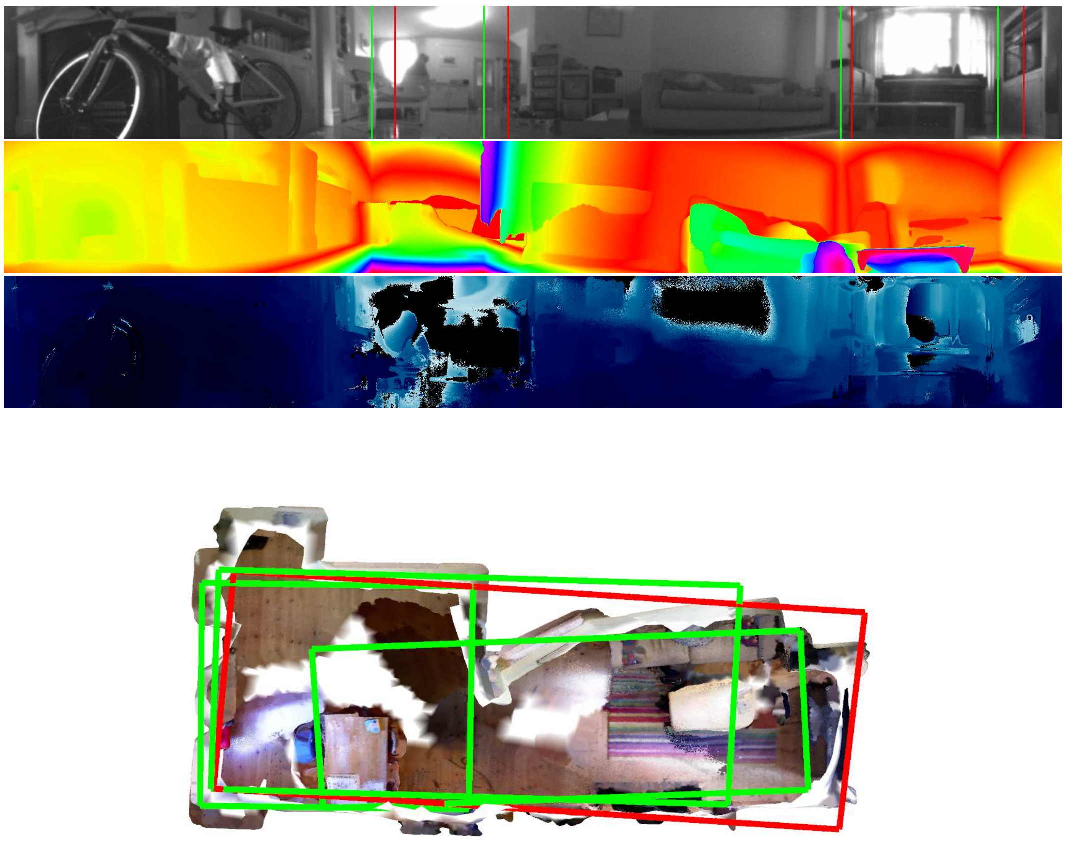


Figure 6.48: Dataset "Ealing Livingroom"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

Dataset "Elephant & Castle Livingroom" Here our method faces the same challenges that were described in the previous experiment (6.3.4). The scale of the room and multiple textureless areas (kitchen furniture, overexposed windows) produce unreliable measurements, thus skewing the box estimation results.

Dataset "Elephant & Castle Bathroom" Bathrooms are extremely challenging datasets for any kind of computer vision method, often even for active sensors (e.g. RGB-D), due to the lack of texture and specularity of virtually all surfaces. The only saving grace is that such rooms are frequently small. In this particular example the side of the bathtub became a wall for the robot, which was to be expected.

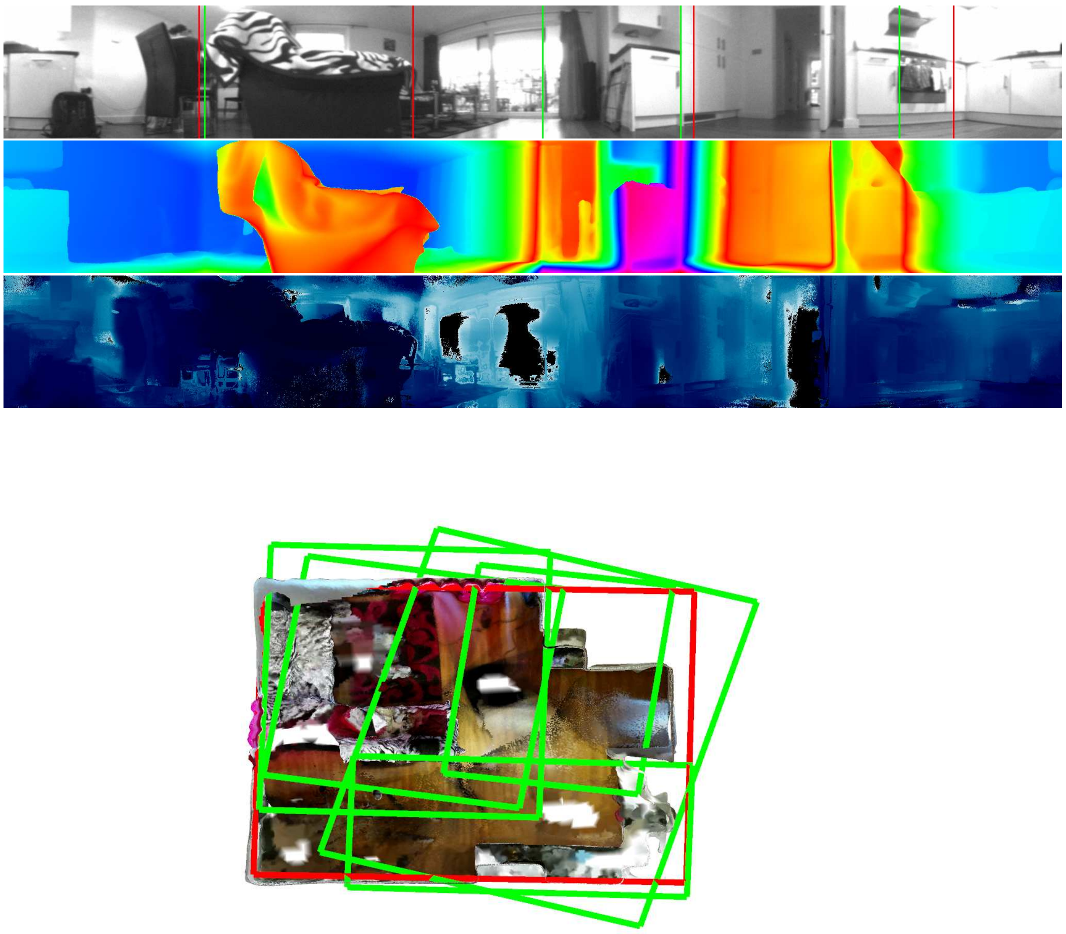
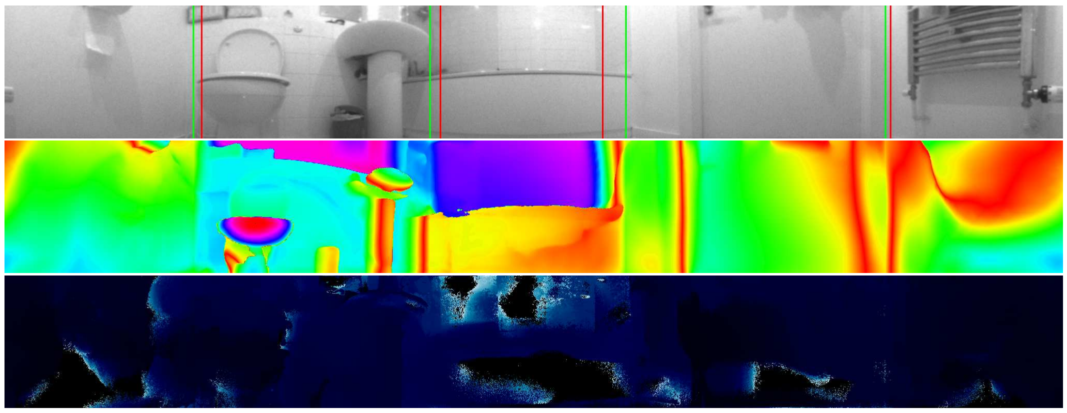


Figure 6.49: Dataset "Elephant & Castle Livingroom"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

	Box 1	Box 2	Box 3	Box 4	Box 5
Ground Truth Dimensions $[m]$	5.69 x 4.78				
Ground Truth Area $[m^2]$	27.2				
Estimated Dimensions $[m]$	3.41 x 3.64	3.68 x 4.36	2.58 x 3.46	4.46 x 2.16	4.38 x 5.54
Estimated Area $[m^2]$	12.46	16.10	8.93	9.65	24.27
Area ratio (E/G) [%]	45.81	59.18	32.83	35.48	98.22
Intersection area $[m^2]$/[%]	11.63 / 42.74	14.19 / 52.17	4.64 / 17.08	9.14 / 33.61	18.63 / 68.47
Box union area $[m^2]$/[%]	34.71 / 127.6				

Table 6.23: Dataset "Elephant & Castle Livingroom", metrics.

Dataset "Elephant & Castle Bedroom 1" As mentioned before, our box model is not always a good fit for real rooms. Here we see that the ground truth of



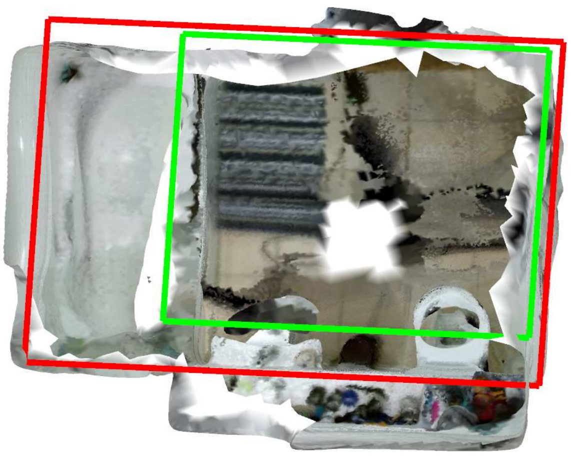


Figure 6.50: Dataset "Elephant & Castle Bathroom"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

Ground Truth Dimensions $[m]$	1.90 x 2.29
Ground Truth Area $[m^2]$	4.37
Estimated Dimensions $[m]$	1.59 x 1.63
Estimated Area $[m^2]$	2.60
Area ratio (E/G) $[\%]$	59.46
Intersection area $[m^2]/[\%]$	2.60 / 59.46

Table 6.24: Dataset "Elephant & Castle Bathroom", metrics.

the room is actually false and our incremental estimation represents the L-shaped room faithfully.

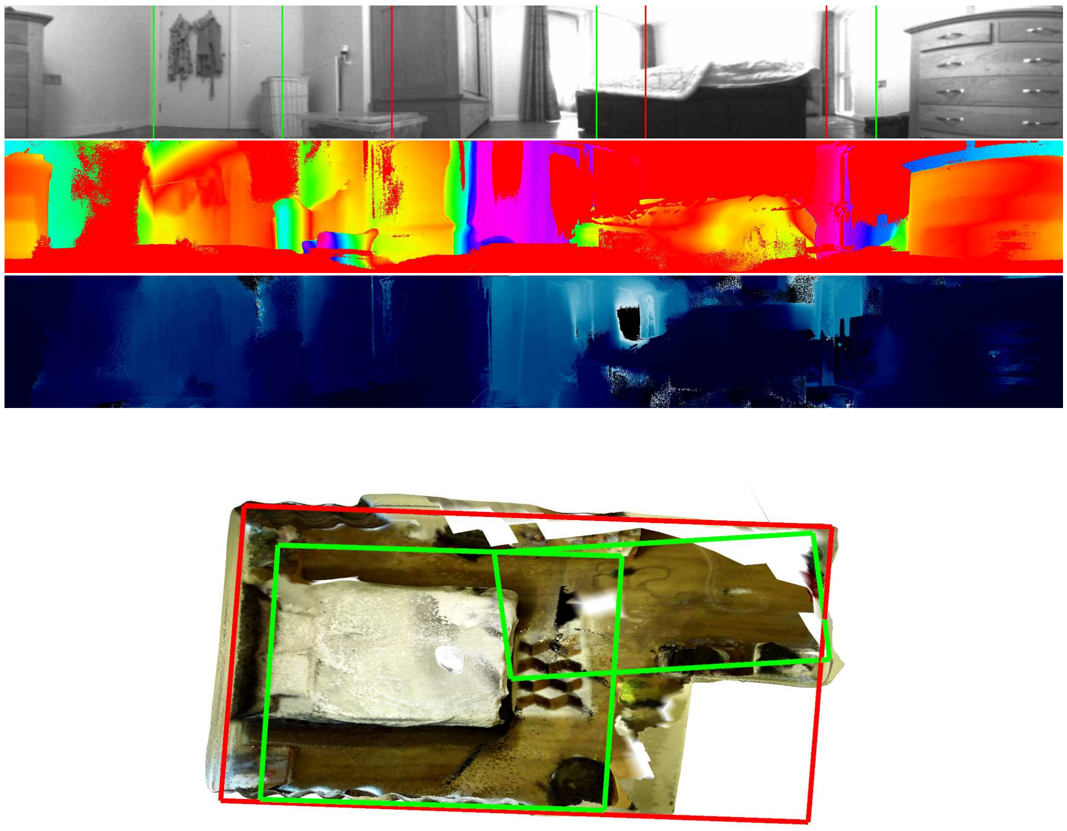


Figure 6.51: Dataset "Elephant & Castle Bedroom 1"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

	Box 1	Box 2
Ground Truth Dimensions $[m]$	3.45 x 4.63	
Ground Truth Area $[m^2]$	15.99	
Estimated Dimensions $[m]$	1.48 x 2.51	2.97 x 2.71
Estimated Area $[m^2]$	3.73	8.07
Area ratio (E/G) $[\%]$	23.34	50.45
Intersection area $[m^2]/[\%]$	3.71 / 23.24	6.80 / 42.56
Box union area $[m^2]/[\%]$	10.70 / 66.88	

Table 6.25: Dataset "Elephant & Castle Bedroom 1", metrics.

Dataset "Elephant & Castle Bedroom 2" Our method has some drawbacks. Due to the purely geometric nature of our optimization sometimes the estimated cube doesn't align with the vanishing lines and fits the depth map instead. In this

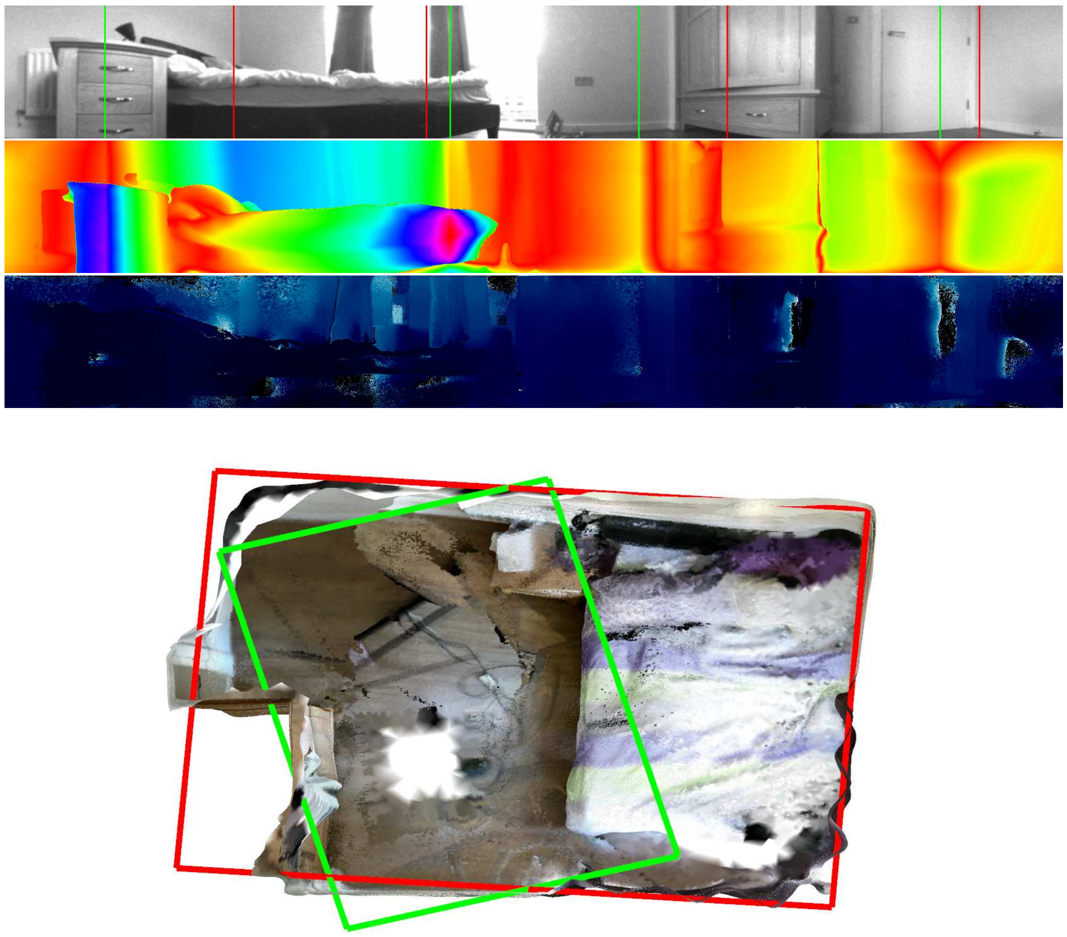


Figure 6.52: Dataset “Elephant & Castle Bedroom 2”; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

experiment the complex structure of the room, combined with the robot’s location (viewpoint) produced an estimate that is roughly correct in size, but erroneously rotated to encompass maximum area. The vanishing lines of interest in our case are primarily the horizontal ones, that are not well estimated due to the planar motion of the robot.

Dataset “Wandsworth Attic” Here our method underestimates the size of the room due to two factors. In one axis the distances are relatively large, and in this axis one wall is rather textureless, while the opposite is almost invisible due to the furniture. On the other axis there is of course prominently featured furniture, but

Ground Truth Dimensions $[m]$	3.49 x 2.65
Ground Truth Area $[m^2]$	9.26
Estimated Dimensions $[m]$	1.82 x 2.60
Estimated Area $[m^2]$	4.74
Area ratio (E/G) [%]	51.19
Intersection area $[m^2]$/[%]	4.55 / 49.18

Table 6.26: Dataset “Elephant & Castle Bedroom 2”, metrics.

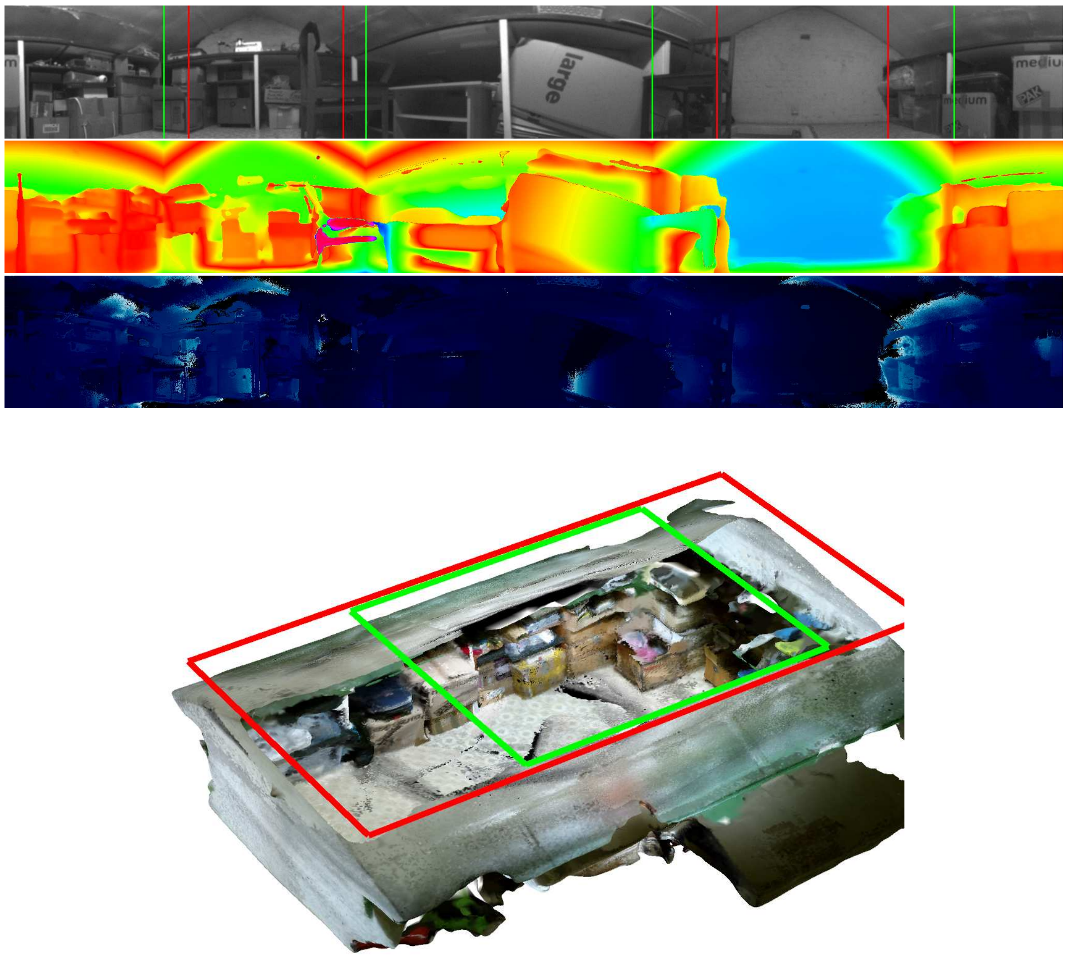
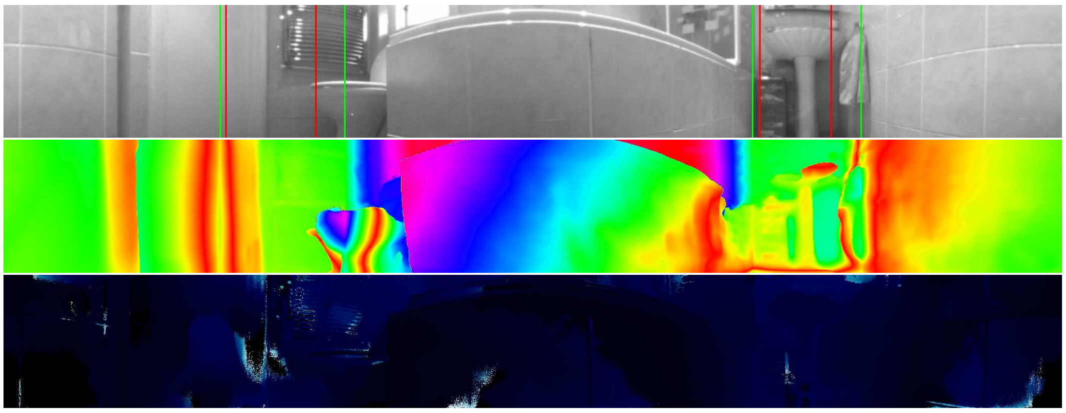


Figure 6.53: Dataset “Wandsworth Attic”; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

also the angled roof of the room diverges massively from our assumed box model influencing the final result.

Ground Truth Dimensions $[m]$	4.43 x 2.31
Ground Truth Area $[m^2]$	10.28
Estimated Dimensions $[m]$	2.44 x 2.16
Estimated Area $[m^2]$	5.28
Area ratio (E/G) [%]	51.39
Intersection area $[m^2]$/[%]	5.28 / 51.39

Table 6.27: Dataset "Wandsworth Attic", metrics.



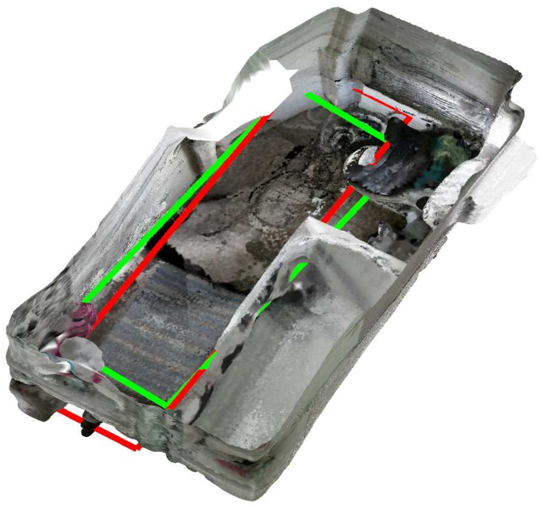


Figure 6.54: Dataset "Wandsworth Bathroom"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

Dataset "Wandsworth Bathroom" As in previous bathroom datasets, these one is no exception, posing challenges even for RGB-D cameras that we have employed to create the 3D model of the room. In a small space like this the bathtub is what the robot sees from its position low on the ground so it becomes the boundary of our and ground truth based estimate. The door is at an angle, therefore an outlier, again for both estimates, for the box model. Despite these adversities the overlap is quite high and the resulting shape is actually meaningful for a small vacuum cleaning robot.

Ground Truth Dimensions $[m]$	0.6 x 2.48
Ground Truth Area $[m^2]$	1.49
Estimated Dimensions $[m]$	0.71 x 1.98
Estimated Area $[m^2]$	1.41
Area ratio (E/G) [%]	94.83
Intersection area $[m^2]$/[%]	1.19 / 79.74

Table 6.28: Dataset “Wandsworth Bathroom”, metrics.

Dataset “Wandsworth Bedroom 1” Similarly to dataset 6.3.4, this one suffered from the nature of the input data (depth maps) to our method. Instead of lining up the box along the vanishing lines, the method seeks the biggest possible fit, rotating the estimate along the major furniture. External vanishing line estimation method could help, although it might be not without difficulty, as we need to have vanishing lines in the image, at some level above or below the horizon, which is not always the case.

	Box 1	Box 2
Ground Truth Dimensions $[m]$	2.85 x 2.42	
Ground Truth Area $[m^2]$	6.92	
Estimated Dimensions $[m]$	3.40 x 0.69	2.69 x 0.93
Estimated Area $[m^2]$	2.36	2.51
Area ratio (E/G) [%]	34.22	36.32
Intersection area $[m^2]$/[%]	1.95 / 28.27	1.67 / 24.19
Box union area $[m^2]$/[%]	3.05 / 44.16	

Table 6.29: Dataset “Wandsworth Bedroom 1”, metrics.

Dataset “Wandsworth Bedroom 2” As noted in the previous sections’ experimental section, linear motion produces high uncertainty depth estimates along the direction of motion. This caused slight misestimation of the room’s length in this experiment. The angular error is actually partialy due to the misalignment of the

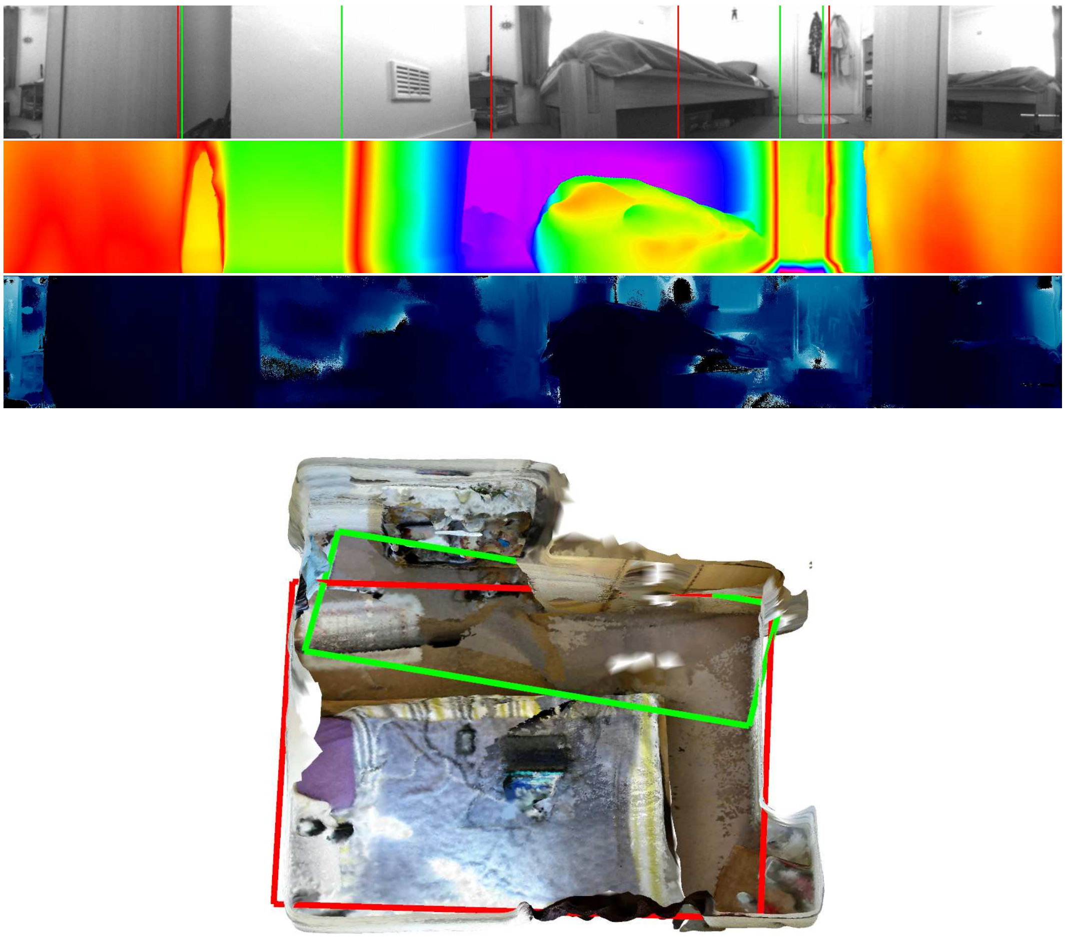
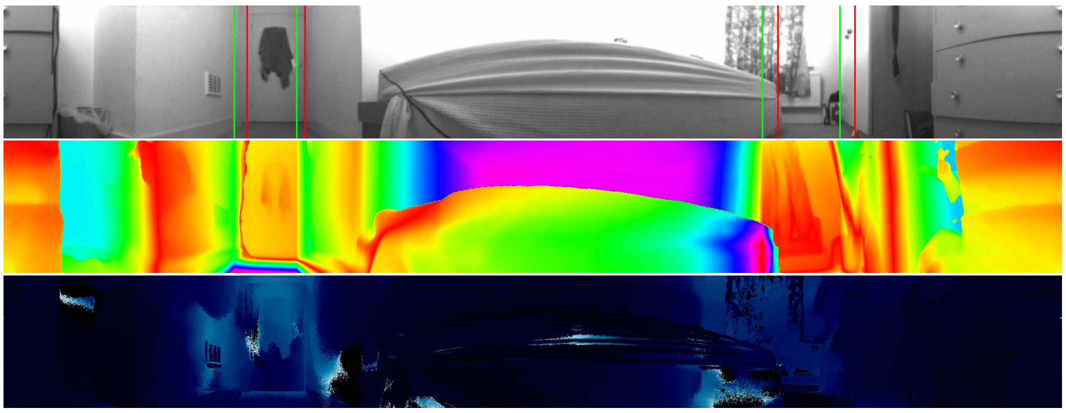


Figure 6.55: Dataset "Wandsworth Bedroom 1"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

3D model to the omnidirectional camera coordinate system, but due to the length of the room the angular errors visibly manifest themselves.

Ground Truth Dimensions $[m]$	4.32 x 0.85
Ground Truth Area $[m^2]$	3.70
Estimated Dimensions $[m]$	3.94 x 0.819
Estimated Area $[m^2]$	3.23
Area ratio (E/G) [%]	87.18
Intersection area $[m^2]/[\%]$	3.01 / 81.36

Table 6.30: Dataset "Wandsworth Bedroom 2", metrics.



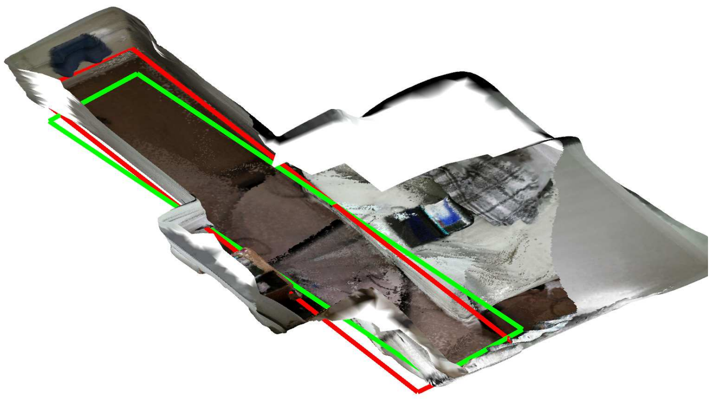
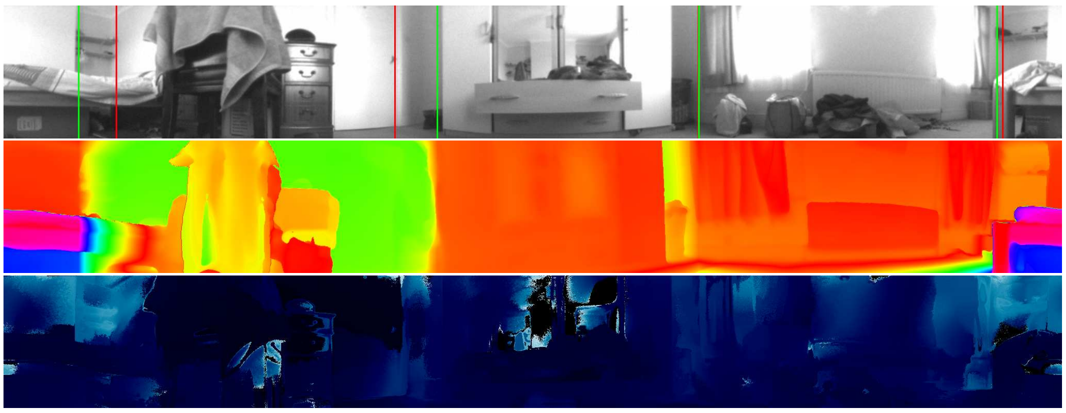


Figure 6.56: Dataset "Wandsworth Bedroom 2"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

Dataset "Wandsworth Bedroom 3" This result is almost a perfect fit, despite the complex nature of the dataset (overexposed windows, mirrors). The only difference when compared to ground truth is caused by the chair/desk being prominent in the field of view and highly textured, while the wall behind is completely textureless thus devoid of depth estimates that our method could rely on.

Ground Truth Dimensions $[m]$	3.09 x 3.98
Ground Truth Area $[m^2]$	12.34
Estimated Dimensions $[m]$	2.17 x 3.53
Estimated Area $[m^2]$	7.68
Area ratio (E/G) $[\%]$	62.24
Intersection area $[m^2]/[\%]$	7.68 / 62.24

Table 6.31: Dataset "Wandsworth Bedroom 3", metrics.



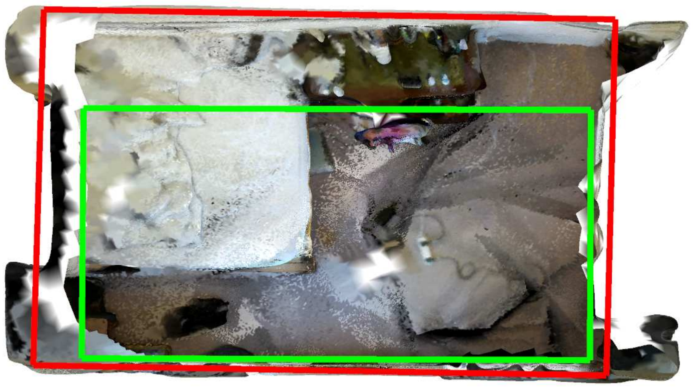


Figure 6.57: Dataset "Wandsworth Bedroom 3"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

Dataset "Wandsworth Kitchen" Despite the unfavourable nature of this dataset (see Section 6.2.3), our method performed correctly. The error on the right hand side was caused by a large number of accurate depth estimates on the vertical edges (right of the bin or around the drawers), while the space in the back was burden with high uncertainty due to the door being open and this side of the room was along the direction of the linear motion.

Dataset "Wandsworth Livingroom" We also wanted to test how our estimates overlap with multiple viewpoints in the same room. As in 6.3.4 the misalignment of the boxes is an inherent drawback of our method in some types of scenes, but the

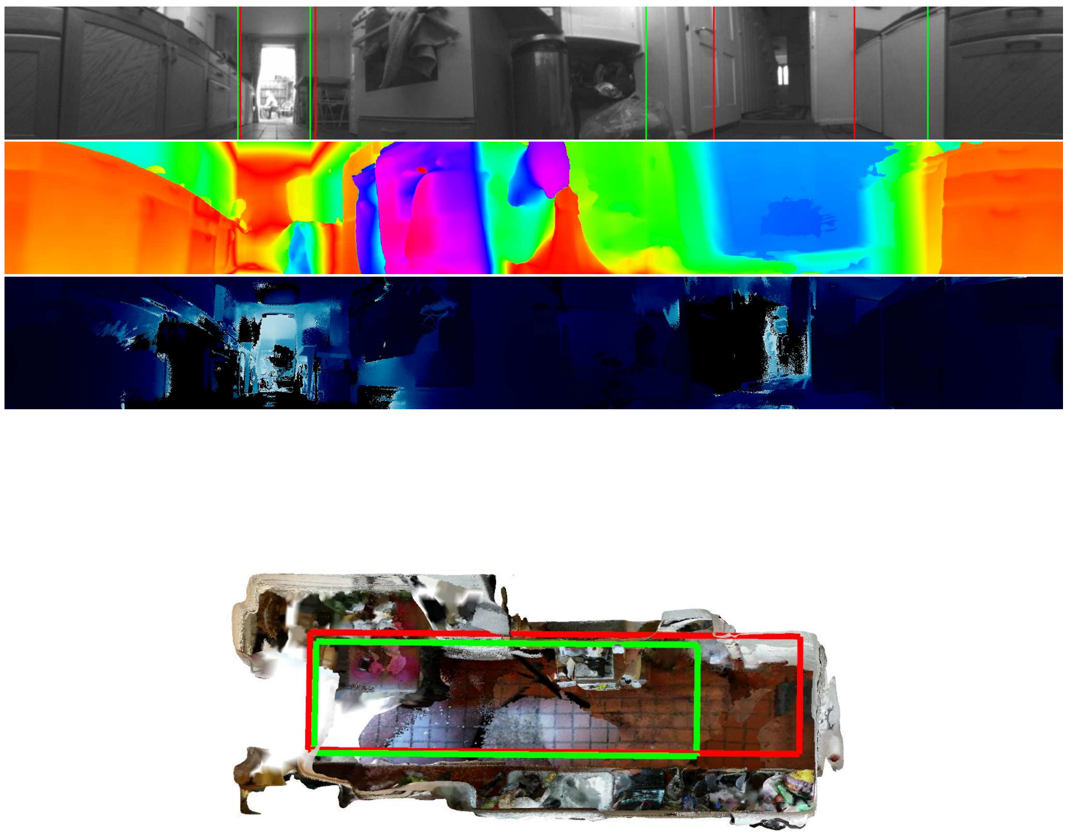


Figure 6.58: Dataset "Wandsworth Kitchen"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

Ground Truth Dimensions $[m]$	4.32 x 1.30
Ground Truth Area $[m^2]$	5.63
Estimated Dimensions $[m]$	3.36 x 1.23
Estimated Area $[m^2]$	4.14
Area ratio (E/G) $[\%]$	73.63
Intersection area $[m^2]/[\%]$	4.03 / 71.68

Table 6.32: Dataset "Wandsworth Kitchen", metrics.

overall overlap of the boxes is very good. It is important to note that one cannot guarantee perfectly overlapping boxes from varying viewpoints, as different parts of the scene would exhibit themselves differently from each viewpoint. This can be seen

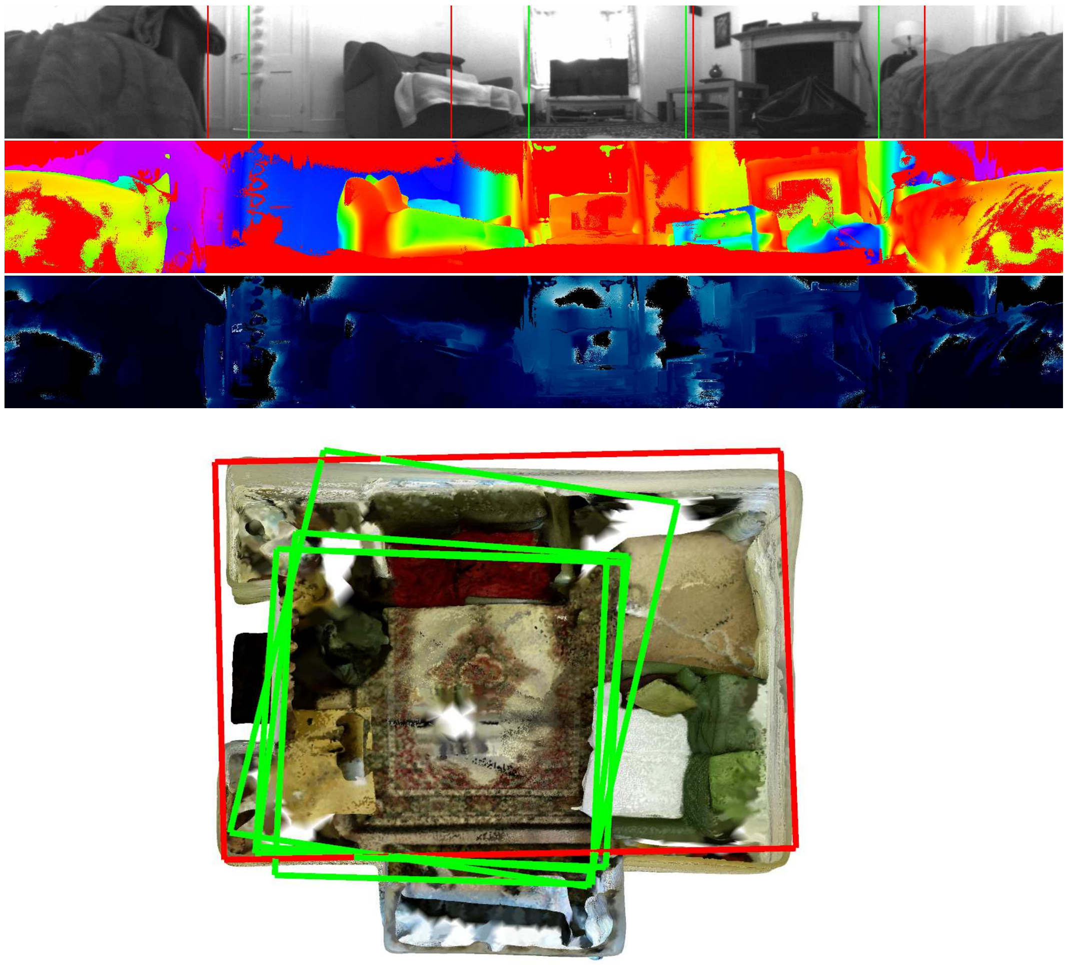


Figure 6.59: Dataset "Wandsworth Livingroom"; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

in the first box, when the robot was far away from the sofa (close to the TV), thus the sofa on the opposite side was not overwhelming large part of our depth map. That is why the estimate reaches the back wall correctly. However, in subsequent viewpoints, in the centre of the room and on the opposite side this scenario no longer holds and the estimates are guided more by the large pieces of furniture around the robot.

Dataset "Wandsworth Lab" It is important to try to understand the world that the robot sees with its omnidirectional camera. While we are aiming in this method to estimate the shape of the room, this is often difficult, as the robot is low

	Box 1	Box 2	Box 3	Box 4
Ground Truth Dimensions $[m]$	3.62 x 3.19			
Ground Truth Area $[m^2]$	11.61			
Estimated Dimensions $[m]$	2.30 x 3.13	2.13 x 2.53	2.22 x 2.44	2.61 x 2.01
Estimated Area $[m^2]$	7.23	5.40	5.43	5.25
Area ratio (E/G) [%]	62.3	46.59	46.83	45.23
Intersection area $[m^2]$/[%]	7.04 / 60.6	4.97 / 42.86	5.43 / 46.83	5.14 / 44.32
Box union area $[m^2]$/[%]	8.07 / 69.55			

Table 6.33: Dataset “Wandsworth Livingroom”, metrics.

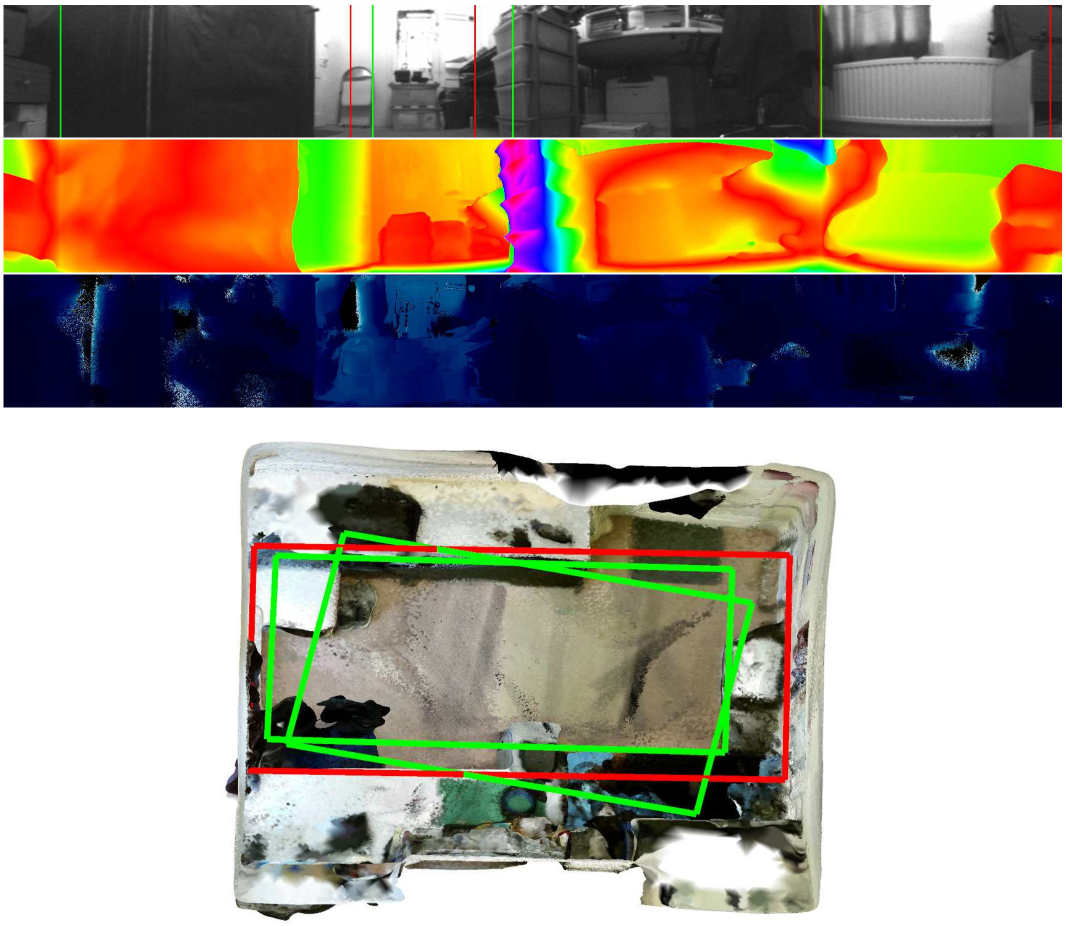


Figure 6.60: Dataset “Wandsworth Lab”; from top to bottom: reference frame, raytraced depth map error, raw depth map, top down view with the box estimate.

on the ground, close to furniture, especially one that exhibits overhangs. This was the case here, thus the top-down view sometimes doesn’t do justice to the method.

While the first square is roughly correct, the second one is reduced in size and misaligned due to two factors. The bedside on the left hand side is the furthest away object that still has discernible texture, thus acceptable depth estimates. The wall behind it is even further away and textureless, the same stands for the back of the upright bed that is entirely black. On the other side, the robot was very close to a workbench, which has an empty cavity below the surface, therefore these under the desk depth estimates allowed the final box to rotate.

	Box 1	Box 2
Ground Truth Dimensions [m]	3.00 x 1.57	
Ground Truth Area [m^2]	4.74	
Estimated Dimensions [m]	2.33 x 1.51	2.55 x 1.28
Estimated Area [m^2]	3.54	3.29
Area ratio (E/G) [%]	74.77	69.61
Intersection area [m^2]/[%]	3.34 / 70.61	2.13 / 44.94
Box union area [m^2]/[%]	3.05 / 44.16	

Table 6.34: Dataset "Wandsworth Lab", metrics.

Office Data

In addition to typical household environments, the system was evaluated on office rooms. These are, in our experience, more challenging as often the amount of texture is substantially lower, the overall scale is larger and there are more specular objects (e.g. big windows, glass walls, panel lighting). Our results are presented in Figures 6.61 and 6.62. Ground truth was obtained with a Bosch PLR15 laser range finder and manually overlaid onto the 3D reconstruction.

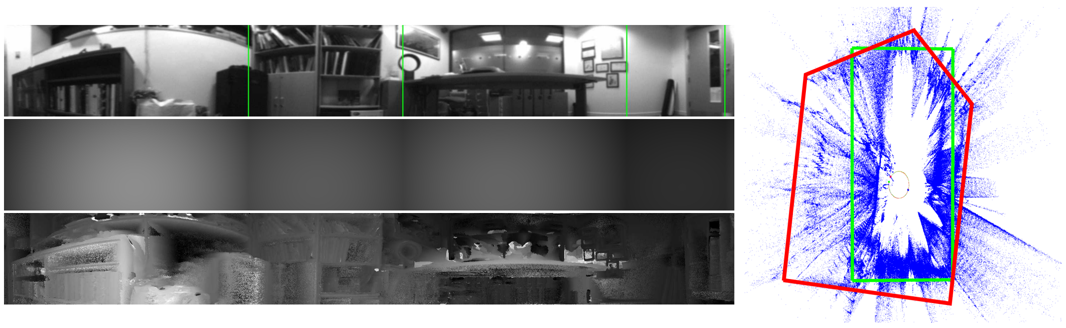


Figure 6.61: Dataset "Cluttered Office"; from top to bottom: reference frame, raytraced depth map, raw depth map. On the right: top down view of the point cloud with the box estimate.

Cluttered Office In this case the room is of pentagonal shape, with one wall curved and another being made of glass. Here the method fits the room model to the most pronounced furniture, bookshelves and the edge of the desk (3.54 [m] $\times$ 2.03 [m]). The ground truth is unkown in this case.

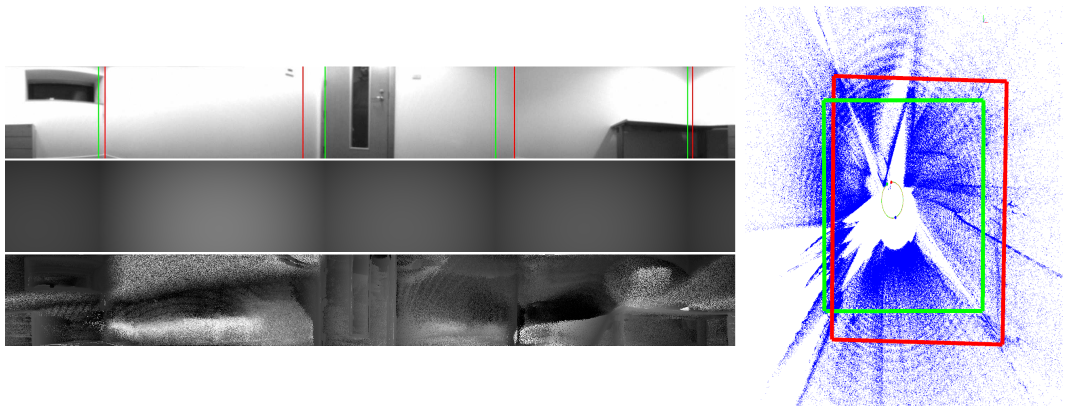


Figure 6.62: Dataset "Empty Office", from top to bottom: reference frame, raytraced depth map, raw depth map. On the right: top down view of the point cloud with the box estimate.

Empty Office This is another dataset aimed to test the method at extreme boundary conditions. Here the room is almost completely without texture. A slight error in one direction is caused by the desk being the only major feature in the room, skewing the estimate.

Ground Truth Dimensions $[m]$	3.14 x 2.87
Ground Truth Area $[m^2]$	9.01
Estimated Dimensions $[m]$	2.90 x 2.29
Estimated Area $[m^2]$	6.64
Area ratio (E/G) [%]	73.69

Table 6.35: Dataset "Empty Office", metrics.

6.4 Conclusions

The two methods described in this chapter introduce novel ways of creating more abstract representations from raw omnidirectional SLAM data. Both methods underwent extensive experimental evaluation on around 20 datasets, ranging from purely synthetic scenes, through office environments to typical residential rooms. Some of the datasets were intended to stress test the methods on the boundary conditions, like bathrooms (textureless and non-lambertian environment) or scenes not conforming to the assumed model (e.g. attic with slanted walls).

The free-space estimation described in Section 6.2, while conceptually rather simple, proved itself very robust when tested in extreme real world conditions, full of non-lambertian surfaces, sensor exposure problems or large scale scenes. We think the algorithm could be a useful addition to omnidirectional camera equipped mobile robots, aiding their sensing capabilities. It is crucially important to note that the major reason behind this method is to quickly, after a short, preferably circular motion and a tens of seconds of computation, uncover $5\,\mathrm{m}^2$ to $7\,\mathrm{m}^2$ of neighbouring free space area almost instantly. This is in contrast to the current operation of, for example, the Dyson 360 Eye, that employs the omnidirectional camera only for tracking, while mapping relies on short range infrared sensors. This means that the robot has to “sniff” around the whole room to build a free space map. Our method brings a massive improvement in this context.

The parametrized model fitting method from Section 6.3 sometimes cannot estimate the true shape of the room, the limitation lies in the purely visual nature and the constraints of the robot (on the floor, limited field of view or resolution). The method itself while robust in general, could benefit from some additional heuristics (e.g. vanishing line estimation). Still in many cases the method produces useful results, both for the visualisation purpose for the end user, but more importantly in our opinion, the representation seems to have a lot of potential for the robot operation itself. Boxes can be estimated incrementally and then boolean unions and intersections can be computed, refining the model of the world. Such polygons could form chain-like descriptors, fed into a machine learning method and aiding room understanding tasks. The possible applications listed in the Section 6.3.3 are definetely worth exploring. The authors also think that the simplicity of the method itself, due to the CPU/GPU cooperation along with Automatic Differention, forming in effect a simple optimization problem, is a great value for possible real world implementations.

It is important to mention that the omnidirectional datasets themself are important contribution of this thesis (available datasets discussed in Section 1.1.6). Capturing and processing these datasets, along with ElasticFusion based ground truth (see Section 2.5) was one of the major efforts in this thesis. While the quantity and variety of these datasets is satisfactory to experimentally verify the methods presented in this chapter, it is, alas, not good enough for any sensible Machine Learning approaches, therefore our methods are purely data driven.

CHAPTER 7

Conclusions

Contents

7.1 Contributions **137**
7.2 Recommendations **139**
7.3 Future Research **141**

7.1 Contributions

This thesis covered the entire computer vision pipeline for an omnidirectional camera with a particular focus on mobile robotics. The majority of work presented was performed to reach the opportunities and methods enjoyed by perspective camera researchers for a decade or more.

The first step on the way to reach "the shoulders of giants" was the analysis of the camera models and camera calibration techniques, as described in Chapter 3. Transforming the raw camera model into a spherical camera model allowed for better memory utilization, access performance and potentially better performance when using algorithms expecting rectangular, uniformly sampled, buffers - methods involving image patches or feature descriptors.

The next step was to obtain accurate Visual Tracking, similar to MonoSLAM or PTAM methods. This was described in Chapter 4 and our method, while not claiming to be a complete, fully-featured, sparse SLAM system, is able to accurately estimate camera poses with correct scale. The accuracy was verified against the VICON system ground truth and the tracking results are within the limits provided in the ground truth system specification (1 mm/1°).

As we found the sparse map representation deficient for subsequent high level understanding tasks, we have implemented an omnidirectional dense mapping method, described in Chapter 5. The most important contribution there is a very comprehensive evaluation of the method, missing from the original paper [NLD11], that brought interesting results. That is the depth map quality is constrained by the optical performance of the lens and sensor, with its limited resolution and dynamic range, as well as our peculiar application (indoor robot moving on a plane). The positive aspect of this means we need to analyse very few frames to get the depth map, as integrating more frames, while being computationally expensive, does not improve the final result. This is highly important on low power mobile platforms.

Before we list our scientific contributions, it is important to mention the system and engineering efforts described in Chapter 2 that enabled some of our contributions and allowed us to perform extensive experimental work along with appropriate sources of ground truth for a thorough evaluation of the methods. These include:

- multiple robotic platforms (see Section 2.1), along with control & capture software (Section 2.2), that enabled us to collect multiple datasets in varying environment,
- wide usage of GPGPU computing (see Section 2.4) that enables efficient dense mapping or estimation tasks,
- usage of Automatic Differentiation (see Section 2.3) was very beneficial during the proof-of-concept checks and rapid prototyping of algorithms,
- VirtualCamera system (see Section 2.5) that enabled us to generate semi-synthetic ground truth (depth) even for residential datasets, without the need for expensive 3D laser scanners.

Our main novel contributions, both aiming for room understanding (see Chapter 6), are occupancy grid based free space mapping from a monocular stereo omnidirectional camera and parametrized model fitting to omnidirectional depth maps.

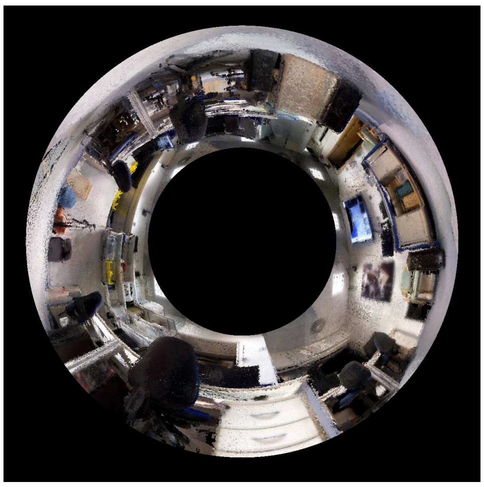
The occupancy grid based free space mapping method, as presented in Section 6.2, while conceptually simple, we think that this method may find its way to the real applications and products really soon. The representation is very familiar to roboticists, with multiple path planning and active exploration methods employing such representation as the input. The rapid nature of free space discovery is a big step forward when compared to the current mapping scheme in the Dyson 360 Eye robot.

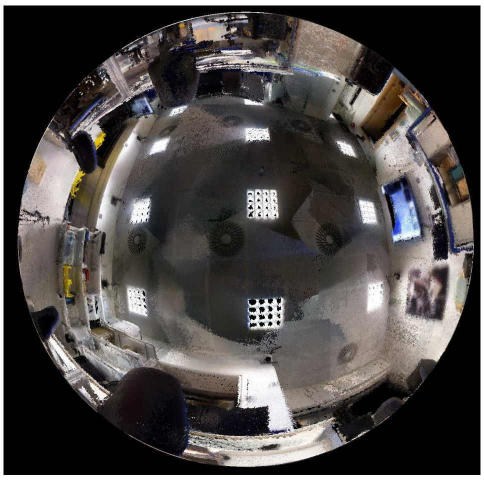
Another contribution is an elaborate parametrized model fitting (see Section 6.3) to a monocular stereo omnidirectional camera depth maps. This enables another level of room understanding, unavailable in simple and coarse occupancy grids. The method proved itself to be very robust to various adversities of the robot's environment. The representation can be very useful for the end user, as the concept of a room is much more familiar than occupancy grids or other robotic representations. Some possible additions/extensions to this method (see Section 6.3.3) could prove themselves useful for the operation of the robot as well.

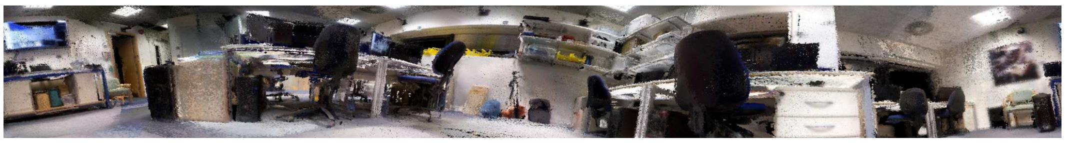
Both methods were extensively tested on a multitude of datasets of high variety, from synthetic scenes to office and residential environments. The engineering efforts mentioned above enabled us to not only capture these datasets, but provide a semi-synthetic ground truth of adequate quality. Some of the datasets were intended to test the boundary condition or method's failure modes (e.g. graceful degradation property).

7.2 Recommendations

At this point we would like to make some recommendations for the reader, if one evaluates computer vision systems. Care must be taken to carefully analyze the physical limitations and properties of the imaging pipeline. Each solution has its inherent strengths and weaknesses. For an omnidirectional camera of the catadioptric variety, as used in this work, the wide angle nature provides a noticeable improvement in tracking quality and might help, for example, relocalization tasks. However, the drawbacks cannot be neglected. We discuss some of them in the Section 3.1, namely worse low light performance (darker lens, low aperture), limited resolution per solid angle and broken automatic gain/exposure/shutter controls due to partial sensor coverage. Additionally, after our experiments with parametrized model fitting, we would rather recommend a fisheye lens, as it has more active pixels and wider field of view that covers the wall-ceiling division and would improve room estimation. This will also improve the imaging performance, as these lenses are more common, manufacturing techniques more reliable and some of the issues from Section 3.1 less pronounced. In Figure 7.1 we present semi-synthetic images (see Section 2.5), generated using camera models estimated from real lenses, each taken at the same viewpoint, and the camera pointing upwards. The benefit of fisheye lens over the catadioptric can be clearly seen and obvious when compared to a perspective camera.


(a) Image from an omnidirectional camera before unwrapping.


(b) View from a fisheye camera (185°×185° field of view).


(c) Panoramic unwrapped omnidirectional image.

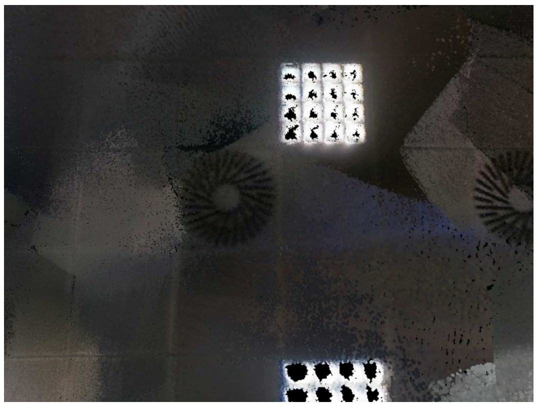

(d) Perspective camera, wide angle (70°×50°) lens.

Figure 7.1: Synthesized views of various camera models.

With any kind of wide angle lens we also would like to recommend using high dynamic range sensors (and global shutter). The issue is that, especially in indoor environments, the wide angle camera is very likely to see both extremely bright parts of the scene (ceiling lights for example), as well as the dark ones (under the table etc.). This does not fare well with classical automatic gain/exposure algorithms and may be alleviated by the higher dynamic range. Additionally, the complex nature of wide angle optics often results in pronounced vignetting that should be corrected for to have output data that the photoconsistency principle can rely on. Therefore, not only geometric calibration (as in Chapter 3) has to be performed, but photometric calibration as well.

7.3 Future Research

As mentioned before, large parts of this thesis were simply devoted to achieving the research environment close to the one available for perspecive cameras. This somehow limited the time available for even higher levels of understanding and more abstract models of the robot's world. Some of the future research ideas were listed in Section 6.3.3, such as incremental room shape estimation, portal detection (doors) and creating contour chain-like descriptors for room shapes for Machine Learning based room recognition.

Another big thing in the future might be related to the new advances in machine learning, namely deep learning. When this work was commenced the field of deep learning was at the turning point, therefore not considered in this thesis. With the recent advances and popularity it might become feasible to perform the higher level understanding with such machine learning methods, requiring only vast amounts of omnidirectional training data.

The author hopes that this work, the techniques and methods described, will be of use for the reader and find its way, in one form or another, to the practical applications and end products, helping to bring a robot to every home.

Bibliography

[AKO] S. Agarwal, Mierle K., and Others. Ceres solver. http://ceres-solver.org. 24, 49, 110

[Bar04] João Pedro de Almeida Barreto. *General central projection systems: Modeling, calibration and visual servoing.* PhD thesis, University of Coimbra, 2004. 6, 37

[BN98] S. Baker and S.K. Nayar. A theory of catadioptric image formation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1998. 4

[BP12] J. C. Bazin and M. Pollefeys. 3-line ransac for orthogonal vanishing point detection. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012. 6

[BVF09] Luigi Bagnato, Pierre Vandergheynst, and Pascal Frossard. A Variational Framework for Structure from Motion in Omnidirectional Image Sequences. Technical Report LTS-2009-013, Signal Processing Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), 2009. 8

[CC15] Alejo Concha and Javier Civera. DPPTAM: Dense Piecewise Planar Tracking and Mapping from a monocular sequence. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015. 7

[CEC15] David Caruso, Jakob Engel, and Daniel Cremers. Large-scale direct SLAM for omnidirectional cameras. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 141–148. IEEE, 2015. 6

[CF14] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 9

[CL85] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1985. 8

[CLSF10] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010. 5, 46

[CP11] A. Chambolle and T. Pock. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. 61

[dB05] Willem den Boer. Chapter 3 - manufacturing of {AM} {LCDs}. In Willem den Boer, editor, *Active Matrix Liquid Crystal Displays*, pages 49 – 85. Newnes, Burlington, 2005. 40

[DCE$^+$11] A Dworak, P Charrue, F Ehm, W Sliwinski, and M Sobczak. Middleware Trends And Market Leaders 2011. *Conf. Proc.*, C111010(CERN-ATS-2011-196):FRBHMULT05. 4 p, Oct 2011. 17

[DMRS07] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007. 5

[Ead14] Ethan Eade. Lie groups for computer vision, 2014. 11

[Elf89] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *EEE Computer*, pages 46–57, June 1989. 8, 78

[ESC14] Jakob Engel, Thomas Schoeps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 6

[Fau93] O. D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993. 5

[FCSS09] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhattan-world stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 9

[FMR11] A. Flint, D. W. Murray, and I. Reid. Manhattan Scene Understanding Using Monocular, Stereo, and 3D Features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 9

[FPS14] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 6

[GD00] Christopher Geyer and Konstantinos Daniilidis. A unifying theory for central panoramic systems and practical applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, ECCV '00, pages 445–461, London, UK, UK, 2000. Springer-Verlag. 4

[Gey03] C.M. Geyer. *Catadioptric Projective Geometry: theory and applications*. PhD thesis, University of Pennsylvania, 2003. 6, 37

[GSB05] G. Grisetti, C. Stachniss, and W. Burgard. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005. 8

[GWSV00] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based Navigation and Environmental Representations with an Omni-directional Camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898, 2000. 5

[Har95] R.I. Hartley. In Defence of the 8-Point Algorithm. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1995. 5

[HHF09] Varsha Hedau, Derek Hoiem, and David A. Forsyth. Recovering the spatial layout of cluttered rooms. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009. 8, 9

[HNAD11] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. Applications of the Legendre-Fenchel transformation to computer vision problems. Technical Report DTR11-7, Imperial College London, 2011. 61

[HNAD12] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. Real-Time Camera Tracking: When is High Frame-Rate Best? In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. 64

[HST10] K. He, J. Sun, and X. Tang. Guided Image Filtering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010. 63

[JM05] W.Y. Jeong and K. Mu Lee. CV-SLAM: A new Ceiling Vision-based SLAM technique. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2005. 6

[KB06] Juho Kannala and Sami S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28:1335–1340, 2006. 39

[KM07] G. Klein and D. W. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007. 6, 48

[KMS01] H. Koyasu, J. Miura, and Y. Shirai. Real-time omnidirectional stereo for obstacle detection and tracking in dynamic environments. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2001. 8

[KMS02] Hiroshi Koyasu, Jun Miura, and Yoshiaki Shirai. Recognizing moving obstacles for robot navigation using real-time omnidirectional stereo vision. *Robotics*, 14(2):147–156, 2002. 5

[KSD$^+$09] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, 2009. 54

[LB84] You-Dong Liang and B. A. Barsky. A new concept and method for line clipping. *ACM Transactions on Graphics*, 3(1):1–22, January 1984. 79

[LB14] Matthew Loper and Michael J. Black. OpenDR: An Approximate Differentiable Renderer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 10

[LCS11] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariance scalable keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 46

[LLD15] R. Lukierski, S. Leutenegger, and A. J. Davison. Rapid Free-Space Mapping From a Single Omnidirectional Camera. In *Proceedings of the European Conference on Mobile Robotics (ECMR)*, 2015. 12

[LLD17] R. Lukierski, S. Leutenegger, and A. J. Davison. Room Layout Estimation from Rapid Omnidirectional Exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 12

[Low99] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1999. 5

[LZCH14] R. Landaverde, Tiansheng Zhang, A. K. Coskun, and M. Herbordt. An investigation of unified memory access performance in cuda. In *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*, pages 1–6, Sept 2014. 23

[MAMT15] R. Mur-Artal, J. M. M Montiel, and J. D. Tardós. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics (T-RO)*, 31(5):1147–1163, 2015. 6, 56

[Mei06] C. Mei. *Laser-Augmented Omnidirectional Vision for 3D Localisation and Mapping*. PhD thesis, INRIA Sophia-Antipolis, 2006. 6

[MGS07] A. C. Murillo, J. J. Guerrero, and C. Sagues. Surf features for efficient robot localization with omnidirectional images. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007. 5

[MHB$^+$10] Elmar Mair, Gregory D. Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2010. 46

[ML00] Don Murray and James J. Little. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8(2):161–171, 2000. 8

[MLS94] R.M. Murray, Z. Li, and S.S. Sastry. *A mathematical introduction to robotic manipulation.* CRC, 1994. 11

[MNS02] J. Miura, Y. Negishi, and Y. Shirai. Mobile robot map generation by integrating omnidirectional stereo and laser range finder. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2002. 8

[MR07] C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2007. 5

[MT97] Tomas Möller and Ben Trumbore. Fast , Minimum Storage Ray / Triangle Intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997. 110

[MT02] C. Martin and S. Thrun. Real-time acquisition of compact volumetric 3d maps with mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002. 8

[NIH$^+$11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 7

[Nis04] D. Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–777, 2004. 5

[NLD11] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 7, 60, 63, 74, 138

[Poc08] T. Pock. *Fast Total Variation for Computer Vision.* PhD thesis, Graz University of Technology, 2008. 7

[QCG$^+$09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. *ICRA workshop on open source software*, 3(3.2):5, 2009. 20

[RD06] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006. 5, 46

[SBC$^+$09] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *International Journal of Robotics Research (IJRR)*, 28(5):595–599, 2009. 10

[Sca08] D. Scaramuzza. *Omnidirectional vision: from calibration to robot motion estimation.* PhD thesis, ETH Zurich, 2008. 6, 43

[Sch14] Miriam Schönbein. *Omnidirectional Stereo Vision for Autonomous Vehicles.* PhD thesis, 2014. 8, 10, 43

[SCN05] A. Saxena, S.H. Chung, and A.Y. Ng. Learning Depth from Single Monocular Images. In *Neural Information Processing Systems (NIPS)*, 2005. 7

[SMD12] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual SLAM: Why filter? *Image and Vision Computing (IVC)*, 30(2):65–77, 2012. 48

[SP02] Tomáš Svoboda and Tomáš Pajdla. Epipolar geometry for central catadioptric cameras. *International Journal of Computer Vision (IJCV)*, 49(1):23–37, 2002. 4

[SS01] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision (IJCV)*, 47:7–42, 2001. 7

[SS07] D Scaramuzza and R Siegwart. *Vision Systems Book*, chapter A Practical Toolbox for Calibrating Omnidirectional Cameras. 2007. 5

[SSN09] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(5):824–840, May 2009. 7

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Cambridge: MIT Press, 2005. 80

[Thr03] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, September 2003. 8

[TK09] Hideyuki Tamura and Hirokazu Kato. Proposal of international voluntary activities on establishing benchmark test schemes for AR/MR geometric registration and tracking methods. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009. 10

[TMHF99] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. In *Proceedings of the International Workshop on Vision Algorithms, in association with ICCV*, 1999. 6

[TPD08] J. P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2008. 5

[Var74] V. Varadarajan. *Lie Groups, Lie Algebras and their Representations.* Springer-Verlag, 1974. 11

[Wen64] R. E. Wengert. A simple automatic derivative evaluation program. *Commun. ACM*, 7(8):463–464, August 1964. 21

[WHH$^+$11] K. M. Wurm, D. Hennes, D. Holz, R. Rusu, C. Stachniss, K. Konolige, and W. Burgard. Hierarchies of octrees for efficient 3D mapping. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2011. 7

[WLSM$^+$15] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015. 26, 85

[WMK$^+$12] T. Whelan, J. B. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *Workshop on RGB-D: Advanced Reasoning with Depth Cameras, in conjunction with Robotics: Science and Systems*, 2012. 7

[XEOT12] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 10

[YK05] Kuk-Jin Yoon and In-So Kweon. Locally Adaptive Support-Weight Approach for Visual Correspondence Search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 63

[Zha00] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22:1330–1334, 2000. 5

[ZLD13] Jacek Zienkiewicz, Robert Lukierski, and Andrew J. Davison. Dense, Auto-Calibrating Visual Odometry from a Downward-Looking Camera. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2013. 12

[ZSTX14] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 9