3D Sensor Placement and Embedded Processing for People Detection in an Industrial Environment

Joacim Dybedal

UiA University of Agder

Doctoral Dissertations at the University of Agder 433

Joacim Dybedal

3D Sensor Placement and Embedded Processing for People Detection in an Industrial Environment

Doctoral Dissertation for the degree *Philosophiae Doctor (Ph.D.)*
at the Faculty of Engineering and Science, Specialisation in Mechatronics

University of Agder
Faculty of Engineering and Science
2023

Doctoral Dissertations at the University of Agder 433

ISSN: 1504-9272
ISBN: 978-82-8427-150-7

Printed by Make!Graphics
Kristiansand

Preface and Acknowledgements

This thesis presents the results of the Ph.D. project in work package 3.1 in the research centre SFI Offshore Mechatronics at the University of Agder, Grimstad, Norway. The project started in August 2016 and was completed in June 2023. I am grateful for the opportunity to work on this project, which has widely broadened my knowledge of relevant and interesting topics such as machine vision and artificial intelligence. To be able to work with such topics from a time when they were still buzz-words until the recent release of AI services such as ChatGPT to the public has been humbling and extremely rewarding. And after a period of time with a global pandemic and the addition of two family members along the way, I am proud to finally submit this thesis and complete this important chapter of my life, and I am excited to see what the future holds.

I would like to extend my special thanks to Professor Geir Hovland, who has been my main supervisor throughout this entire project. Your guidance and support have been excellent, and I have been inspired by your broad knowledge and expertise.

I would also like to thank Dr. Atle Aalerud for his cooperation in establishing the Industrial Robotics Lab at the University of Agder, where we both did most of our research. The time at the university would not be the same without you, and I am proud of the results we accomplished.

To my girlfriend and partner in life, Tina, and the rest of my family: Thank you for all your love, support and patience through this period of my life and beyond.

Lastly, I would like to thank NOV Norway AS for giving me the opportunity to take on this project, Wei Zhao and Kai Erik Nilsen for the help with assembling the prototype sensors, and all the fellow Ph.D. candidates at the SFI OM for their contribution to a fun and rewarding work environment.

The research presented in this thesis has received funding from the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896.

Joacim Dybedal
Kristiansand, Norway
June 2023

Abstract

At a time when autonomy is being introduced in more and more areas, computer vision plays a very important role. In an industrial environment, the ability to create a real-time virtual version of a volume of interest provides a broad range of possibilities, including safety-related systems such as vision based anti-collision and personnel tracking. In an offshore environment, where such systems are not common, the task is challenging due to rough weather and environmental conditions, but the result of introducing such safety systems could potentially be lifesaving, as personnel work close to heavy, huge, and often poorly instrumented moving machinery and equipment.

This thesis presents research on important topics related to enabling computer vision systems in industrial and offshore environments, including a review of the most important technologies and methods. A prototype 3D sensor package is developed, consisting of different sensors and a powerful embedded computer. This, together with a novel, highly scalable point cloud compression and sensor fusion scheme allows to create a real-time 3D map of an industrial area.

The question of where to place the sensor packages in an environment where occlusions are present is also investigated. The result is algorithms for automatic sensor placement optimisation, where the goal is to place sensors in such a way that maximises the volume of interest that is covered, with as few occluded zones as possible. The method also includes redundancy constraints where important sub-volumes can be defined to be viewed by more than one sensor.

Lastly, a people detection scheme using a merged point cloud from six different sensor packages as input is developed. Using a combination of point cloud clustering, flattening and convolutional neural networks, the system successfully detects multiple people in an outdoor industrial environment, providing real-time 3D positions.

The sensor packages and methods are tested and verified at the Industrial Robotics Lab at the University of Agder, and the people detection method is also tested in a relevant outdoor, industrial testing facility. The experiments and results are presented in the papers attached to this thesis.

Oppsummering

I en tid hvor autonomi blir introdusert i stadig flere områder, spiller maskinsyn en svært viktig rolle. I en industriell kontekst gir evnen til å skape en sanntids virtuell versjon av et interesseområde en rekke muligheter, inkludert sikkerhetssystemer som baserer seg på maskinsyn for å unngå kollisjoner og for å spore personell. I en offshore kontekst, der slike systemer ikke er vanlige, er oppgaven utfordrende på grunn av tøffe vær- og miljøforhold, men å introdusere slike sikkerhetssystemer vil potensielt kunne redde liv, da personell ofte må arbeide nær tunge, store og dårlig instrumenterte maskiner og utstyr.

Denne avhandlingen presenterer forskning på viktige emner knyttet til muliggjøring av maskinsyn-systemer i industrielle og offshore miljøer, inkludert en gjennomgang av de viktigste teknologiene og metodene. Det er utviklet en prototyp på et 3D-sensorsystem, bestående av pakker med ulike sensorer og en innebygd datamaskin. Dette, sammen med en ny og svært skalerbar punktskykomprimerings- og sensorfusjons-metode, gjør det mulig å skape en sanntids 3D-kartlegging av et industrianlegg.

Spørsmålet om hvor man skal plassere sensorpakkene i et miljø der det forekommer objekter som kan hindre sikt, er også undersøkt. Resultatet er algoritmer for optimalisering av automatisk plassering av sensorer, hvor målet er å plassere sensorer slik at mest mulig av interesseområdet blir observert, med så få skjulte soner som mulig. Metoden inkluderer også mulighet for å legge inn krav om redundans, der delvolumer kan defineres som ekstra viktige for å bli overvåket av mer enn én sensor.

Til slutt er det utviklet en medtode for detekjon av personer ved bruk av en sammenslått punktsky fra seks ulike sensorpakker. Ved hjelp av en kombinasjon av filtrering av punktskyen, transformering til bilder, samt maskinlæring, kan systemet vellykket detektere flere personer i et utendørs industrianlegg og angi sanntids posisjoner i tre dimensjoner.

Sensorpakkene og metodene er testet og verifisert ved Industrial Robotics Lab ved Universitetet i Agder, og persondeteksjons-metoden er også testet på data fra et relevant utendørs industrianlegg. Eksperimentene og resultatene presenteres i artiklene vedlagt denne avhandlingen.

Publications

The following papers are included as part of this thesis.

Paper A Joacim Dybedal and Geir Hovland. Optimal placement of 3D sensors considering range and field of view. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1588–1593, Munich, 2017.
doi: 10.1109/AIM.2017.8014245.

Paper B Joacim Dybedal and Geir Hovland. GPU-Based Optimisation of 3D Sensor Placement Considering Redundancy, Range and Field of View. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1484–1489, Kristiansand, Norway, 2020.
doi: 10.1109/ICIEA48937.2020.9248170.

Paper C Joacim Dybedal and Geir Hovland. GPU-Based Occlusion Minimisation for Optimal Placement of Multiple 3D Cameras. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 967–972, Kristiansand, Norway, 2020.
doi: 10.1109/ICIEA48937.2020.9248399.

Paper D Joacim Dybedal, Atle Aalerud and Geir Hovland. Embedded Processing and Compression of 3D Sensor Data for Large Scale Industrial Environments. *Sensors*, 19(3):636, 2019.
doi: 10.3390/s19030636.

Paper E Joacim Dybedal and Geir Hovland. CNN-based People Detection in Voxel Space using Intensity Measurements and Point Cluster Flattening. *Modeling, Identification and Control*, 42(2):37–46, 2021.
doi: 10.4173/mic.2021.2.1.

In addition to the papers included in this thesis, the author has also contributed to the following publications as part of the research done in the SFI Offshore Mechatronics research centre.

Atle Aalerud, Joacim Dybedal and Geir Hovland. Scalability of GPU-Processed 3D Distance Maps for Industrial Environments. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Oulu, Finland, 2018. doi: 10.1109/MESA.2018.8449160.

Erind Ujkani, Joacim Dybedal, Atle Aalerud, Knut Berg Kaldestad and Geir Hovland. Visual Marker Guided Point Cloud Registration in a Large Multi-Sensor Industrial Robot Cell. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Oulu, Finland, 2018. doi: 10.1109/MESA.2018.8449195.

Atle Aalerud, Joacim Dybedal and Geir Hovland. Industrial Environment Mapping Using Distributed Static 3D Sensor Nodes. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Oulu, Finland, 2018. doi: 10.1109/MESA.2018.8449203.

Atle Aalerud, Joacim Dybedal and Geir Hovland. Automatic Calibration of an Industrial RGB-D Camera Network using Retroreflective Fiducial Markers. *Sensors*, 19(7) 2019. doi: 10.3390/s19071561.

Atle Aalerud, Joacim Dybedal and Dipendra Subedi. Reshaping Field of View and Resolution with Segmented Reflectors: Bridging the Gap Between Rotating and Solid-State LiDARs. *Sensors*, 20(12) 2020. doi: 10.3390/s20123388.

Joacim Dybedal. Replication Data for: CNN-based People Detection in Voxel Space using Intensity Measurements and Point Cluster Flattening. In *DataverseNO*, 2021. doi: https://doi.org/10.18710/HMJVFM.

Atle Aalerud and Joacim Dybedal. "Reflector for reflecting electromagnetic waves from a rotating electromagnetic wave source", European Patent EP3899604B1, Mar. 29, 2023.

Contents

List of Figures

Chapter 1

Introduction

This chapter presents the main research questions and methods used in this Ph.D. project, and the motivation behind them. It further outlines the structure of the thesis, and presents a state-of-the-art analysis of relevant topics that places the project into context. Lastly, the contributions of the papers that were published during the project are summarised.

1.1 Motivation and Problem Statement

In 2016, when the oil and gas industry was facing hard times, there was a growing need for products that could reduce costs, improve safety, and improve efficiency. A machine vision 3D sensor that could be used in offshore environments could help improve these fields in many scenarios. From the safety perspective of detecting people on a drill floor, to monitoring or measuring machines and objects, a 3D sensor would be a versatile and helpful tool for the oil and gas engineers when designing the rig of the future.

This Ph.D. project was defined as a part of the "Robotics and Automony" work package at the Centre for Research-based Innovation "SFI Offshore Mechatronics" at the University of Agder [1]. Being a research centre with many partners from the offshore industry, this sent a statement that the introduction of computer vision to offshore environments was highly desirable. The main goal of the project was to investigate and prototype a 3D sensor package that could be operational in an offshore environment, including day/night, rough weather conditions, and so on.

The project was initialised by the research partners and industry partners in close collaboration. As an industry-driven project, it was expected to create demonstrable results using off-the-shelf components, prototyping and lab testing. The Technology Readiness Level (TRL) [2] is often used to describe the development phases of a technology (see Table 1.1). While a typical Ph.D. thesis may present technology or theory at level 2 or below, the work done in this thesis presents, and was expected

to present, technology at level 4 to 5.

Table 1.1: Technology Readiness Level - EU version [2]

TRL	Description
1	Basic principles observed
2	Technology concept formulated
3	Experimental proof of concept
4	Technology validated in lab
5	Technology validated in relevant environment
6	Technology demonstrated in relevant environment
7	System prototype demonstration in operational environment
8	System complete and qualified
9	Actual system proven in operational environment

1.1.1 Main Research Questions

Given the task of creating a 3D sensor package for offshore use, and after analysing the state-of-the art, the following research questions were formulated:

- Can a 3D sensor package suitable for offshore environments be developed, using existing technologies?
- Is it possible to automatically optimise the placement of such 3D sensors on a given offshore rig?
- Is it possible to robustly and accurately fuse data from different sensors in an offshore environment?
- Is it possible to develop a simple and efficient calibration method for the sensors selected for this project?
- Can deep learning and neural networks assist sensor fusion, calibration and/or human detection in offshore environments?

1.1.2 Research Methods

To develop the 3D sensor package, in addition to the theoretical work on algorithms and software, extensive practical work and prototyping was expected. Based on the research questions, the project was divided into the following sub-tasks:

- Develop a prototype 3D sensor package for offshore environments based on existing sensor technologies, and test it in both indoor and realistic outdoor environments.

- Develop a computational platform, which may include looking at embedded solutions, FPGAs (Field Programmable Gate Arrays) and/or GPUs (Graphic Processing Units).
- Develop a sensor calibration and fusion scheme for the chosen instrumentation, possibly using machine learning.
- Develop a people detection method, possibly using machine learning.
- Develop an algorithm for 3D sensor layout optimisation for an offshore rig environment, using Mixed Integer Linear Programming.

Based on the defined research questions and methods, it was expected that several novelties and publishable results could be created in this project.

1.2 Thesis Outline

In addition to the introduction and concluding remarks, the thesis is divided into four main chapters, each covering the four main topics of the project: Chapter 2 – 3D Optimisation, Chapter 3 – 3D Sensors and Data Structures, Chapter 4 – Experimental Setup and Prototyping, and Chapter 5 – People Detection. These chapters aim to provide the reader with a broader context and relevant background information for the appended papers.

In addition, as the research into these topics are rapidly progressing and much new work has been published since the start of this PhD project, the chapters also touch on new, relevant state-of-the-art techniques. The appendices A to E contain the full-text papers that were published during this project.

The chapters are further described below:

Chapter 1 – Introduction

The introduction establishes the research questions and motivation for the research presented in this thesis. The chapter includes the preliminary state-of-the-art review on 3D sensor technologies conducted in the startup phase of this project, as well as a summary of the appended papers and their contributions.

Chapter 2 – 3D Optimisation

This chapter gives relevant background information on the optimisation methods used in the appended papers A, B and C.

Chapter 3 – 3D Sensors and Data Structures

Here, an overview of different 3D sensors and their applications and concepts are presented, both as background information for the appended papers and

as an update on the state-of-the-art 3D sensor technologies. The chapter also investigates different data structures for storing 3D data, and techniques for point cloud compression and filtering as used in Paper D.

Chapter 4 – Experimental Setup and Prototyping Presents the experimental setup and 3D sensor package prototype that was used throughout this project.

Chapter 5 – People Detection
This chapter presents a summary of different people detection techniques in both two and three dimensions and using different input data, as a relevant background for the final appended paper, Paper E.

Chapter 6 – Concluding Remarks
The Concluding Remarks summarises the project as a whole and the main conclusions of the published papers. The chapter also looks at possible future work on the research presented in this thesis.

Appended Papers
The appendices contain the full-text version of the published papers listed on page xi. The papers have been modified to fit the format of this thesis and some small spelling mistakes have been corrected, but they are otherwise identical to their published versions.

1.3 Preliminary Analysis of the State-of-the-art on 3D Sensor Technology

In rapidly developing fields such as 3D sensors, technologies and methods emerge and are being improved at a very fast pace. To place the research presented in the appended papers in this thesis into the correct context, the author includes this yet unpublished state-of-the-art review performed at the start of the project in 2016.

Updated references as of 2022 for the topics 3D Sensors, Data Structures and People Detection are given later in Chapters 3 and 5.

With respect to developing a 3D sensor package, several topics were considered: Firstly, the sensor technology itself including different computational platforms had to be evaluated for use in offshore environments. Secondly, algorithms to robustly and efficiently fuse and calibrate sensory data had to be implemented. And lastly, a method to optimise where to place the sensors to reduce cost and/or ensure redundancy. In the next sections, the overview of the state-of-the-art as of 2016 of these topics is given.

1.3.1 3D Sensors for Offshore Environments

Sensors for three-dimensional data acquisition were being used in a wide variety of fields, from medicine to automotive. Apart from some work on 3D sensors for offshore Remotely Operated Vehicles (ROVs), no published work was found on 3D sensors for offshore (drilling) rigs. For this project, industrial robotics and automotive sensors were therefore considered, as they are used in conditions that most closely resemble offshore conditions.

When looking at these industries, the main sensors that were being used were acoustic, radar, laser scanners and lidars (light based range sensors), as well as optical sensors (cameras) [3]. Acoustic sensors have short range and are mostly used for parking assistant systems, and was therefore deemed not suitable for this project.

Radars could be a good choice for all-weather and rough environments, as they are robust in rainy, foggy or night-time conditions. Traditionally, radars with one antenna can only detect the range and speed of an object, not the direction to the object. 3D imaging using multiple-input-multiple-output (MIMO) radars in different forms were starting to gain research interest [4]. But with relatively low angular resolution, radars may only be useful to detect a region of interest, and may need to be supplemented by cameras for object detection [3]. However, in 2016, Tesla Motors announced that they would use their on-board radar as the primary sensor for its Autopilot system, meaning radar was still part of the state-of-the-art in automotive sensor packages [5].

LiDAR, or lidar (Light Detection And Ranging) sensors, with their high angular resolution, were being adopted by autonomous car developers such as Uber [6]. Lidar scanners range from 2D line scanners to rotating 360 degree 3D scanners such as the top-of-the-line Velodyne lidars. They use time-of-flight technology to determine the distance to an object, and the known horizontal and vertical angles of the laser beam to determine the direction, and they are also reasonably robust against outdoor weather conditions [7]. Lidars were very expensive compared to radars and cameras, but prototypes like the MIT "Lidar-on-a-chip" or the Quanergy solid state lidar were, and are still, being developed, promising a much lower price than currently available mechanical sensors [8].

One of the drawbacks with radars and lidars, is that they cannot see colour. Colours are useful for distinguishing (recognising) different objects. Optical cameras have this advantage, and combined with stereo vision or projected structured light, they can also be used as a stand-alone sensor to obtain the distance to an object, creating what is called an RGB-D image (a picture with depth information). The Microsoft Kinect is an example of such a camera, widely used in academic research because of its favourable pricing and free development software. In [9], a Kinect

camera was used to detect indoor mobile robots by using the Kinect as an objective (static) sensor. A state-of-the-art, industrial alternative to Kinect was launched by the Norwegian company Zivid Labs [10], and a small 3D scanning and mapping camera based on stereo vision called ZED, with 20m range, was released by Stereolabs [11].

1.3.2 Embedded Solutions for Processing of 3D Sensor Data

When designing a sensor package with the goal of fusing different sensors together, processing of 3D sensor data in real-time, and even some form of machine learning, might be necessary to perform locally in the package. For such circumstances, robust and powerful embedded hardware is essential.

There are several technologies that are suitable for processing large quantities of data, the first being Field Programmable Gate Arrays (FPGA). FPGAs are essentially microchips with a huge number of logical gates that can be programmed using hardware description languages (HDL). These can be tailor-made to a specific purpose and are extremely fast when used correctly. These massively parallel devices also require low power compared to CPUs (Central Processing Units), making them attractive for use in consumer electronics and embedded systems where low cost and low power consumption is a requirement [12].

Another technology is GPU (Graphics Processing Unit) based systems. GPUs have the advantage over CPUs by being able to process large amounts of data in parallel, and they are much more flexible than FPGAs. They have typically been used in personal computers or supercomputers where size and power consumption is not a major issue, but developers are now focusing on making low-power GPUs, as an alternative to FPGAs. In 2015, NVIDIA launched its upgraded Jetson TX1 platform, combining CPU and GPU processors for embedded purposes. The TX1 promised to deliver high performance, low energy computing for artificial intelligence, deep learning and machine vision [13]. Companies like Abaco and Aitech have specialised in making extremely robust versions of these systems with military-grade protection from the surrounding environment [14].

Artificial intelligence and deep learning for object detection using computers to find objects in images, are being researched and developed at a remarkable pace. This requires a huge amount of computational power, and with companies such as NVIDIA investing in machine learning and developing ever more powerful GPU based embedded computers, this field in computer science was on top of the Gartner 2016 hype cycle [15].

1.3.3 Sensor Fusion

In the paper [16], data from 2D laser scanners and 3D vision sensors were fused and combined for use in obstacle detection. In modern data fusion applications, probabilistic methods are considered to be the standard approaches. Probabilistic fusion expresses the data uncertainty using a probability distribution for each information source. In [16], each device provided an object list containing the position, size, orientation, velocity, and classification of each detected object along with the uncertainty of these measurements. For each object in the fused object list, a Kalman filter based prediction was applied to the object centre, object box size, heading, and object velocity. Afterwards, if there was an association to a current measurement, the respective Kalman states were corrected accordingly. The sensors and the data needed to be time synchronised as well. The computation units of 2D perception and 3D vision have different clocks. To merge the lists of tracked objects provided by each device, a global time reference is necessary and each data source must provide a precise timestamp. This was achieved by network time protocol (NTP) synchronisation where both the 2D perception system and the 3D vision system synchronise with the reference clock of the global control centre.

The 2D lasers and 3D cameras need to be registered to a common reference frame. In paper [16] this was performed by perceiving a reference object, at the same time, with both 3D vision and laser sensors and then estimating the optimal transformation parameters between the two coordinate systems by using a least squares technique. In [17], a similar approach was taken for detecting vehicles using lasers and stereo vision.

The paper [18] contains an extensive state-of-the-art analysis of multi-sensory data fusion, including 197 references, published in 2013. The challenging problems of multi-sensory fusion are categorised in [18] as follows:

1. Data imperfection,
2. Outliers and spurious data,
3. Conflicting data,
4. Data modality,
5. Data correlation,
6. Data alignment/registration,
7. Data association,
8. Processing framework,

9. Operational timing,

10. Static vs. dynamic phenomena,

11. Data dimensionality.

While many of these problems have been identified and heavily investigated, no single data fusion algorithm was capable of addressing all the aforementioned challenges. The variety of methods in the literature focus on a subset of these issues to solve, which would be determined based on the application in hand.

1.3.4 Sensor Calibration

Closely related to sensor fusion, when a sensor package with multiple different sensors, or even multiple sensor packages, are deployed, there is a need for a simple and fast calibration procedure. Traditionally, calibration has been a tedious process where manual measurement and labelling of a calibration target is involved. In [19], a software-aided procedure where a person only needs to walk in front of the sensors to calibrate the system was developed. This procedure was developed for 2D localisation, and a 3D version was not discussed in this paper. Although this procedure makes the calibration simpler, it must nonetheless be performed before the system can be deployed (offline), and it cannot detect miscalibration over time as the system is running.

In [20], an automatic calibration algorithm was developed to calibrate camera images to lidar (Velodyne) scans. By mapping laser points to the image and detecting if the depth of the laser points corresponds to edges in the image, a statistical method was developed to give the probability of whether the sensors are calibrated correctly or not. This technique can track and correct both sudden calibration errors and drifting errors over time, updating the translation and rotation parameters in real-time using a laptop CPU. It is stated in the paper that: *"Yet even using less than one second of data, our results are more accurate than state-of- the-art offline techniques which require a calibration target or hand-labelling of camera-laser correspondences"*.

As machine learning and artificial intelligence was gaining interest in the academic world, work had also been published where deep convolutional neural networks were used to fuse, calibrate, and detect objects in different sensor streams. The networks can learn to fuse different sensors, and even learn when to trust or not to trust a specific sensor. In the paper [21] members of the Autonomous Intelligent Systems and Computer Vision groups at the University of Freiburg, Germany, have developed a system of deep learning networks that can fuse multi-modal RGB and depth streams while performing semantic segmentation of a scene. It is stated that *"We introduced a new deep adaptive fusion architecture for end-to-end segmentation of*

multi-modal and multi-spectral images. Our [...] model achieves state-of-the-art performance compared to uni-modal segmentation and existing fusion approaches. More importantly, the model demonstrates considerable robustness to commonly observed environmental disturbances, critical for real-world robotic perception".

1.3.5 Optimisation Techniques

No published work addressing 3D sensor placement in a drilling rig was found in the literature. Most published work considers sensor placement in 2D environments (floor plans), or the 3D solutions presented are based on heuristic approaches which are non-repeatable and usually end up in local minima. Based on the literature review, the development of a fast and optimal sensor placement solution for the 3D drilling rig problem would be a novelty. Below is a summary of some of the papers which were reviewed.

In the paper [22], the problem of optimally placing a minimum number of distributed sensors to fulfil and optimise task requirements in a 2D environment was addressed. As stated in [22]: *"Overall, the research focus of indoor localisation lies on the development of signal extraction and localisation methods. Sensor placement is often done by hand, using the system developers best guess. Thus the resulting localisation error is neither calculated nor considered"*. In [22] the problem was formulated in a discrete and continuous search space. The discrete formulations were evaluated using Binary Integer Programming (BIP) and Mixed Integer Programming (MIP). The continuous formulation was evaluated using nonlinear programming (NLP) methods. All evaluations were done using the properties of a visual sensor system that exploited the thermal infrared radiation of humans for indoor localisation. All the presented methods were implemented using a 2D environment description and the resulting problems were solved respectively using MIP, BIP and NLP solvers. The results were compared against each other, taking their exactness and solving speed into account. An extension of the proposed methods to a 3D environment would be possible using the appropriate visibility calculations but was not part of the work.

In [23] the following was stated: *"Currently most designers of multi-camera systems place cameras by hand because there exists only little theoretical research on visual sensor placement. As video sensor arrays are getting larger, efficient camera placement strategies need to be developed"*. The approaches taken in [23] were subdivided into algorithms which give a global optimal solution but are complex and time/memory consuming, and heuristics which solve the problem in reasonable time and with reasonable complexity. The main contributions of [23] were:

- Space was sampled according to an underlying importance distribution instead

of using a regular grid of points.

- A linear programming model for each problem was presented which gave an optimal solution to the respective problem. It was shown how to reduce the number of variables and constraints significantly, thus enabling an optimal solution for larger problems.

- Several heuristics were proposed to approximate the optimal solution of the different camera placement problems.

- An interface that enables the user to comfortably enter and edit the space, the optimisation problems as well as the other setup parameters was presented.

- An experimental and competitive evaluation of the different approaches was given showing the different algorithms' specific advantages.

Future work will be modifying and applying the approaches to the 3D case and introducing more complex field-of-view and coverage constraints.

In [24] it is stated that *"although the discovery of an algorithm that can solve the most general case of the camera layout problem for a given volume of interest is highly desirable, it may prove quite challenging. We therefore focus on a more manageable subclass of this general problem that can be formulated in terms of planar regions that are typical of a building floor plan. We then approximate the region by a polygon. This is a valid assumption since most buildings and floor plans consist of polygonal shapes or can be approximated by a collection of polygons. The problem then becomes to reliably compute a camera layout given a floor plan to be observed, approximated by a polygon. A solution to this problem can be obtained via binary optimisation over a discrete problem space"*. The work is limited to 2D, and the authors write: *"In future work, we hope to pursue solutions to the optimisation in the continuous space as opposed to the discrete one"*.

In [25] it is stated: *"Although there are many studies about coverage for wireless multimedia sensor networks, most of them are based on two-dimensional terrain assumptions. However, particularly for outdoor applications, three-dimensional (3-D) terrain structure affects the performance of the WMSN remarkably"*, where WMSN refers to the wireless multimedia sensor network. The optimisation method used in the paper is a hybrid heuristic Genetic-Algorithm. The optimisation formulation was found to be NP-complete and for such problems there is no known efficient way to locate a solution. Hence, the authors implemented a GA-based heuristic approach, but such approaches usually end up in a local minimum.

1.4 Contributions

The research presented in this thesis has focused on making contributions to three major topics: 3D optimisation techniques, a 3D sensor package and framework for industrial/offshore use, and people detection in a three dimensional space, where the common denominator of all the topics was voxels, i.e. three dimensional pixels. The aim was to deliver technology which could be implemented and further developed by the industry. The contributions were published in the journal and conference papers listed below. All the papers are appended to this thesis in their full version.

The most important contributions are as follows:

- Novel algorithms using massively parallel GPU processing to solve the Optimal Sensor Placement problem in three dimensions.
- A prototype sensor package providing real-time 3D mapping of the Industrial Robotics Lab at the University of Agder, tested in both indoor and outdoor locations, enabling further research by this author and fellow colleagues at the SFI Offshore Mechatronics research centre.
- A ready-to-use ROS (Robot Operating System) software framework for a 3D sensor network for large scale industrial environments, including methods to compress, transmit and fuse live streams from multiple 3D sensor packages to a centralised server.
- A novel people detection scheme for voxel-based 3D data streams based on machine learning, as well as the publication of training datasets from an outdoor industrial environment.

1.4.1 Summary of Papers

Paper A – Optimal placement of 3D sensors considering range and field of view

This paper describes a novel approach to the problem of optimal placement of 3D sensors in a specified volume of interest. The coverage area of the sensors is modelled as a cone having limited field of view and range. The volume of interest is divided into many, smaller cubes each having a set of associated Boolean and continuous variables. The proposed method could be easily extended to handle the case where certain sub-volumes must be covered by several sensors (redundancy), for example ex-zones, regions where humans are not allowed to enter or regions where machine movement may obstruct the view of a single sensor. The optimisation problem is formulated

as a Mixed-Integer Linear Program (MILP) utilising logical constraints and piecewise linearisation of nonlinear functions. The final MILP problem is solved using the Cplex solver interfaced with Matlab.

Paper B – GPU-Based Optimisation of 3D Sensor Placement Considering Redundancy, Range and Field of View

This paper presents a novel and efficient solution for the 3D sensor placement problem based on GPU programming and massive parallelisation. Compared to prior art using gradient-search and mixed-integer based approaches, the method presented in this paper returns optimal or good results in a fraction of the time compared to previous approaches. The presented method allows for redundancy, i.e. requiring selected sub-volumes to be covered by at least n sensors. The presented results are for 3D sensors which have a visible volume represented by cones, but the method can easily be extended to work with sensors having other range and field of view shapes, such as 2D cameras and lidars.

Paper C – GPU-Based Occlusion Minimisation for Optimal Placement of Multiple 3D Cameras

This paper presents a fast GPU-based solution to the 3D occlusion detection problem and the 3D camera placement optimisation problem. Occlusion detection is incorporated into the optimisation problem to return near-optimal positions for 3D cameras in environments containing occluding objects, which maximises the volume that is visible to the cameras. In addition, the authors' previous work on 3D sensor placement optimisation is extended to include a model for a pyramid-shaped viewing frustum and to take the camera's pose into account when computing the optimal position.

Paper D – Embedded Processing and Compression of 3D Sensor Data for Large Scale Industrial Environments

This paper presents a scalable embedded solution for processing and transferring 3D point cloud data. Sensors based on the time-of-flight principle generate data which are processed on a local embedded computer and compressed using an octree-based scheme. The compressed data is transferred to a central node where the individual point clouds from several nodes are decompressed and filtered based on a novel method for generating intensity values for sensors which do not natively produce such a value. The paper presents experimental results from a relatively large industrial robot cell with an approximate size of $10\,\mathrm{m} \times 10\,\mathrm{m} \times 4\,\mathrm{m}$. The main advantage of processing point cloud data locally on the nodes is scalability. The proposed solution could, with a dedicated

Gigabit Ethernet local network, be scaled up to approximately 440 sensor nodes, only limited by the processing power of the central node that is receiving the compressed data from the local nodes. A compression ratio of 40.5 was obtained when compressing a point cloud stream from a single Microsoft Kinect V2 sensor using an octree resolution of 4 cm.

Paper E – CNN-based People Detection in Voxel Space using Intensity Measurements and Point Cluster Flattening

In this paper real-time people detection is demonstrated in a relatively large indoor industrial robot cell as well as in an outdoor environment. Six depth sensors mounted at the ceiling are used to generate a merged point cloud of the cell. The merged point cloud is segmented into clusters and flattened into grey-scale 2D images in the xy and xz planes. These images are then used as input to a classifier based on convolutional neural networks (CNNs). The final output is the 3D position (x, y, z) and bounding box representing the human. The system is able to detect and track multiple humans in real-time, both indoors and outdoors. The positional accuracy of the proposed method has been verified against several ground truth positions, and was found to be within the point-cloud voxel-size used, i.e. 0.04 m. Tests on outdoor datasets yielded a detection recall of 76.9 % and an F1 score of 0.87.

1.5 Published Software and Dataset

A large portion of the work done in this project was dedicated to writing software. The source code for the people detection method, and the point cloud compression, decompression, and fusion methods are published under a BSD 3-Clause License, enabling the industry, other researchers, or interested parties to freely use and further develop the software. Mostly written in C++ and MATLAB, the source code can be found here: https://github.com/dybedal.

In addition, a large dataset of 3D sensor data recorded in an outdoor industrial test facility is published here: https://doi.org/10.18710/HMJVFM. This dataset was also used when training and testing the people detection method in Paper E.

Chapter 2

3D Optimisation

In this thesis, the 3D sensor placement problem is considered given various constraints, such as obstacles and limited range of the sensors. This chapter contains relevant background information on the two approaches taken in Papers A, B and C in this thesis: 1) Mixed-integer programming and 2) Massive parallelisation (or brute force search) using a GPU.

2.1 Mixed Integer Programming

A standard linear program can be formulated as follows:

$$\min \mathbf{g}^T \mathbf{x} \tag{2.1}$$

subject to:

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{2.2}$$

where:

$\mathbf{x} \in \mathcal{R}^n$	is a vector of n real decision variables
$\mathbf{g}$	is a vector of objective function coefficients
$\mathbf{A} \in \mathcal{R}^{n \times n}$	is a square matrix of constraint coefficients
$\mathbf{b} \in \mathcal{R}^n$	is a vector of constraint coefficients

A problem is called mixed-integer when some of the variables in $\mathbf{x}$ are forced to be integer values only, i.e.

$$\mathbf{x}_i \text{ is integer where } i \in \mathcal{I}. \tag{2.3}$$

Consider the following example:

$$\min (x_1 - 2x_2 + 3x_3) \tag{2.4}$$

subject to:

$$\begin{bmatrix} -1 & -1 & 0 \\ 0 & -1 & -1 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ -3 \\ 3.5 \end{bmatrix}. \tag{2.5}$$

The optimal solution for this problem is $\mathbf{x} = [5.9090909, 3.5, -3.5]$ which gives an objective function value of -11.59090909. If x_2 is now constrained to be an integer, i.e. $\mathcal{I} = \{2\}$, then the optimal solution becomes: $\mathbf{x} = [5.455, 3, -3.0]$ with an objective function value of -9.54545454. This example demonstrates that the objective function value of an integer constrained problem becomes equal or less than the unconstrained problem. Special solvers are required for solving mixed-integer problems, typically using some type of branch-and-bound algorithm. Going into the details of mixed-integer solvers, however, is outside the scope of this thesis.

One aspect of mixed-integer programming which makes it particularly attractive for practical applications is the possibility to formulate logical constraints. Some examples of logical constraints are presented here, but the interested reader is referred to [26]. Below, some logical implications are presented and it is shown how these can be converted to linear constraints suitable for use in a standard mixed-integer program.

Boolean variables are simply integer variables constrained to have a value of either zero or one. Implication 1 sets the Boolean variable δ to one (true) if the continuous variable x lies between the two functions $f_1(x)$ and $f_2(x)$. If x does not lie between these two functions, there is no constraint on δ and its value can be either zero or one.

Implication 1

Constraints:

$$(f_1(x) \leq x \leq f_2(x)) \implies \delta \tag{2.6}$$

is equivalent to:

$$(m - \epsilon)\delta_1 \leq (f_1(x) - x) - \epsilon \tag{2.7}$$

$$(m - \epsilon)\delta_2 \leq (x - f_2(x)) - \epsilon \tag{2.8}$$

$$\delta_1 + \delta_2 - \delta \leq 1, \tag{2.9}$$

where δ, δ_1 and δ_2 are Boolean variables, while x, $f_1(x)$ and $f_2(x)$ are continuous. m is a constant smaller than the lower limit of both $f_1(x)$ and $f_2(x)$. ϵ is a small positive tolerance value, which can be selected as low as possibly accepted by the particular solver in use.

Implication 2 sets the Boolean variable δ to zero if the continuous function $f(x) \geq k$. Otherwise, there is no constraint on δ, it can then be either zero or one.

Implication 2

Constraints:

$$(f(x) \geq k) \implies \delta = 0 \tag{2.10}$$

is equivalent to:

$$-(\epsilon + M)(1 - \delta) \leq -f(x) + k - \epsilon, \tag{2.11}$$

where δ is a Boolean variable, while x and $f(x)$ are continuous. M is a constant larger than the upper limit of $f(x)$. ϵ is a small positive tolerance value, which can be selected as low as possibly accepted by the particular solver in use.

As an example, let $f(x) = 2x + 1$, $k = 4$, $M = 11$ for $x < 5$, and $\epsilon = 0.001$. If $x = 1$, we then have

$$-(0.001 + 11)(1 - \delta) \leq -(2 + 1) + 4 - 0.001 \tag{2.12}$$

$$-(11.001)(1 - \delta) \leq 0.999, \tag{2.13}$$

which shows δ can be either zero or one. If $x = 1.5$, which is the intersection point of $f(x)$ and k, we then have

$$-(0.001 + 11)(1 - \delta) \leq -(2 * 1.5 + 1) + 4 - 0.001 \tag{2.14}$$

$$-(11.001)(1 - \delta) \leq -0.001, \tag{2.15}$$

which shows that δ has to be zero, and it also shows the purpose of introducing the tolerance value ϵ.

Once a Boolean variable δ is defined as described in Implications 1 and 2 above, it can for example be multiplied together with a continuous function $f(x)$ as shown below:

$$z = \delta \cdot f(x) \tag{2.16}$$

is equivalent to:

$$-M\delta + z \leq 0 \tag{2.17}$$

$$m\delta - z \leq 0 \tag{2.18}$$

$$-m\delta + z \leq f(x) - m \tag{2.19}$$

$$M\delta - z \leq -f(x) + M, \tag{2.20}$$

where m, M are lower and upper bounds on the function $f(x)$ and z is the product of the Boolean variable and the continuous function, i.e. if $\delta == 1$ then $z = f(x)$, otherwise $z = 0$. As demonstrated in Paper A in this thesis, such logical constraints can for example be used to formulate and solve 3D sensor placement problems.

2.2 Massive Parallelisation on GPUs

As an alternative to traditional optimisation techniques such as mixed-integer programming or gradient-search based methods, parallelisation using Graphical Processing Units (GPUs) has become a viable alternative, given the rapid growth in the capabilities of GPUs in recent years. Traditional approaches have some drawbacks, for example getting stuck in local optima, and it can take a long time to find a good solution, which is not necessarily the optimal one for non-convex problems. Some benefits of GPU-based optimisation are the fact that the entire search-space can be covered relatively quickly, and the best solution from a GPU-based search can later be used as an initial condition for a gradient-based search in case the discretisation used in the GPU-based search is too coarse.

To illustrate the benefits of GPU-based optimisation, consider the following objective function:

$$F(x, y) = x + x \cdot \cos(5\pi \cdot x/N) \cdot y \cdot \sin(5\pi \cdot y/N) \tag{2.21}$$

With $N = 200$ this objective function is illustrated in Figure 2.1. As seen from the figure, with this objective function $F(x, y)$ there are two optima with the same global maxima, i.e. $x = -100$ and $y = \pm 90$ giving a value of $F(x, y) = 8900$. Since there are many local optima in Figure 2.1, a gradient search-based method would have serious problems in finding the global optimal maxima in this example.

The following code example using Python and Numba library shows how a GPU-

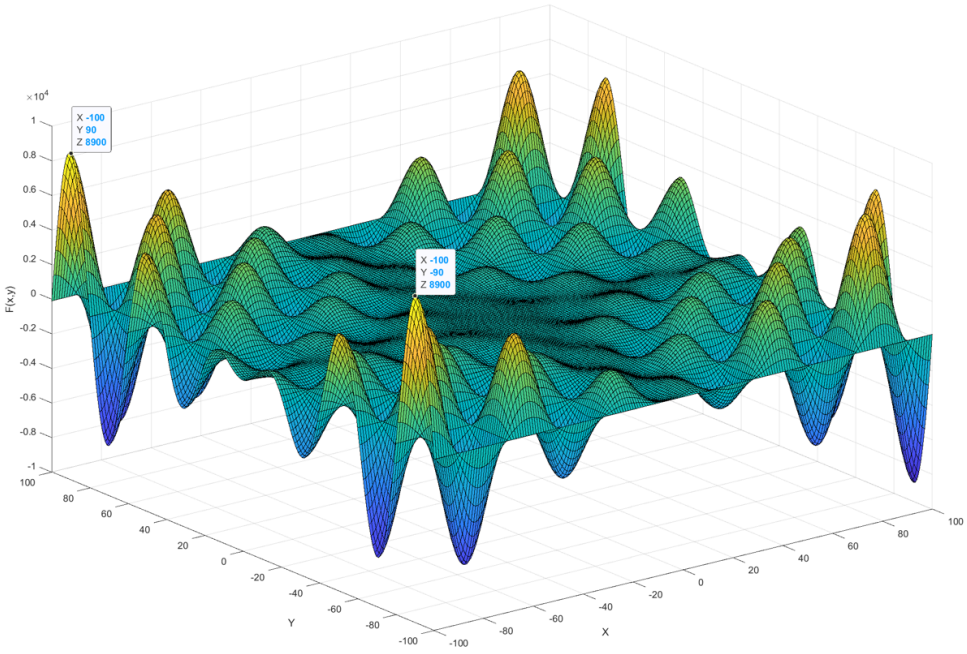


Figure 2.1: Illustration of non-convex objective function with many local maxima.

based search can be implemented. Using the Numba JIT (Just In Time) compiler, the function declared with the `@jit` decorator is compiled and can be set to execute either on the CPU or a GPU.

```
from numba import jit
import numpy as np
from math import cos, sin, pi
from timeit import default_timer as timer

@jit("void(float64[:,:],int32)")
def optim_func(a,n):
    N = n/2
    for i in range(n):
        for j in range(n):
            x = i - N
            y = j - N
            a[i,j] = x+x*cos(x/N*5*pi)*y*sin(y/N*5*pi)

if __name__=="__main__":
    n = 200
    Z = np.zeros(shape=(n,n), dtype = np.float64)
```

```
start = timer()
optim_func(Z,n)
print("Time taken:", timer()-start)

Jmax = Z.max(0).max(0)
print("Maximum value found: {}".format(Jmax))
index = np.where(Z == Jmax)
for result in index:
    x = result[0] - n/2
    y = result[1] - n/2
    print("Maximum value found at: x={} y={}".format(
       x,y))
```

Running the same code on both the CPU and the GPU for comparison, the output is as follows:

```
Maximum value found: 8900.0
Maximum value found at: x=-100.0 y=-100.0
Maximum value found at: x=-90.0 y=90.0
CPU: Time taken: 0.014300544979050756
GPU: Time taken: 0.0006120820180512965
```

In this particular example with the particular hardware chosen, the GPU implementation is about 23 times faster than the CPU implementation, even for this small problem. The speed benefit will increase with the size of the problem (N).

In a typical optimisation problem the search-space is too large to cover, even when using a GPU. In such cases an approach based on randomisation of the search variables can be used instead. A typical optimisation problem can be formulated as follows:

$$\mathbf{k}^* = \text{argmax}_k F(\mathbf{x}, \mathbf{k}) \tag{2.22}$$

$$G(\mathbf{x}, \mathbf{k}) \leq \mathbf{b}, \tag{2.23}$$

where $\mathbf{k}$ is a vector of optimisation or design variables, $\mathbf{k}^*$ is the optimal solution and $F(\mathbf{x}, \mathbf{k})$ is the objective function. One example is a crane or a robot where the design is optimised with respect to some parameters $\mathbf{k}$ in the workspace $\mathbf{x}$. The workspace $\mathbf{x}$ can be 2D as in the previous example (x, y), 3D, or higher dimension. The parameters $\mathbf{k}$ could for example be arm lengths, stiffness parameters, gear ratios, etc. The objective function $F()$ can be nonlinear and/or mixed discrete/continuous. If the search space in $\mathbf{k}$ is small, more search loops can be added, just like for x and

y in the previous example.

If the search in $\mathbf{k}$ is large, randomisation in the generation of $\mathbf{k}$ can be used, which was the approach taken in, for example, Paper B in this thesis. The equation $G(\mathbf{x}, \mathbf{k}) \leq \mathbf{b}$ describes a set of linear or nonlinear constraints. When generating the samples of the vectors $\mathbf{x}, \mathbf{k}$, either by search loops or by randomisation, $\mathbf{k}^*$ is only updated if the constraints are satisfied.

With randomisation, the GPU is used until a solution that is considered "good enough" is found. There is no guarantee that the optimal solution has been found. The optimisation problem described above is very hard, even more so for traditional approaches.

The following Python code shows a code example where the parameters k1 and k2 are generated by randomisation:

```
from numba import jit
import numpy as np
import random
from math import cos, sin, pi
from timeit import default_timer as timer

@jit("float64[:](float64[:,:],int32)")
def optim_func(a,n):
    N_rand = 100000
    N = n/2
    Jmax = 0.0
    a_max = np.zeros(1, dtype = np.float64)
    k_best = np.zeros(2, dtype = np.float64)
    for l in range(N_rand):  # Extra randomization loop
        k1 = random.randrange(1,10)
        k2 = random.randrange(-10,10)
        a_max = 0.0
        for i in range(n):
            for j in range(n):
                x = i - N
                y = j - N
                a[i,j] = x + 1/k1*cos(x/N*5*pi) * k2*sin(
                   y/N*5*pi)
                if a[i,j] > a_max:
                    a_max = a[i,j]
        if a_max > Jmax:
            Jmax = a_max
```

```
                k_best[0] = k1
                k_best[1] = k2
    return k_best

if __name__=="__main__":
    n = 200
    Z = np.zeros(shape=(n,n), dtype = np.float64)

    start = timer()
    k_best = optim_func(Z,n)
    print("Time taken:", timer()-start)

    Jmax = Z.max(0).max(0)
    print("Maximum value found: {} with k1 = {} and k2 =
       {}".format(Jmax,k_best[0],k_best[1]))
    index = np.where(Z == Jmax)
    for result in index:
        x = result[0] - n/2
        y = result[1] - n/2
        print("Maximum value found at: x={} y={}".format(
           x,y))
```

Running the same code on both the CPU and the GPU for comparison, the output is as follows:

```
Maximum value found: 99.74 with k1 = 1.0 and k2 = -10.0
Maximum value found at: x=99.0 y=99.0
Maximum value found at: x=-70.0 y=-30.0
Time taken with GPU: 60.96680168795865
Time taken with CPU: 2297.6071298610186
```

In this particular example with the particular hardware chosen, the GPU implementation is about 37 times faster than the CPU implementation.

To conclude, brute-force optimisation using a GPU can be a good solution, when: 1) The objective function has many local optima, 2) The objective function is strongly nonlinear or contains mixed float & integer variables and 3) A good initial starting point is needed for the traditional approaches to optimisation.

Chapter 3

3D Sensors and Data Structures

This chapter presents the different 3D sensor types and 3D data formats that were studied and utilised during the project. The aim is to give the reader a broader background than is given in the appended papers, and an insight in the most used 3D sensor hardware and software technologies.

3.1 3D Sensor Types

The following section outlines the different 3D sensor types and technologies that were evaluated or used in this project, ranging from image-based to laser and radar based.

3.1.1 Stereo Vision Sensors

A stereo vision sensor, also referred to as a stereo camera or binocular vision, is made up of two cameras, placed a given distance from each other and looking in the same direction, much like the human eyes. The two cameras create images from slightly different points of views resulting in a position shift or parallax of objects in the two images. By knowing the relative position and rotation between the cameras, depth information can be extracted by calculating the disparity between the two images, as seen in Figure 3.1.

Solving the stereo matching problem to gather the disparity information (i.e. finding the same object in the two images) has been the most prominent challenge with stereo vision. In recent years, machine learning has been used to enhance stereo matching, and the research on learning-based depth extraction in stereo images has also enabled new possibilities in estimating depth information from monocular (single) images [27].

The typical output of a stereo camera system is an RGB-D Image (depth image containing colour information, see Section 3.2). At low cost compared to other 3D

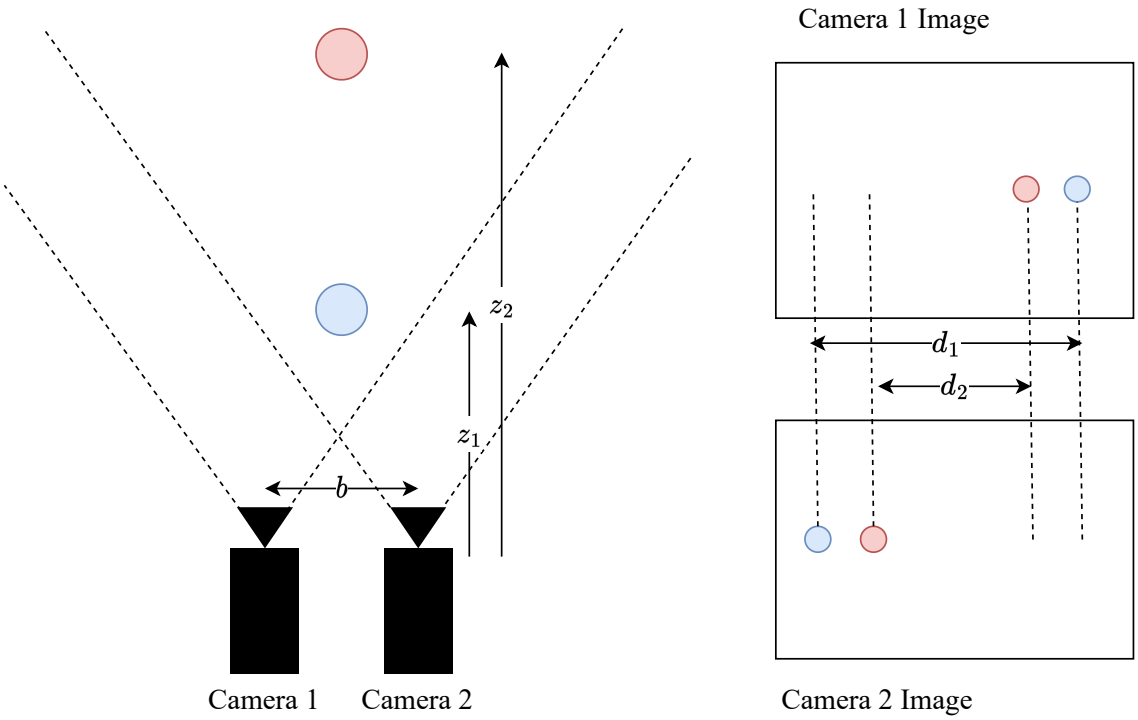


Figure 3.1: Simple stereo vision setup for identical, parallel cameras. Distance z to an object can be calculated using triangulation; $z = \frac{fb}{d}$, where f is the focal length of the camera, b is the baseline distance between the cameras, and d is the disparity. The left hand side shows the setup from above, the right side shows the resulting images from each sensor

sensors, stereo cameras have been common to see in drones and other unmanned vehicles, often used as input to SLAM (Simultaneous Localisation And Mapping) algorithms to create a three dimensional map of an environment [28, 29]. But the low cost also comes at a price: When the distance from an object to the sensor increases, the change in disparity decreases, resulting in higher and higher depth resolution error. An example of a SLAM output using the first version of the ZED Stereo Camera by Stereolabs [11] can be seen in Figure 3.2. Here it is visible how the algorithm tries to connect the depth information in each stereo image to form a connected mesh of the environment.

3.1.2 Structured Light Cameras

Structured Light sensors were becoming increasingly popular at the early stage of this thesis, with numerous brands and types emerging, including Zivid, founded in 2015 [10], and Intel RealSense cameras [30]. These brands and more have continued to improve the technology with updated and enhanced sensors, with Zivid now claiming time-coded structured light is the "potentially most accurate 3D technology" [31]. One of the most popular and well-known sensor using structured light was the first

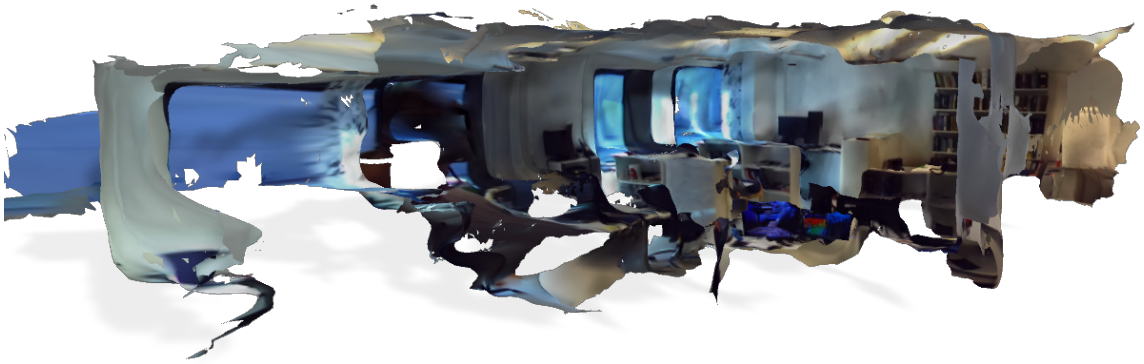


Figure 3.2: Output of a SLAM algorithm using a stereo camera

version of the Microsoft Kinect [32]. This sensor was widely used in academia due to its low cost and freely available SDK (Software Development Kit), with the Kinect for Windows version released in 2012. However, in 2014, Microsoft released an updated version, referred to as Kinect V2 in this thesis, which changed the technology from structured light to time-of-flight (see Section 3.1.3).

Structured light sensors consist of a camera and a projector, where a known pattern is projected onto objects and reflected back at the camera. Projected using visible or invisible (IR) light, the pattern is warped by objects in the sensor's field of view. Since the original pattern is known, the displaced pattern is used to infer three dimensional shapes, thus creating three dimensional measurements. Different pattern shapes or time-varying/time-coded patterns can be used to increase the sensor's performance in a given application.

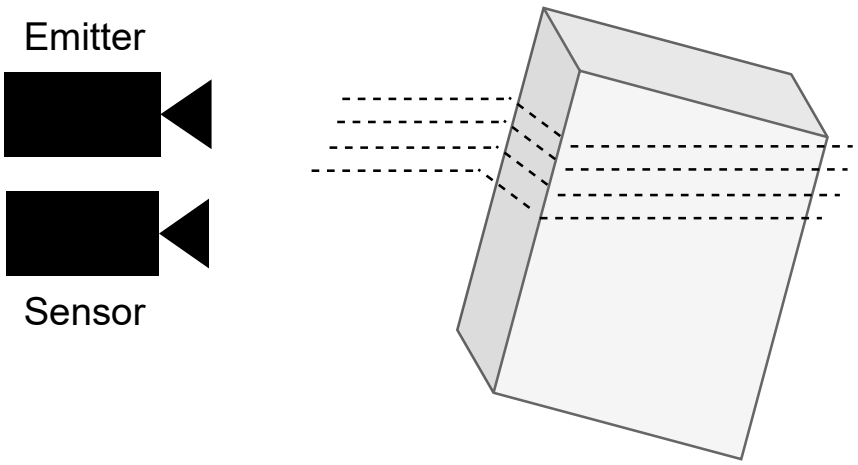


Figure 3.3: A simple illustration of a structured light sensor, comprised of an emitter/projector that is sending out a light pattern, an array of lines in this case. The sensor records the transformed lines and software is used to deduce the shape of the measured object by comparing the original pattern to the resulting pattern.

Figure 3.3 shows a simple illustration of a structured light sensor. For an in-depth

explanation of the workings of structured light sensors, the interested reader is referred to [33].

3.1.3 Time-of-Flight Cameras

Time-of-flight (ToF) cameras, such as the Microsoft Kinect V2 [32] as used in this project, use a light source to illuminate a scene, and measures distance based on the reflected signal. The light source is typically an IR LED, as in the Kinect V2, and each pixel in the camera's sensor measures the distance, resulting in a depth image. There are several techniques for measuring the distance, where the simplest is to emit a light pulse and measure the amount of light reflected in a specific timeframe. The less reflected light that is received during this window, the larger the distance. A more common method is to use modulated IR pulses and measure the phase shift in the reflected signal. Figure 3.4 shows an illustration of a ToF camera, with an emitter and a sensor, and a more in-depth description of the workings of ToF cameras can be found in [34].

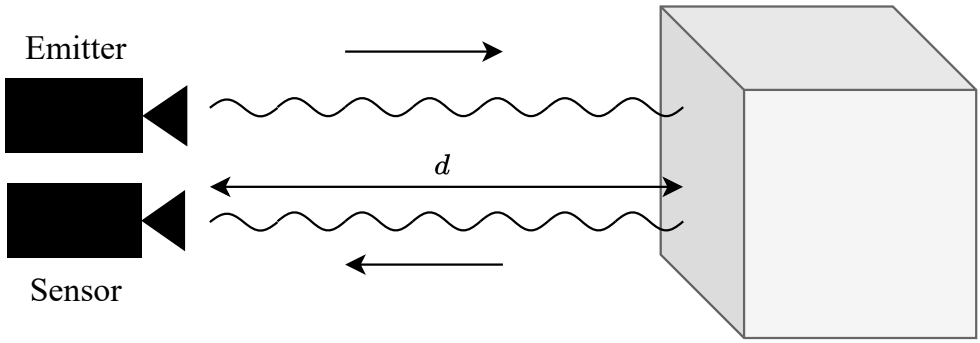


Figure 3.4: A simple illustration of a time of flight sensor. An IR LED is typically used as the emitter. The distance d is measured by measuring the amount of returned light in a certain window using a shutter, and/or measuring the phase shift of the returned signal.

A drawback of this type of sensor, which was also noted during the work on this thesis, is that not all surfaces are reflective enough to send a strong signal back to the sensor. A typical scenario is people wearing very dark clothing. Another challenge is the sensor's sensitivity to ambient light, especially sunlight, which can lead to saturation. However, it was found in this thesis that the Microsoft Kinect V2, specifically, can produce good measurements even in outdoor areas. An in-depth analysis of IR light reflection was performed in [35].

3.1.4 Lidar

LiDAR, or lidar, as the name suggests, is a light based form of radar (Radio Detection and Ranging), where radio waves are replaced by light waves, and more specifically,

laser beams. Laser based range measurement devices exists in a wide variety of forms, from simple single beam measuring devices, via spinning lidars with up to hundreds of beams, to solid state, camera-like lidars.

There are two main types of lidars used in 3D computer vision, the first being spinning lidars, and the second being solid state lidars. Spinning lidars consist of an array of laser emitters and sensors, arranged vertically with fixed angles between them. The array is then mounted on a rotating platform, resulting in each of the individual lasers measuring a circle around the rotational axis. A very simplified conceptual drawing of a spinning lidar can be seen in Figure 3.5.

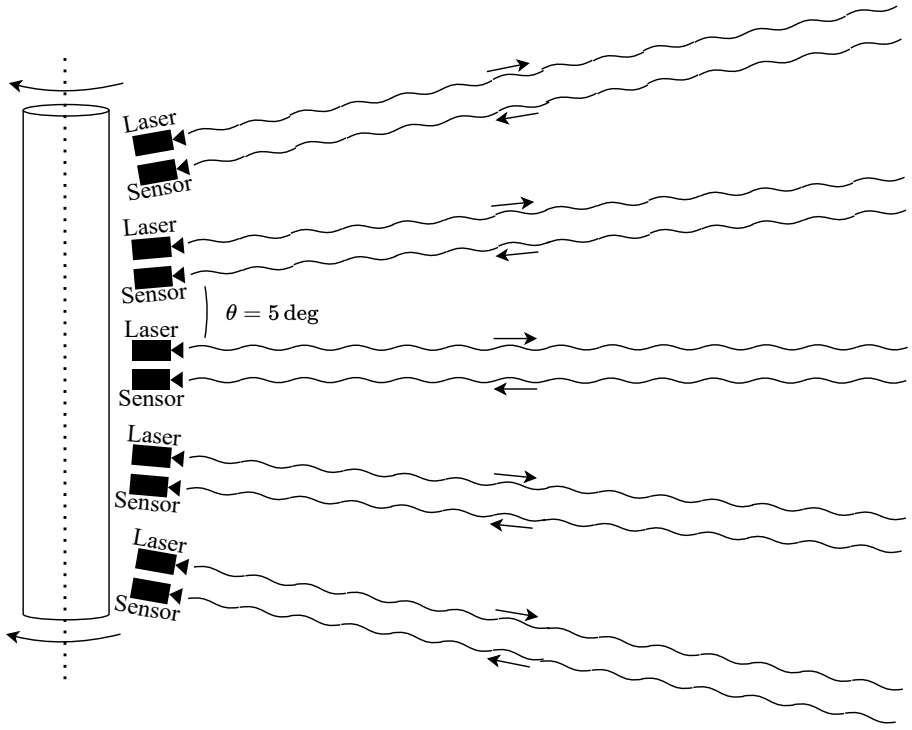


Figure 3.5: A conceptual illustration of a spinning lidar, with 5 sensors mounded with a 5 degree angle between them. An IR laser is typically used in each emitter.

A Solid State Lidar on the other hand, is a lidar with no moving parts. One category of solid state lidar, also known as the Flash Lidar, is a subset of the time-of-flight cameras discussed in the previous section. Based on the same principles, the main difference is that the entire scene is typically illuminated using a single laser pulse, and distance is estimated by measuring the direct time-of-flight of the pulse to each camera pixel. Other types of solid state lidars use micro-electromechanical system (MEMS) mirrors [36] or optical phase arrays (OPA) [37] to steer a laser beam in different directions.

3.1.5 Radar

RADAR, or radar (Radio Detection and Ranging) sensors use electromagnetic waves in the microwave range. Widely adopted in the automotive industry for anti-collision and cruise control systems, radar sensors use the same time-of-flight principle as the lidar. As radar uses lower frequency waves than lidar and other IR/visible light based sensors, typically around 77 GHz in modern automotive mm-wave radar, it has both advantages and disadvantages over other sensors. The main advantage of radar are its range and penetration abilities, making detection possible at longer ranges and in rougher weather conditions such as heavy rain and fog [38]. The main disadvantage in the context of this thesis is the 3D resolution. The distance resolution is limited by the bandwidth of the radar, and the angular resolution is dependent on the number of sending and receiving antenna arrays. A typical state-of-the art automotive radar yields a distance resolution of around 0.2 m, and an angular resolution of 2-3 degrees [39, 40].

Due to these limitations compared to camera and lidar, radar sensors were not investigated further during this project. However, in the very recent years, research on people detection based on mm-wave radar has gained some momentum. One advantage of radar is that it provides information about an object's velocity in addition to distance, due to the Doppler component of the returned signal. Another reason to use radar instead of camera vision is privacy, as there are concerns about images and videos being leaked or stolen [41]. Most published research on this topic performs tracking in indoor environments, with focus on identification and tracking [41, 42, 43] and Human Activity Recognition (HAR) [44].

3.2 3D Data Representation

This section gives a brief introduction to different formats for storing the three dimensional measurements taken by 3D sensors. While all formats store depth information, different formats are used based on the data source (sensor), but also based the application.

3.2.1 Depth Maps and RGB-D Image

A depth map, or depth image, is a two dimensional matrix with depth measurements stored in each cell. For many types of 3D sensors, this is the raw format in which the sensor captures the measurements. In e.g. the Microsoft Kinect V2 which was used in multiple of the papers in this thesis, the depth sensor is an infrared time-of-flight sensor with a resolution of 512 x 424 pixels.

The depth map in its raw format cannot be used to generate three dimensional points from the measurements, as it contains no information about the transformation from the image sensor to the measured points. In other words, the only information in the depth image is the distance to a measured object, not the direction.

An RGB-D image is a depth image with additional colour information. In addition to the depth channel, three more channels are added for red, green and blue, hence the abbreviation RGB-D. To accomplish this, a colour image is typically projected onto the depth image, or vice versa, using a calibrated transformation. RGB-D images can be generated by multiple types of sensors, either by using stereo images or a combination of an image sensor and a depth measurement sensor. In sensors that contain both a colour sensor and a depth sensor, matching the two images is usually referred to as an intrinsic calibration.

3.2.2 Point Clouds

Point clouds are the most common used representation form used when displaying 3D sensor measurements, as they contain the exact three dimensional coordinates of the measured points and can be easily interpreted by humans. Where depth and RGB-D images are stored in a two dimensional array, point clouds are typically stored in a one-dimensional list.

As discussed above, generating a point cloud from a depth map or RGB-D image requires knowledge about the sensor, its field of view, and optics (lens distortion, etc.) to create a transform from the two-dimensional image to the three dimensional space.

While storing data in point clouds has the advantage that the points are represented by x,y,z coordinates, one of the disadvantages is storage size. Depth images commonly use a single 32 bit number per point to store depth information, and RGB-D images include 24-bit RGB colour depth. Point clouds on the other hand typically require three 32 bit floating-point numbers per point just to store the coordinate, resulting in at least 3 times more required storage space, which is why some sort of compression is essential when transmitting real-time point cloud streams. Additional information such as colour and intensity could also need to be stored, as detailed in Paper D. An example of a point cloud generated by the depth maps of three Kinect V2 sensors can be seen in Figure 3.6.

3.2.3 Voxels and Octree

In this thesis, some of the most used 3D data representations are voxels and octrees, where both these types add structure to the data. Voxels, a term derived from the two dimensional pixel, can be considered pixels in 3D space. In a voxel space, the

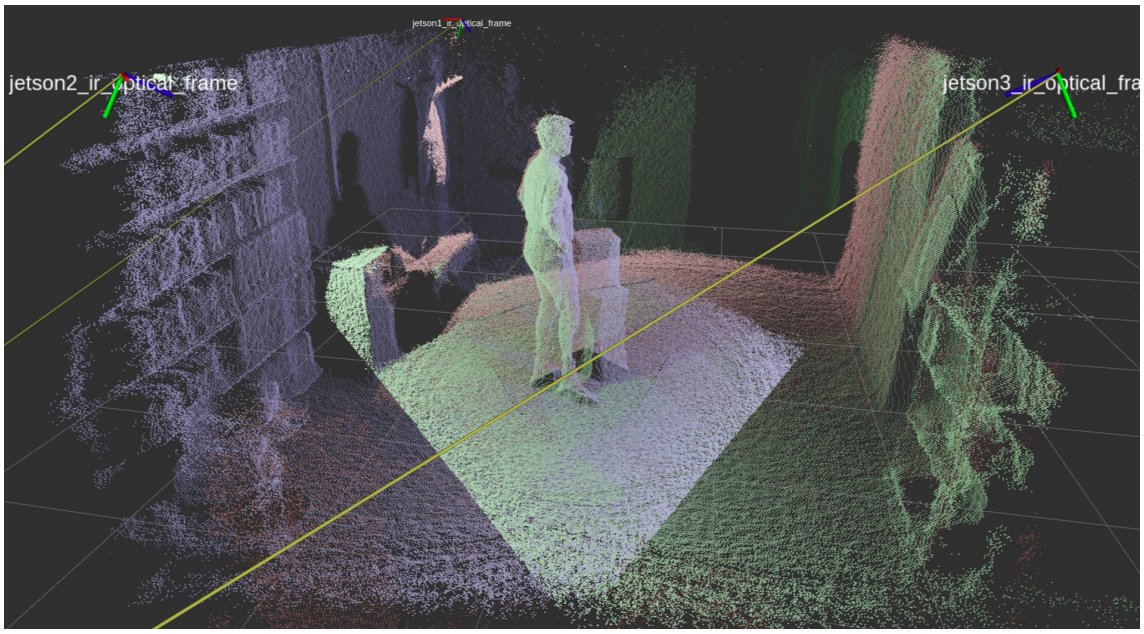


Figure 3.6: An example of a point cloud generated by Kinect V2 sensors.

volume of interest is divided into a three dimensional grid, resulting in a number of equally sized cubes, or voxels. Adjusting the size of the cubes will therefore adjust the resolution of the voxel space. This division of a space to a fixed set of sub-spaces was used in Papers A, B and C, where each voxel, or cube, was assigned it's own thread in the CUDA optimisation algorithm. In [45], a voxel space was also used to perform massively parallel GPU computations for real-time collision detection. A figure of a volume divided into voxels can be seen in Figure 3.7.

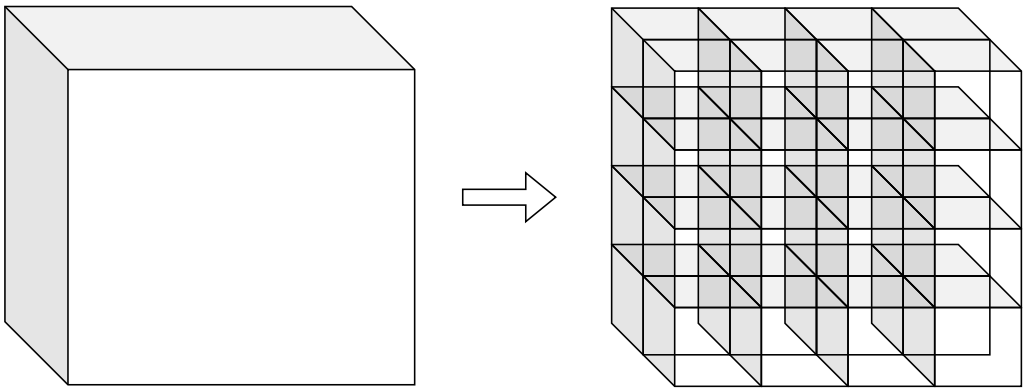


Figure 3.7: A representation of 3D pixels, or voxels. Here the original volume is divided into a $4 \times 2 \times 4$ grid of voxels.

When creating a voxel space where the data source is is e.g. a point cloud, each voxel can contain data from none to many original points. This results in multiple points being merged to one if a voxel encompasses more than one point. One approach, used in Paper D, is to add the number of points inside a voxel together, creating an artificial intensity metric, where voxels with more points have a higher

intensity.

While the use of voxels helps generate structure, storing the data in such a format can be inefficient, especially in sparse volumes where there is a lot of empty space. In 1980, the octree was proposed as a new way of modelling 3D objects [46]. As the name suggests, the octree is a tree structure with parent and child nodes.

When creating an octree, the entire volume of interest is covered by a single voxel. Then, the following algorithm is used:

- If the voxel contains more than one point, divide the voxel into eight smaller voxels (octants).
- For each of the new voxels, repeat this process recursively.
- Stop the process when all voxels contain only a single point, or when the desired tree depth is reached.

Adjusting the depth of the tree will thus adjust the resolution of the octree. The octree model was used when compressing, transferring and decompressing point clouds in Paper D in this thesis, allowing multiple real-time point cloud streams to be received and merged by a server on a standard Ethernet connection. An example of an octree is shown in Figure 3.8.

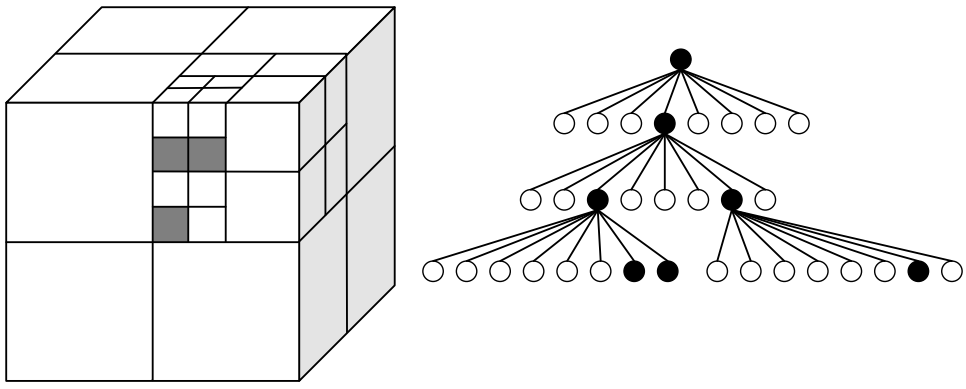


Figure 3.8: A representation of an octree. The occupied voxels are coloured grey, and the resulting tree is shown to the right. Branches with empty leaf nodes can be pruned, as they store no information.

3.3 Point Cloud Compression and Filtering

In 2011, the Point Cloud Library (PCL) was presented [47]. This software library includes multiple tools for working with point clouds, including an octree-based compression based on the work presented in [48]. This method was the basis for the

point cloud sensor network technique used in Paper D, where multiple real-time point clouds generated from a sensor network were compressed, transferred and merged to one single point cloud. This merged, real-time point cloud was in turn used as input for the human detection algorithm presented in Paper E.

Point cloud compression of real-time (dynamic) point clouds differ from the compression of static point clouds. Where static point clouds can be compressed by more conventional compression methods such as DEFLATE commonly used in zip files [49], streams of point clouds require a fast and efficient algorithm. While the custom octree approach used in this thesis includes temporal compression, a standard compression method such as the well known JPEG (by the Joint Photographic Experts Group) for images and MPEG (by the Moving Picture Experts Group) for videos was not available at the time the research in Paper D was performed. However, both these groups are now working on standards including point cloud compression, which could enable more standardised way of storing, viewing, transferring and manipulating point cloud streams in the future [50, 51, 52].

In Paper D, the merged point cloud was de-noised using a custom algorithm based on the intensity of the voxels. Knowing the distance from the sensor to the measured object, measurements closer to the sensor were given lower weight than measurements further away. In a multi-sensor environment, this resulted in a point cloud where objects close to and far from the sensors would get similar intensity. Setting a lower limit for allowed intensity value efficiently removed noisy measurements from the point cloud, leaving mostly valid measurement points. This efficient method allowed the point cloud to be segmented based on minimum Euclidean distance between points as described in Paper E, without the presence of many outliers.

An extensive review of other point cloud filtering methods was performed in [53], and the interested reader is therefore referred here for more information on this topic.

Chapter 4

Experimental Setup and Prototyping

This chapter presents the experimental setup and 3D sensor package prototypes that were created and used throughout this project. Most prototyping and experiments described in this thesis were performed in the Industrial Robotics Lab (IRL) at the University of Agder, Grimstad, Norway. The lab was built to resemble an industrial environment, including industrial robots and processing equipment. An overview of the lab can be seen in Figure 4.1. This chapter gives an introduction to the sensors and computing hardware that were used in the lab, and the rationale behind it. Paper D describes the selected hardware and setup in more detail.

Figure 4.1: Overview of the Industrial Robotics Lab at the University of Agder.

4.1 Selected 3D sensors

Multiple 3D sensor types were tested for applicability, including range, field of view, and resolution. Based on the initial tests, the state-of-the-art analysis in Section 1.3, the technology analysis in Section 3.1 and the availability and cost of the sensors, three different sensor technologies were chosen for this project: A time-of-flight camera (Microsoft Kinect V2), a Lidar (Velodyne VLP-16 PUCK) and an industrial stereo imaging camera (Carnegie Robotics Multisense S21). The different technologies were selected both to be able to compare the measurements, but also to take advantage of sensor fusion where one sensor might perform better than another in different environmental scenarios.

The Kinect Time-Of-Flight camera was selected due to its cost, availability and extreme popularity in academia. This allowed for a relatively fast prototyping phase due to available open source drivers and ROS software. It would also make results more easily comparable to other research using the same type of sensor.

The Velodyne Lidar was chosen due to it's high accuracy, and increasing popularity among the computer vision community, especially withing autonomous machines and autonomous driving. Experience with Lidar systems was deemed to be highly valuable in future research.

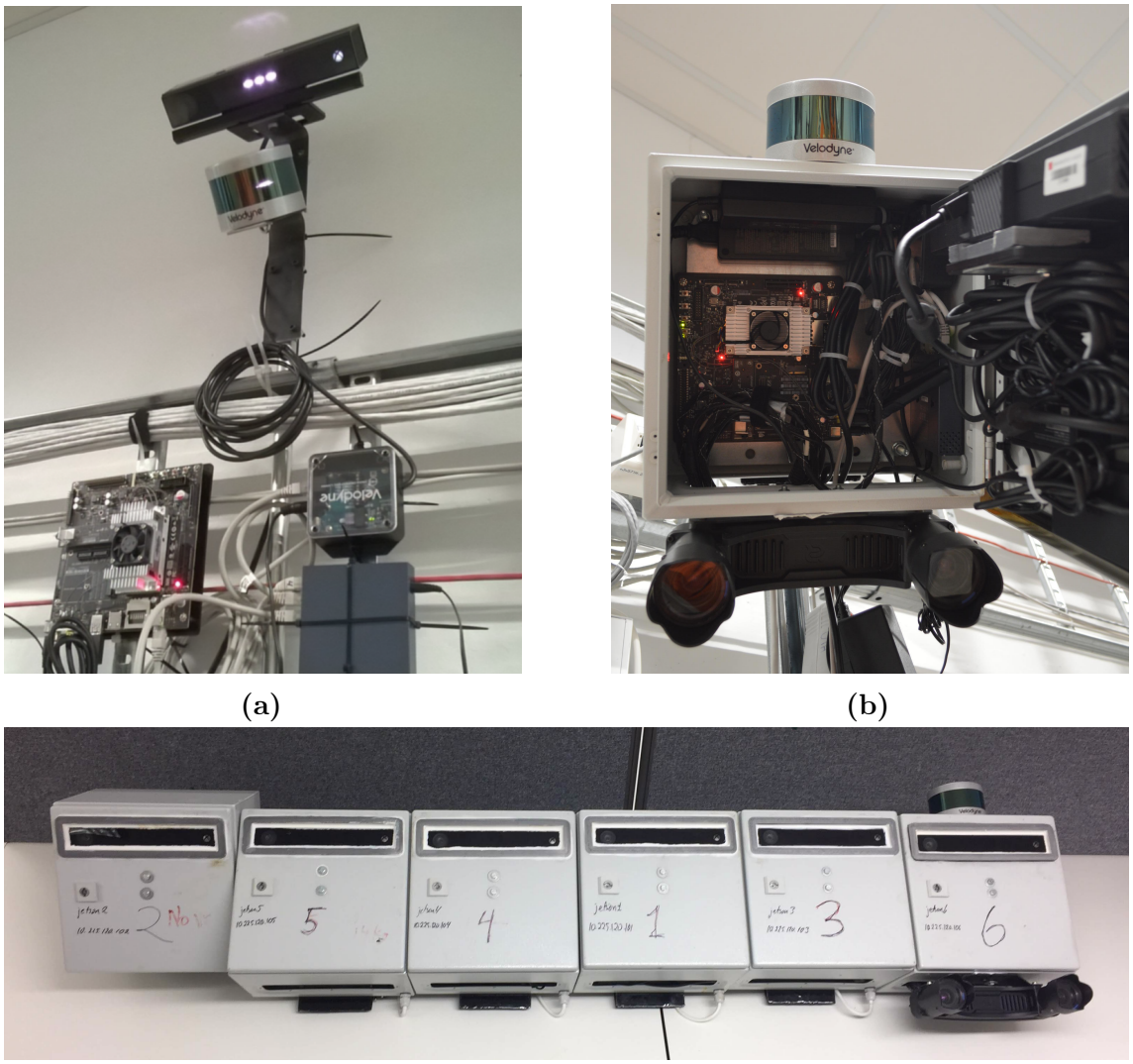
The Carnegie Robotics Stereo Vision Camera was selected to review the state-of-the art in industrial stereo vision in comparison to the consumer grade Kinect V2 sensor.

4.2 3D Sensor Package

Part of the motivation for this thesis, outlined in Section 1.1, was to build a sensor package for offshore use. To create a prototype for outdoor use, a water tight cabinet was chosen, where the sensors and related hardware were mounted. To make it as compact as possible, the already waterproof VLP-16 and Multisense S21 were mounted externally, and the Kinect V2 and embedded computer were mounted internally (see Figure 4.2). Five additional sensor packages were made, which only incorporated the Microsoft Kinect V2.

The 3D Sensor package also needed an embedded computer to be able to gather sensor data (RGB-D and point cloud streams), perform compression, and transfer the data to a central server. After reviewing alternatives, the NVIDIA Jetson TX2 was selected. At the time, this was the state-of-the-art embedded computer from NVIDIA, capable of handling the data volume produced by the connected sensors. It also included a GPU used for depth image processing.

To test the ruggedness of the package, it was mounted in an exposed location



(a) **(b)**

(c)

Figure 4.2: (**a**): The first prototype of a sensor package; (**b**): Embedded sensor package consisting of an NVIDIA Jetson TX2, a Kinect V2, a Velodyne VLP-16 PUCK Lidar and a Carnegie Robotics S21; (**c**) The six sensor packages used to cover the entire Industrial Robotics Lab.

outdoors on the roof of the University of Agder for approximately one month. The system was operational for the duration of this time, proving the ruggedness of the sensor package. However, it is clear that for operation in offshore environments, especially on drilling rigs, more work is needed to ensure the integrity of the sensor package, including an EX (explosive environment) certified enclosure.

4.3 Other Hardware and Prototyping Environment

In cooperation with fellow researchers in the SFI Offshore Mechatronics research centre, a prototyping framework was installed in the Industrial Robotics Lab. A

central server was custom built and installed in a server rack along with a switch that connected the server, the 6 sensor nodes and the ABB robot control system in a closed, wired Ethernet network.

On both the sensor nodes and server, the Robot Operating System (ROS) [54] was chosen as the middleware for running most of the software produced in this project. As ROS was gaining popularity in the research community, it allowed for rapid prototyping and testing due to already available drivers for sensors, an efficient communication protocol using a publish and subscribe scheme, and the ability to create and deploy custom packages, known as ROS nodes. For a detailed description of the sensor nodes and ROS system, see Paper D.

The goal of the experimental setup was to cover the IRL with the 3D sensors, and the sensor packages were therefore placed around the outer perimeter of the lab, with their field of views pointing inward. Once the sensors were mounted, research was performed to find the best calibration scheme to transform the point clouds generated by all six sensor nodes into a common coordinate system. In [55] Aalerud, Dybedal, et al. presented a manual calibration approach, and in [56] and [35] Ujkani, Aalerud, Dybedal et al. performed intrinsic sensor calibration and proposed an automatic extrinsic calibration scheme based on ArUco markers.

As the research progressed in parallel, the hardware and calibration techniques used in the papers attached to this thesis evolved over time. The exact specifications and methods at the time are specified in each paper. For the same reason, the sensor placement optimisation and occlusion minimisation methods presented in Papers B and C were not used when the sensors were originally mounted, but the robotics lab was used in one of the case studies in Paper C.

Together, the calibration schemes, sensor packages, and computational framework provided an efficient sensor fusion solution, where, when calibrated correctly, data from "any" sensor connected to a sensor package were transformed into the same voxel space and the same format. The transmitted data from each sensor was time-synchronised using synchronised clocks and timestamps. Using the point cloud intensity and filtering technique from Paper D, any overlapping sensor field-of-views meant a stronger intensity/confidence value in the merged point cloud. Once operational, the prototype environment provided this author and other researchers in the SFI Offshore Mechatronics with a ready-to-use real-time point cloud stream of the Industrial Robotics Lab. Figure 4.3, adapted from Paper D, shows how the system merges both RGB-D and lidar based point clouds into a single voxelised space.

In August 2018, the sensor packages were temporarily moved to an outdoor industrial testing facility at MHWirth AS (now HMH) in Kristiansand, Norway. Here, the dataset published in [57] was recorded. The dataset was part of a total

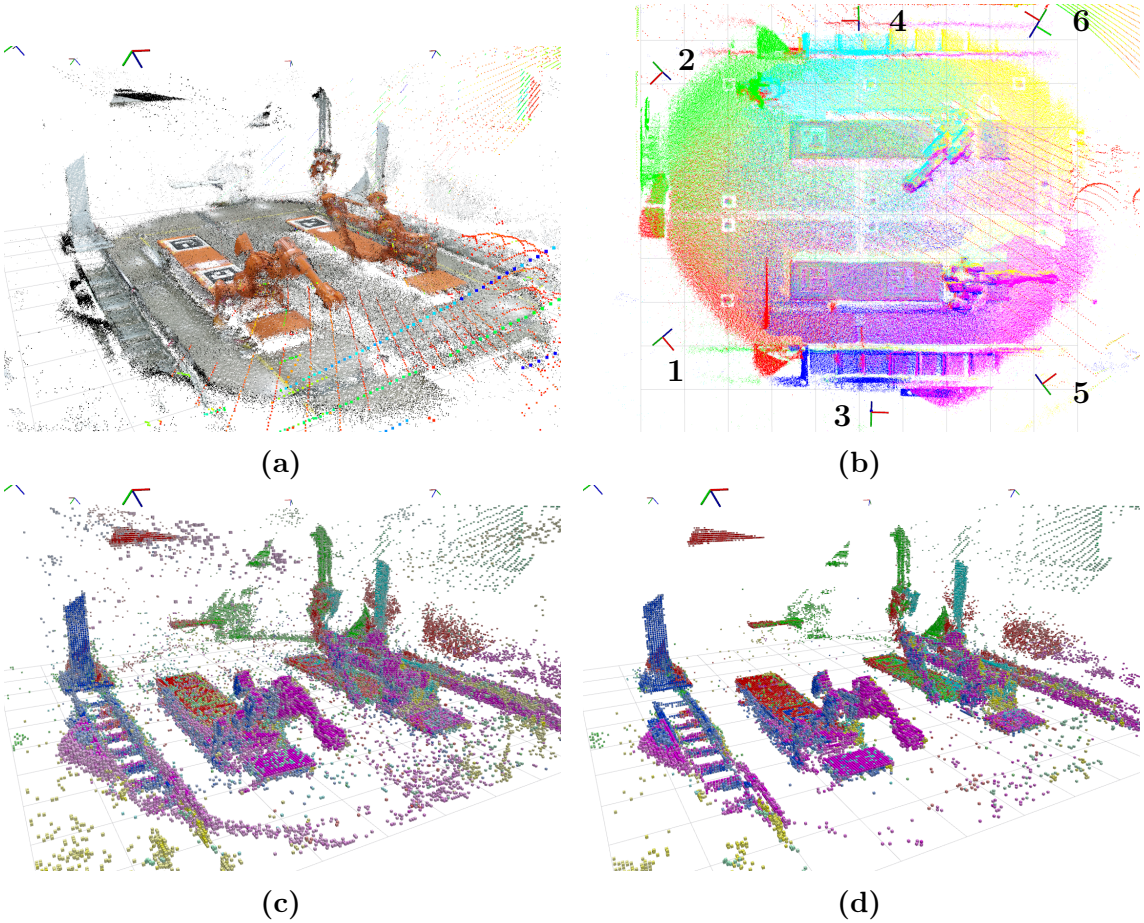


Figure 4.3: (**a**) Original point clouds from all sensors, including colour information for the Kinect V2 sensors and points from the VLP-16 lidar; (**b**) Top down view of original point clouds, where the different sensors' points are colour coded and the sensor packages are numbered; (**c**) Voxelised and merged, but unfiltered point cloud; (**d**) Voxelised and filtered point cloud.

of 1.4 terrabytes of captured data including colour images, IR images, lidar scans and depth maps. The dataset contains point cloud streams for different weather conditions such as heavy rain and low sun, and was later used in Paper E to train and validate the people detection method. A birds-eye-view of the test facility can be seen in Figure 4.4.

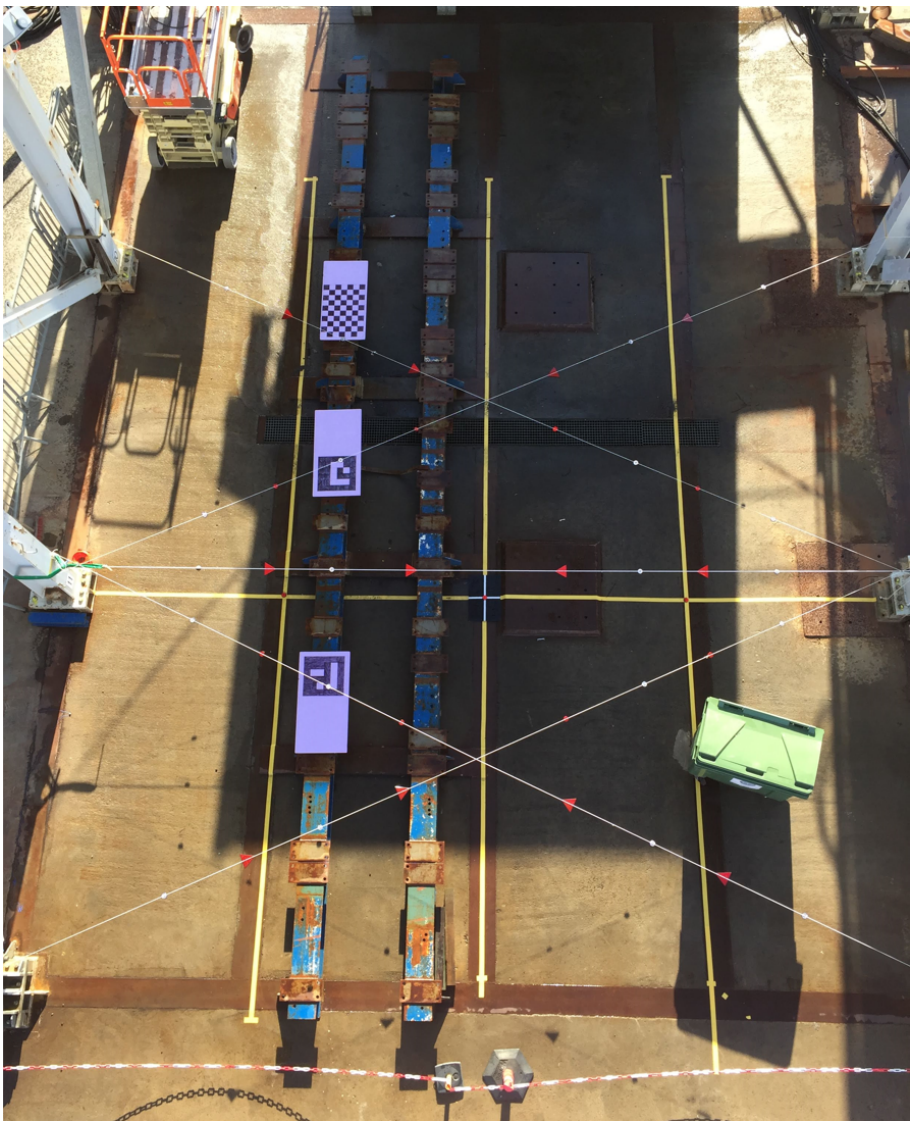


Figure 4.4: Overview of the outdoor testing facility at MHWirth.

Chapter 5

People Detection

The work presented in Paper E contains a solution for people detection in a voxelised three-dimensional space. This chapter presents relevant background information on different people detection techniques.

In the last decade, computer vision systems have become a more and more important part of different forms of systems: Modern cars, from conventional to fully autonomous, drones, logistics robots and vacuum cleaners are all examples of systems where computer vision sensors are used to detect obstacles both for security reasons and for autonomy. A common factor for these systems is that they are mobile, and have sensors mounted on them to help them navigate.

Another approach, as the one taken in this thesis, is to use computer vision systems to map a defined area, using multiple static sensors looking at the entire scene. This approach allows vehicles or equipment that do not have "eyes" of their own to be monitored and instructed. The Industrial Robotics Lab at the University of Agder is one example of such an area, where conventional robots without any vision systems are operational inside a cell where other obstacles could be present.

The following sections give a brief introduction to the methods for human detection used in this thesis, and how the accuracy measures are calculated.

5.1 Statistical Analysis of Binary Classification Performance

In the paper E, a method for classifying humans in a three dimensional space was developed. The classification was binary as the candidates were either classified as human or not human. This section provides a short introduction to binary classification with emphasis on the metrics used to measure classification accuracy.

Machine learning is much more than artificial intelligence and black boxes that magically learns features of datasets. The field of machine learning stretches back to

the 1960s [58] and is based on statistical analysis methods. The classic classifiers such as the Naïve Bayesian classifier, and random environment learning machines are all using statistical approaches to learn and predict.

Whether a legacy or a state-of-the art classifier is used, a common way to describe its accuracy is needed to be able to compare performances. One such approach is the Confusion Matrix (see Figure 5.1). This matrix shows how many of the predictions that were correct (the accuracy, shown on the diagonal) and how many that were assigned to each of the other possible classes. (In non-binary classifiers, the matrix can be extended, with correct predictions for each class on the diagonal). Thus the matrix shows the complete picture, including false positives and false negatives for each class.

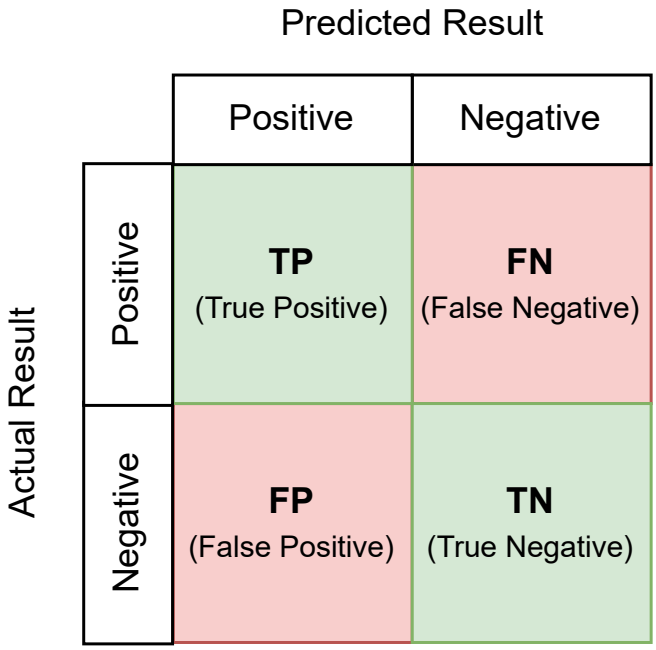


Figure 5.1: Example of a binary (2-class) Confusion Matrix.

If a classifier is not 100 % accurate, two other metrics known as precision and recall are often used to describe the performance [59]. The precision of the classifier is defined as the number of correct predictions divided by the total number of predictions of that class:

$$\text{precision} = \frac{tp}{tp + fp}, \tag{5.1}$$

where tp is true positives and fp is false positives. This measure then describes how many of the predictions are actually true. On the other hand, the recall of the classifier is defined as the number of correct predictions divided by the number of actual occurrences of that class:

$$\text{recall} = \frac{tp}{tp + fn}, \tag{5.2}$$

where fn is false negatives. The recall thus describes how often the classifier correctly predicts (or recalls) a given class.

Precision and recall must be used together to describe the performance of a classifier, as it is very well possible to have 100 % recall (every occurrence of a 'class1' is correctly predicted), but very low precision (all other classes were also classified as 'class1').

Another frequently used method to describe the performance using only a single metric is the F1-score, or harmonic mean, defined as

$$F_1 = 2\frac{p_p \cdot p_r}{p_p + p_r}, \tag{5.3}$$

where $0 \leq p_p \leq 1$ is the precision and $0 \leq p_r \leq 1$ the recall.

Although this measurement takes both precision and recall into account, it does ignore true negatives and it weighs precision and recall equally, which might not be desirable. A weighted version can be written as

$$F_\beta = (1 + \beta^2)\frac{p_p \cdot p_r}{\beta^2 \cdot p_p + p_r}, \tag{5.4}$$

where $\beta > 1$ means precision is more important, and vice versa. In both methods, the score is zero when either precision or recall is zero, and the maximum score is 1.0 when both precision and recall is 1.0.

Accuracy and the F1 score, which was used to describe the performance of the classifier developed in Paper E, has been among the most widely used statistical rates to evaluate classifiers, but other metrics such as the Matthews Correlation Coefficient (MCC) are suggested to be more precise and reliable [60], and might take over as a standard measurement in the future.

5.2 Detection Based on Images

The detection of humans in images can be generically described as the following process: Using an image as input, extract candidate regions where humans could be present and use a method to find features in the regions to describe the content. Then, classify the candidate regions as either human or not human.

For machine learning and deep learning based people detection methods, the following is a brief summary of the necessary steps.

1. **Training** - The training phase is typically performed on annotated images, where the true value is known (human or not human). Also known as supervised learning, this approach is most common in computer vision. A dataset of annotated images must be gathered, annotated and fed to a training model.

2. **Testing** - The trained model is tested using a subset of the annotated dataset which was not used in the training phase. This results in metrics used to measure the accuracy of the model, as described in the previous section.

3. **Detection and Classification** - The trained and tested model is used on data from a production environment to detect candidate regions and classify them as human/not human. Some methods perform both the detection and classification steps simultaneously – these are known as one-stage detectors, while the opposite is known as a two-stage detector.

There are multiple approaches to selecting candidate regions. Without prior knowledge, candidate regions, or windows, of different positions and sizes can be randomly selected, or systematically checked using a sliding window. After a first round of classification, the regions would then be merged and adjusted based on the classification result.

With some prior knowledge of the input data, the number of candidate regions can be reduced. If the background of an image is known (i.e. a scene with no humans or other foreign objects present), background subtraction could be used to extract the initial candidate regions. Or, if the source is a video, subsequent images could be compared to extract regions with movement.

In the end, the aim is that a human is enclosed by just one region. Thus, the output of the detection method is typically a four-sided bounding box surrounding the person. Instead of a bounding box, segments of an image can be classified as human, another object, or an abstract class such as "road" or "forest". This is called semantic segmentation, which essentially assigns a category to each pixel in the image. However, segmentation is outside the scope of this thesis and will not be discussed further.

Popular human detection methods based on images often use edge-based feature descriptors such as histogram of oriented gradients (HOG). It has been found that such shape-type features (rather than appearance) often yield better results, and the HOG approach has been widely adopted [61]. Classifiers such as Support Vector Machines (SVMs) are typically used together with HOGs to detect humans [62].

In recent years, deep learning classifiers such as the convolutional neural network (CNN) has gained popularity, see for example [63, 64]. A wide variety of neural network detectors have been developed and improved over time, typically aiming to not only detect humans, but classify objects in large datasets such as the KITTI dataset for autonomous driving [65] and the COCO (Common Objects in Context) dataset for object detection and segmentation [66]. Some examples of these models are Faster RCNN [67] and YOLO (You Only Look Once) [68]. Originally based on the DarkNet [69] framework, YOLO has been developed further by multiple

instances, such as YOLO by Ultralytics [70] and YOLOv7 [71].

YOLO is one example of a one-stage detector, meaning the two phases of selecting a region-of-interest and then classifying each region is merged into a single operation, producing bounding boxes that are already classified. These detectors are typically used on live video streams.

For a comprehensive review of inner workings of deep learning algorithms, see [72].

5.3 Detection Based on Point Clouds

The previous section described detection of humans or other objects in images. Adding depth information has several benefits in applications such as 3D collision detection. The most obvious is the additional dimension that enables direct measurement of the distance to and position of a detected object or person.

In [73], people detection was performed on RGB-D data from a Microsoft Kinect sensor. The HOG detector was used as an inspiration to create a HOD (Histogram of Oriented Depths) detector, looking at the direction of depth changes in a depth image. This detector was then combined with a HOG detector for the RBG part of the image, resulting in a combination, coined as Combo-HOG. The Combo-HOG detector was used together with an SVM classifier to detect people in the RGB-D image.

For pure point clouds with no RGB component, as used in Paper E, one of the most used data sources in the literature is point clouds from a lidar sensor, and the application is typically pedestrian tracking for autonomous driving, or people detection by mobile robots. As in Paper E, many of the methods are based on deep learning. In the literature, three categories for deep learning-based 3D object detection seem to be dominating:

1. Detectors mapping the 3D data to a 2-dimensional representation used as input to a more conventional convolutional neural network.

2. Detectors converting the point cloud do a discretised spaced, i.e, a voxel space.

3. Detectors working directly on the 3D point cloud

Combinations, or fusions, of the approaches also exist.

Expanding on the second version of the 2D YOLO detector described in the previous section, Complex-YOLO [74] was one of the first real-time capable 3D detectors working on lidar data only. As one example of the first category, Complex-YOLO converts a lidar point cloud to a bird's-eye-view RGB map, where the colour channels are encoded with height, density and intensity. The work included a

custom Euler-region proposal network (E-RPN) for estimating the heading of the 3D bounding boxes.

An example of the second category is presented in [75]. Building on the sliding window approach for selecting candidate regions in two dimensions, the paper presents a method to efficiently use a sliding window technique in three dimensions, while exploiting the fact that point clouds tend to be sparse, thus significantly reducing the amount of computation compared to classifying every window.

In [76], VoxelNet was proposed, a generic, voxel-based, deep architecture for object detection in 3D point clouds, which outperformed other state-of-the-art lidar based detection methods when evaluated on the KITTI benchmark. This method removed the need for hand-crafted feature descriptors and introduced a voxel feature encoding (VFE) layer in its one-stage detector architecture.

In [77], a novel approach based on a graph neural network (GNN) is an example of the third category presented above. This paper also presented promising results on the KITTI dataset. A GNN is described as the following in the paper: *"We encode the point cloud natively in a graph by using the points as the graph vertices. The edges of the graph connect neighbourhood points that lie within a fixed radius, which allows feature information to flow between neighbours. Such a graph representation adapts to the structure of a point cloud directly without the need to make it regular. A graph neural network reuses the graph edges in every layer, and avoids grouping and sampling the points repeatedly"*. This paper also gives a good overview of the differences, advantages and disadvantages of different point cloud based detection approaches, and is recommended for further information on the topic.

The research in Paper E falls in both the first and second category. Here, a fusion of the two were used to detect, classify and segment humans in three dimensions. First, the point cloud was implicitly converted to the voxel space by the compression and fusion method presented in Paper D. Then, after segmenting out candidate clusters, two 2D images were generated by flattening the clusters onto the planes defined by the xz and yz axes. The images were used for classification, and the point cloud coordinates were used to get the coordinates for a three dimensional bounding box. In other words, the whole point cloud cluster representing each human was surrounded by a cube which would change dimensions with the pose of the human.

Chapter 6

Concluding Remarks

The work presented in this thesis focused on three main efforts: 1) to automatically optimise the placement of computer vision sensors in a three dimensional environment, 2) to create a 3D sensor package and sensor fusion network, and 3) a people detection method. A common denominator of all the appended papers is the voxel space: A volume divided into equally sized cubes, which allowed for massively parallel operations during placement optimisation, and served as a common format when building the 3D sensor network and the people detection method.

In this chapter, the conclusions to the research questions presented in Chapter 1 are presented. In addition, ideas and suggestions for future work based on this thesis are given.

6.1 Conclusions

The optimal placement of vision sensors in a 3D environment is a hard problem to solve. It is a three dimensional variant of the "Art Gallery Problem", where the aim is to position as few guards as possible in a gallery, but in such a way that all walls must be viewed by at least one guard. The art gallery problem has been proved to be NP-hard [78], so solving a variant of this task in three dimensions was not expected to be easy.

In Paper A, Mixed Integer Linear Programming (MILP) was used to solve the placement problem, and this showed good results. Here, a cube shaped volume was divided into smaller cuboids, or voxels, and the objective of the optimisation was to maximise the number of cubes that were viewed by the sensors when considering limited range and a cone-shaped field of view. Although the method yielded optimal solutions, it was found to not be easily scalable due to the increase in variables introduced by custom, piecewice linearisation of non-linear functions. Thus, getting an optimal solutions was time consuming.

A different approach, using random sampling and the massively parallel processing

power of a GPU was introduced by Paper B. Using the same concept of dividing the volume into smaller voxels, this method provided near-optimal results to scaled-up versions of the problem while using a fraction of the computation time. The method also introduced redundancy constraints to be able to define important sub-volumes that should be viewed by at least n sensors. The promising results and short computation times led to this method being chosen as the basis for further work, which was presented in Paper C. Here, a more realistic scenario of the problem was solved, introducing occlusions, i.e. objects blocking the field of view of the sensors. To combat this problem, an occlusion detection method was developed, and added as an extension to the placement optimisation solver. The solver was modified to account for the pyramid shaped field of view typically found in vision based senors, and to be able to optimise the pose of the sensors in addition to position. The method yielded near-optimal positions of 6 sensors in a volume consisting of 28.800 voxels where occluded objects were present.

A significant amount of work was put in to establish the prototype environment at the Industrial Robotics Lab at the University of Agder. As described in Chapter 4, this work was a joint effort in the SFI Offshore Mechatronics research centre, where this project focused on the sensor packages, edge computing, compression/decompression and sensor fusion to enable other fellow researchers to use the merged point cloud stream of the lab in their research. Calibration methods were developed in [55, 56, 35], and after an extensive state-of-the-art review and prototyping phase, Paper D presented the developed 3D sensor package and a highly scalable sensor fusion method, including a novel point cloud compression scheme to allow six of the sensor packages to transfer data from individual edge computers to a central server. The compression ratio of 40.5 and a very low bandwidth would allow the system to theoretically scale up to 440 sensors. In addition, a filtering technique based on a custom voxel intensity was presented in the same paper.

The sensor packages, methods, and system presented in Paper D was then used when developing the people detection scheme published in Paper E. Using the fused, voxelized, and filtered point clouds from the sensor packages as input, the detection system was able to detect and classify humans with an F1 score of 0.87 and a position accuracy of 4 cm in an outdoor environment. This was accomplished through point cloud segmentation, flattening into 2D images, and a CNN based image scene classifier. This paper also showed that the developed sensor package performed well in an outdoor scenario, despite the fact that the main sensor, the Microsoft Kinect V2, was designed for indoor consumer use.

Based on Papers D and E, it can be concluded that it is possible to create a 3D sensor package for outdoor use using off-the-shelf components, and that it is feasible to develop this further for offshore use, and use it to detect people in possibly

hazardous areas. Indeed, the relevance of the research presented in this thesis can be confirmed by the recent commercial interest in similar sensor systems, such as HaloGuard by Salunda [79] and VisionIQ by the Marsden Group [80]. In addition, the open source software, methods and datasets published during this project will also be suitable to enable computer vision aided monitoring in other scenarios, such as indoor industrial facilities similar to the prototyping environments presented in Chapter 4.

Another result of the work performed by the Robotics and Autonomy work package at SFI Offshore Mechatronics, which has not been discussed earlier, was a patented reflector which can be mounted on a standard spinning lidar to redirect the laser beams in a forward facing direction, effectively changing the field-of-view of the lidar [81]. This opens up new, interesting use cases for a spinning lidar, and was a direct result of the industry- and technology-oriented research in the research centre.

6.2 Future Work

This thesis has presented technology that could be used by the industry to enable the use of more computer vision in industrial and/or offshore environments. For the 3D sensor placement solver, future work should include testing the solution on a more complex, industrial volume. Using the optimiser on a drilling rig's derrick would be optimal, where both the occlusion detection and redundancy constraints would be very relevant. In addition, as the solver allows to restrict the free position variables, possible mounting locations such as beams or pillars could be added as part of the constraints.

The sensor packages and computational platform should be piloted in an operational industrial environment. To accomplish this, the prototype sensor packages should be "industrialized" by selecting rugged versions of components such as the NVIDIA Jetson, and enclosures certified for the applicable environments. In applications where real-time performance is critical, work should also be done to port the software from the Robot Operating System to an industrial real-time system. This should also include optimising the software by e.g. using more of the GPU capabilities in the embedded computers and on the server. And in light of recent advances in the field, using a mm-wave radar as part of the sensor package should be re-evaluated.

Testing the people detection method in an outdoor environment proved its ability to perform well in sub-optimal conditions. Using this method and the published software as a baseline, the method could be developed further in multiple aspects. To decrease the computation time, the software should be ported from Matlab to a more efficient framework. In addition, adding people identification, tracking and

prediction could be a natural step forward. This could be used for e.g. human-machine anti collision software to stop a machine's movement if a person is heading in that direction, and to keep track of the personnel's movements in and out of restricted or hazardous areas. To achieve this, a classification method that takes the temporal element of the point cloud stream into account should be investigated, i.e. exploiting the fact that the same person will be present in consecutive point clouds in approximately the same location.

Bibliography

[1] SFI Offshore Mechatronics. https://sfi.mechatronics.no/. [Online; accessed June 19, 2023].

[2] European Association of Research & Technology Organisations. The TRL Scale as a Research & Innovation Policy Tool, EARTO Recommendations. https://www.earto.eu/wp-content/uploads/The_TRL_Scale_as_a_R_I_Policy_Tool_-_EARTO_Recommendations_-_Final.pdf. [Online; accessed June 19, 2023].

[3] Amir Mukhtar, Likun Xia, and Tong Boon Tang. Vehicle Detection Techniques for Collision Avoidance Systems: A Review. *IEEE Transactions on Intelligent Transportation Systems*, 16:2318–2338, 10 2015.

[4] Xiaowei Hu, Ningning Tong, Yongshun Zhang, Guoping Hu, and Xingyu He. Multiple-input–multiple-output radar super-resolution three-dimensional imaging based on a dimension-reduction compressive sensing. *IET Radar, Sonar & Navigation*, 10:757–764, 2016.

[5] Tesla Shifting Autopilot from Camera to Radar-Based Sensing. http://fortune.com/2016/09/11/tesla-autopilot-shift/. [Online; accessed June 19, 2023].

[6] Uber starts self-driving car pickups in Pittsburgh. https://techcrunch.com/2016/09/14/1386711/. [Online; accessed June 19, 2023].

[7] Alireza Asvadi, Cristiano Premebida, Paulo Peixoto, and Urbano Nunes. 3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes. *Robotics and Autonomous Systems*, 83:299–311, 2016.

[8] IEEE Spectrum. Quanergy Announces $250 Solid-State LIDAR for Cars, Robots, and More - IEEE Spectrum. http://spectrum.ieee.org/cars-that-think/transportation/sensors/quanergy-solid-state-lidar. [Online; accessed June 19, 2023].

[9] Tojiro Kaneko, Hidehisa Akiyama, and Shigeto Aramaki. Detection of Indoor Mobile Robots Using Objective Sensor without Markers. *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 572–575, 2016.

[10] Zivid AS. About Zivid. https://www.zivid.com/about. [Online; accessed Feb. 12, 2023].

[11] Stereolabs Inc. ZED Stereo Cameras. https://www.stereolabs.com/. [Online; accessed Feb. 12, 2023].

[12] Stefano Mattoccia and Matteo Poggi. A passive RGBD sensor for accurate and real-time depth sensing self-contained into an FPGA. In *Proceedings of the 9th International Conference on Distributed Smart Cameras*, pages 146–151, 2015.

[13] NVIDIA Corporation. Jetson TX1 Embedded Systems Module. https://developer.nvidia.com/embedded/jetson-tx1. [Online; accessed June 19, 2023].

[14] Aitech. A176 Cyclone | GPGPU Fanless SFF Supercomputer. https://aitechsystems.com/product/a176-cyclone-gpgpu/. [Online; accessed June 19, 2023].

[15] Gartner Inc. Gartner's 2016 Hype Cycle for Emerging Technologies Identifies Three Key Trends That Organizations Must Track to Gain Competitive Advantage. https://www.gartner.com/en/newsroom/press-releases/2016-08-16-gartners-2016-hype-cycle-for-emerging-technologies-identifies-three-key-trends-that-organizations-must-track-to-gain-competitive-advantage. [Online; accessed June 19, 2023].

[16] Christian Stimming, Annette Krengel, Markus Boehning, Andrei Vatavu, Szilárd Mandici, and Sergiu Nedevschi. Multi-level on-board data fusion for 2D safety enhanced by 3D perception for AGVs. In *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 239–244. IEEE, 2015.

[17] Seungki Kim, Hyunkyu Kim, Wonseok Yoo, and Kunsoo Huh. Sensor Fusion Algorithm Design in Detecting Vehicles Using Laser Scanner and Stereo Vision. *IEEE Transactions on Intelligent Transportation Systems*, 17:1072–1084, 4 2016.

[18] Bahador Khaleghi, Alaa Khamis, Fakhreddine O. Karray, and Saiedeh N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14:28–44, 2013.

[19] Jürgen Kemper, Markus Walter, and Holger Linde. Human-assisted calibration of an angulation based indoor location system. In *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, pages 196–201, 2008.

[20] Jesse Levinson and Sebastian Thrun. Automatic Online Calibration of Cameras and Lasers. *Robotics: Science and Systems (RSS)*, 2(7), 2013.

[21] Abhinav Valada, Ankit Dhall, and Wolfram Burgard. Convoluted mixture of deep experts for robust semantic segmentation. In *IEEE/RSJ International conference on intelligent robots and systems (IROS) workshop, state estimation and terrain perception for all terrain mobile robots*, volume 2, 2016.

[22] Nicolaj Kirchhof. Optimal placement of multiple sensors for localization applications. *2013 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2013*, pages 28–31, 2013.

[23] Eva Hörster and Rainer Lienhart. On the optimal placement of multiple visual sensors. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 111–120, 2006.

[24] Uğur Murat Erdem and Stan Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Computer Vision and Image Understanding*, 103:156–169, 2006.

[25] H. Topcuoglu, M. Ermis, I. Bekmezci, and M. Sifyan. A new three-dimensional wireless multimedia sensor network simulation environment for connected coverage problems. *SIMULATION*, 88:110–122, 1 2012.

[26] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 3 1999.

[27] Matteo Poggi, Fabio Tosi, Konstantinos Batsos, Philippos Mordohai, and Stefano Mattoccia. On the Synergies Between Machine Learning and Binocular Stereo for Depth Estimation From Images: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5314–5334, 2022.

[28] Guodong Zhai, Wentao Zhang, Wenyuan Hu, and Zhendong Ji. Coal Mine Rescue Robots Based on Binocular Vision: A Review of the State of the Art. *IEEE Access*, 8:130561–130575, 2020.

[29] Boyu Gao, Haoxiang Lang, and Jing Ren. Stereo visual slam for autonomous vehicles: A review. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1316–1322, 2020.

[30] Intel Corporation. Intel RealSense Computer Vision - Depth and Tracking Cameras. https://www.intelrealsense.com/. [Online; accessed Feb. 12, 2023].

[31] Zivid AS. 3D Structured Light Vision. https://www.zivid.com/3d-structured-light. [Online; accessed Feb. 12, 2023].

[32] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, and E. Menegatti. Performance evaluation of the 1st and 2nd generation Kinect for multimedia applications. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2015.

[33] Jason Geng. Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160, Jun 2011.

[34] Radu Horaud, Miles Hansard, Georgios Evangelidis, and Clément Ménier. An overview of depth cameras and range scanners based on time-of-flight technologies. *Machine Vision and Applications*, 27:1005–1020, 10 2016.

[35] Atle Aalerud, Joacim Dybedal, and Geir Hovland. Automatic Calibration of an Industrial RGB-D Camera Network Using Retroreflective Fiducial Markers. *Sensors*, 19(7), 2019.

[36] Dingkang Wang, Connor Watkins, and Huikai Xie. MEMS Mirrors for LiDAR: A Review. *Micromachines*, 11(5), 2020.

[37] Ching-Pai Hsu, Boda Li, Braulio Solano-Rivas, Amar R. Gohil, Pak Hung Chan, Andrew D. Moore, and Valentina Donzella. A Review and Perspective on Optical Phased Array for Automotive LiDAR. *IEEE Journal of Selected Topics in Quantum Electronics*, 27(1):1–16, 2021.

[38] Shizhe Zang, Ming Ding, David Smith, Paul Tyler, Thierry Rakotoarivelo, and Mohamed Ali Kaafar. The Impact of Adverse Weather Conditions on Autonomous Vehicles: How Rain, Snow, Fog, and Hail Affect the Performance of a Self-Driving Car. *IEEE Vehicular Technology Magazine*, 14(2):103–111, 2019.

[39] Christian Waldschmidt, Juergen Hasch, and Wolfgang Menzel. Automotive Radar — From First Efforts to Future Systems. *IEEE Journal of Microwaves*, 1(1):135–148, 2021.

[40] Nicolas Scheiner, Ole Schumann, Florian Kraus, Nils Appenrodt, Jürgen Dickmann, and Bernhard Sick. Off-the-shelf sensor vs. experimental radar - How much resolution is necessary in automotive radar classification? In *2020 IEEE*

23rd International Conference on Information Fusion (FUSION), pages 1–8, 2020.

[41] Peijun Zhao, Chris Xiaoxuan Lu, Jianan Wang, Changhao Chen, Wei Wang, Niki Trigoni, and Andrew Markham. mID: Tracking and Identifying People with Millimeter Wave Radar. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 33–40, 2019.

[42] Han Cui and Naim Dahnoun. High Precision Human Detection and Tracking Using Millimeter-Wave Radars. *IEEE Aerospace and Electronic Systems Magazine*, 36(1):22–32, 2021.

[43] Zhongfei Ni and Binke Huang. Gait-Based Person Identification and Intruder Detection Using mm-Wave Sensing in Multi-Person Scenario. *IEEE Sensors Journal*, 22(10):9713–9723, 2022.

[44] Youngwook Kim, Ibrahim Alnujaim, and Daegun Oh. Human Activity Classification Based on Point Clouds Measured by Millimeter Wave MIMO Radar With Deep Recurrent Neural Networks. *IEEE Sensors Journal*, 21(12):13522–13529, 2021.

[45] Andreas Hermann, Florian Drews, Joerg Bauer, Sebastian Klemm, Arne Roennau, and Ruediger Dillmann. Unified GPU voxel collision detection for mobile manipulation planning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4154–4160, 2014.

[46] Donald J. R. Meagher. *Octree encoding: a new technique for the representation, manipulation and display of arbitrary 3-D objects by computer*. Rensselaer Polytechnic Institute, Oct 1980. Print.

[47] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[48] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach. Real-time compression of point cloud streams. In *2012 IEEE International Conference on Robotics and Automation*, pages 778–785, 2012.

[49] L. Peter Deutsch. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951, May 1996.

[50] International Organization for Standardization. Information technology — Plenoptic image coding system (JPEG Pleno) — Part 1: Framework (ISO/IEC 21794-1:2020), 2020.

[51] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A. Chou, Robert A. Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, Joan Llach, Khaled Mammou, Rufael Mekuria, Ohji Nakagami, Ernestasia Siahaan, Ali Tabatabai, Alexis M. Tourapis, and Vladyslav Zakharchenko. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2019.

[52] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing*, 9:e13, 2020.

[53] Xian-Feng Han, Jesse S. Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, and Liping Xiao. A review of algorithms for filtering the 3D point cloud. *Signal Processing: Image Communication*, 57:103–112, 2017.

[54] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3(2), page 5. Kobe, Japan, 2009.

[55] Atle Aalerud, Joacim Dybedal, Erind Ujkani, and Geir Hovland. Industrial Environment Mapping Using Distributed Static 3D Sensor Nodes. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6, 2018.

[56] Erind Ujkani, Joacim Dybedal, Atle Aalerud, Knut Berg Kaldestad, and Geir Hovland. Visual Marker Guided Point Cloud Registration in a Large Multi-Sensor Industrial Robot Cell. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6, 2018.

[57] Joacim Dybedal. Replication Data for: CNN-based People Detection in Voxel Space using Intensity Measurements and Point Cluster Flattening. *DataverseNO*, 2021.

[58] N.J. Nilsson. *Learning Machines*. McGrawHill, New York, 1965.

[59] Michael Buckland and Fredric Gey. The relationship between Recall and Precision. *Journal of the American Society for Information Science*, 45(1):12–19, 1994.

[60] Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21:6, 12 2020.

[61] Duc Thanh Nguyen, Wanqing Li, and Philip O. Ogunbona. Human detection from images and videos: A survey. *Pattern Recognition*, 51:148–175, 3 2016.

[62] Luciano Spinello and Roland Siegwart. Human detection using multimodal and multidimensional features. In *2008 IEEE International Conference on Robotics and Automation*, pages 3264–3269, 2008.

[63] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018.

[64] Sasa Sambolek and Marina Ivasic-Kos. Automatic Person Detection in Search and Rescue Operations Using Deep CNN Detectors. *IEEE Access*, 9:37905–37922, 2021.

[65] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[66] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755. Springer, 2014.

[67] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[68] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[69] Joseph Redmon. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/, 2013–2016. [Online; accessed May 31, 2023].

[70] G. Jocher, A. Chaurasia, and J. Qiu. YOLO by Ultralytics. https://github.com/ultralytics/ultralytics. [Online; accessed May 31, 2023].

[71] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023.

[72] Ajay Shrestha and Ausif Mahmood. Review of Deep Learning Algorithms and Architectures. *IEEE Access*, 7:53040–53065, 2019.

[73] Luciano Spinello and Kai O. Arras. People detection in RGB-D data. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3838–3843, 2011.

[74] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[75] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: science and systems*, volume 1(3), pages 10–15. Rome, Italy, 2015.

[76] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[77] Weijing Shi and Raj Rajkumar. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[78] D. Lee and A. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

[79] Salunda Limited. HaloGuard - Unauthorised Personnel Detection. https://www.salunda.com/haloguard. [Online; accessed June 19, 2023].

[80] Velodyne Lidar Inc. Lidar Helping to Improve Safety in Offshore Drilling. https://velodynelidar.com/blog/lidar-helping-to-improve-safety-in-offshore-drilling/. [Online; accessed June 19, 2023].

[81] Atle Aalerud and Joacim Dybedal. Reflector for reflecting electromagnetic waves from a rotating electromagnetic wave source, 2023. European Patent EP3899604B1.

Paper D

Embedded Processing and Compression of 3D Sensor Data for Large Scale Industrial Environments

Joacim Dybedal, Atle Aalerud and Geir Hovland

This paper has been published as:

Joacim Dybedal, Atle Aalerud and Geir Hovland. Embedded Processing and Compression of 3D Sensor Data for Large Scale Industrial Environments. *Sensors*, 19(3):636, 2019.
doi: 10.3390/s19030636.

Embedded Processing and Compression of 3D Sensor Data for Large Scale Industrial Environments

Joacim Dybedal*, Atle Aalerud* and Geir Hovland*

*University of Agder
Faculty of Engineering and Science
Jon Lilletunsvei 9, 4879 Grimstad, Norway

Abstract **This paper presents a scalable embedded solution for processing and transferring 3D point cloud data. Sensors based on the time-of-flight principle generate data which are processed on a local embedded computer and compressed using an octree-based scheme. The compressed data is transferred to a central node where the individual point clouds from several nodes are decompressed and filtered based on a novel method for generating intensity values for sensors which do not natively produce such a value. The paper presents experimental results from a relatively large industrial robot cell with an approximate size of** $10\,\mathrm{m} \times 10\,\mathrm{m} \times 4\,\mathrm{m}$**. The main advantage of processing point cloud data locally on the nodes is scalability. The proposed solution could, with a dedicated Gigabit Ethernet local network, be scaled up to approximately** 440 **sensor nodes, only limited by the processing power of the central node that is receiving the compressed data from the local nodes. A compression ratio of** 40.5 **was obtained when compressing a point cloud stream from a single Microsoft Kinect V2 sensor using an octree resolution of** $4\,\mathrm{cm}$**.**

D.1 Introduction

The rapid development of 3D sensors based on the time-of-flight principle is currently an important enabling factor for different autonomous systems, such as self-driving vehicles or drones. However, there is also an increasing interest in using these types of sensors in other industries where the sensors are not moving with a vehicle or machine, but are mounted in fixed locations and monitoring a volume of interest. One such application is within the offshore oil and gas industry, where, in recent years, there has been a growing focus on automation, safety systems and better efficiency (see, for example, [D1]). This is highly motivated by cost reductions due to lower investment activity by exploration and production companies. Adding depth

cameras, lidar (Light Detection And Ranging) and other 3D sensors to existing and newly built facilities could contribute to enabling and accelerating this development.

One feature of the 3D sensors is the generation of a potentially large amount of data, requiring a high bandwidth between the sensor and the computer processing the data. For example, the Microsoft Kinect for Xbox One (Kinect V2) sensor (Microsoft Corporation, Redmond, WA, USA) can generate data of several Gbit/s on a USB3 connection. With a dedicated PCI card, it is possible to use a few such sensors on a single computer; however, scaling the solution up to many more sensors is difficult with a centralized solution. A first step to ensure scalability is to lower the bandwidth requirement using point cloud compression. The following subsection will describe previous methods for compressing point cloud data and the main contributions of this paper will be outlined in Section D.1.2.

D.1.1 Related Work

In [D2], a compression and decompression scheme for point cloud streams was presented. The paper presented a novel XOR-based differential octree used by the compressor to detect temporal changes between two consecutively measured point clouds. The authors in [D2] implemented both spatial and temporal compression of arbitrary point clouds originating from a single Kinect depth sensor. It was shown that octrees can be used to efficiently compress and encode such point cloud streams. The authors also showed that the double-buffered octree compression scheme was suitable for compressing even unstructured point clouds. Temporal compression was described to be especially effective when there is a limited amount of movement in the point cloud, which is the case in our application where most of the scenery is static and movement only comes from the machines and/or humans. In addition to the implicit grid filtering performed by inserting points into an octree with a fixed minimum voxel size, the points in the original point cloud could be reconstructed with a given precision at the decompressor due to an additional encoding of point details in each octree leaf node.

The underlying compression algorithm used in [D2] is a variant of an arithmetic entropy encoder known as an integer-arithmetic or range encoder originally presented in [D3]. In short, a range encoder utilizes the probability distribution of a set of symbols (e.g., bytes in this case) with a given range, and encodes the symbols into one number by dividing the initial range into sub-ranges based on the symbol probabilities. The reader is referred to [D3] for further details. The range encoder implemented by [D2] takes a stream of bytes as an input, and produces an encoded stream which can be stored or transmitted over a network.

A dynamic compression scheme based on [D2] was developed in [D4]. Using a

single Kinect depth sensor, the authors were able to achieve an average frame rate of 5.86 Hz on a network subjected to different levels of background transmissions. In our scenario, such background noise on the network is minuscule, as the entire network is designated to transfer data between the sensors, the central computer and the machines in the lab.

Another technique that has been shown to be useful for processing point clouds is graph signal processing (GSP). In [D5, D6], GSP was used to resample and compress point clouds, respectively. Resampling was achieved by applying different filters on the generated graphs and compression was partially based on the double-buffered octree scheme outlined above. Furthermore, in [D7], GSP was used to perform denoising by first removing outlier points based on a weighted graph (similar to a statistical outlier removal filter), and subsequently smoothing the point cloud by means of convex optimization on the graph signal. While these results show that GSP can be well suited for working with point clouds, they also show that the process can be time-consuming. For example, in [D5], the authors were able to process 15 million 3D points in 1000 s, equivalent to 15,000 points per second. However, since the Kinect V2 used in this paper generates point clouds consisting of 217,088 points at rates up to 30 Hz, it would require a much lower processing time to keep up with such frame rates.

In [D8], an octree based mapping framework (OctoMap) was proposed. The generated map was implemented as an occupancy grid, where voxels are labeled occupied, free or unknown based an occupancy probability. The work included a compression scheme combining clamping and octree pruning, where the tree is pruned when all children of a tree node are considered stable (the nodes have an occupancy probability close to 0 or 1). The framework, which also incorporates probabilistic sensor fusion, is intended to build a map of an environment, and thus is of interest in our scenario where the goal is to create a dynamic map of a robotic cell. If all sensors were directly connected to the central computer, OctoMap could be used to fuse the sensor data and generate a map of the complete area. However, the focus of this paper is to lower the amount of data generated by the sensors before the data is inserted into such a map, limiting the required network bandwidth between the sensors and the central computer, and ensuring scalability.

D.1.2 Main Contributions

Unlike in most literature where single sensors often are considered [D9, D4, D10], an industrial application will typically need multiple sensors to cover relatively large areas. To ensure scalability, a new approach is presented, where each sensor is connected to a local computer. The generated point cloud data is processed and

compressed on the embedded sensor node before the data is sent to a central computer for decompression and further processing.

To perform the compression, the method presented in [D2] has been implemented on embedded hardware. The method has further been modified by introducing an attribute that represents the "intensity" of each voxel. The modification was made in order to allow filtering of noisy measurements (outliers) at the central computer and to introduce a trust level indicator for each voxel. Implementation of such an intensity value for RGB-D (color and detph) sensors was also motivated by intensity values returned by lidar sensors and the potential fusion of data from different sensor types.

A compression ratio of 40.5 is achieved, and a denoising scheme based on the calculated point intensities is proposed. With such a setup, the scalability problem is solved by decentralization.

D.2 Materials and Methods

This paper describes a processing, compression and transmission framework for point cloud data generated by 3D sensors. The software is developed as two Robot Operating System (ROS, [D11]) nodes. The first ROS node is deployed on an NVIDIA Jetson TX2 module (NVIDIA Corporation, Santa Clara, CA, USA) which, in addition to a CPU, contains a general purpose graphical processing unit (GPGPU). The module may have one or more 3D sensors connected, and together they form a sensor node. By exploiting the processing capabilities of the Jetson TX2, data from the connected sensor(s) is processed and compressed locally before it is published on the ROS network. The second ROS node is deployed on a central computer and receives the compressed data from one or more sensor nodes. In the following subsections, the different processing steps are described. The developed software is available in public github repositories, see [D12].

D.2.1 Problem Formulation and Motivation

The sensor data from multiple time-of-flight 3D sensors is to be used as input to a "GPU Voxels"-based application [D13], a GPU based collision detection software which is running on a centralized computer. Similar to OctoMap, a voxel-based occupancy grid is created and, due to the fact that the map is stored in the GPU memory, calculations such as distance to the nearest object can be performed efficiently on multiple voxels in parallel and in real-time. As the application requires a fixed voxel size, the voxel size used in the developed compression scheme should be adjustable to match the size used in the application such that the voxel grids on both the

sensor end and the application overlap. The area to be covered by 3D sensors is an industrial robotic cell, approximately 10 m wide, 10 m long and 4 m high. This requires multiple 3D sensors distributed around the cell, focusing inwards.

The end application does not require the RGB data that is generated by the Kinect sensors, thus only the depth measurements and the corresponding point clouds are of interest. However, as mentioned in the introduction, a single computer has limited USB bandwidth, which makes it practically impossible to connect all sensors directly to the central computer. By introducing an embedded computer placed at the sensor location, preprocessing and compression can be done locally before transferring the data to the central computer over a Gigabit local area network, as illustrated in Figure D.1.

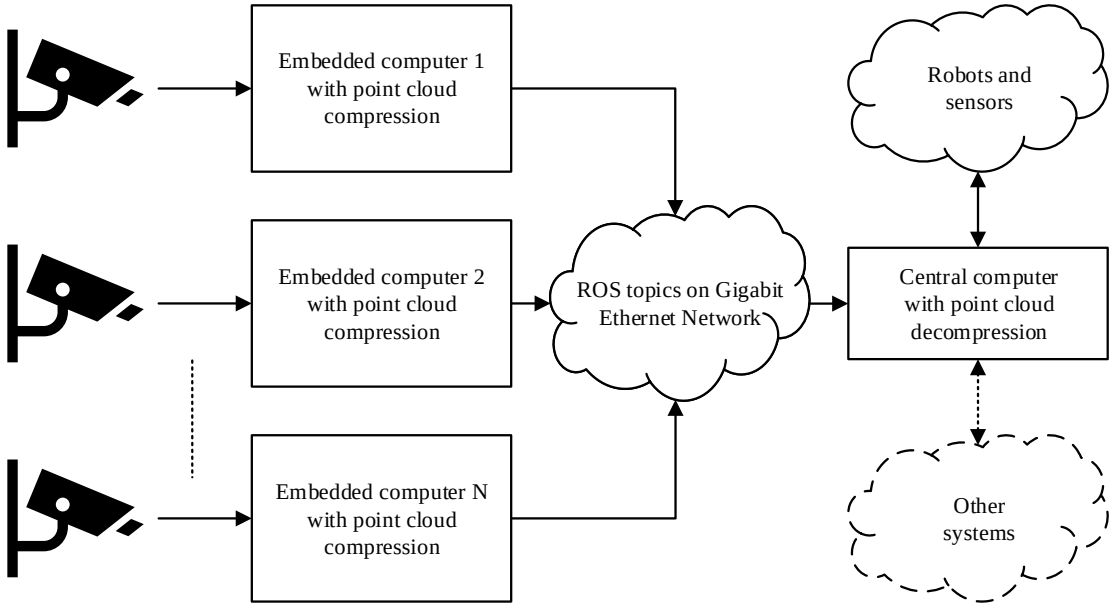


Figure D.1: Schematic overview of the considered sensor network. The number of sensors and embedded computers is scalable.

The software running on the embedded computer should be easy to deploy to an arbitrary number of sensor nodes from a remote location, i.e., it should not be necessary to compile or set up different versions of the software for nodes that use identical hardware. The system should also be scalable to such an extent that adding several more sensor nodes should not exhaust either the Gigabit Ethernet bandwidth or the CPU and GPU processing capabilities of the central computer. Thus, as much processing as possible should be performed locally at the sensor node, and efficient 3D point cloud processing and compression schemes are therefore needed at the embedded computer.

D.2.2 Point Cloud Preprocessing

The ROS node running on the sensor node is developed to process point clouds from different types of time-of-flight sensors. The software is designed to receive a "PointCloud2" ROS topic which can be generated by different ROS drivers depending on the sensor brand and type. Different point types can also be used, e.g., XYZ (coordinates only), XYZI (coordinates + intensity) and XYZRGB (coordinates + color). When a point cloud is received by the ROS node, it is first converted to an XYZI type cloud regardless of the input type. If the original point cloud already contains an intensity value, it is passed through as is (with some scaling), but, if not, a new intensity value is calculated. The generation of this intensity value will be described in detail in Section D.2.5. Any RGB color information is discarded.

To minimize the amount of data which is transmitted on the network between the embedded and the central computer, the captured point cloud is then transformed into a global coordinate system and cropped. The transformation matrix from the sensor's local coordinate system to the global coordinate system is known and published as an ROS topic by the central computer. By subscribing to this topic, the ROS node performs the transformation by using functions from the Point Cloud Library (PCL, [D14]). Even though the sensors are statically mounted, the transformation matrix is looked up every time a point cloud is received, to ensure that any changes due to updated sensor calibration are incorporated. If, for some reason, the matrix is not received, the latest known transformation is used.

When the transformation is complete, most of the unwanted sensor data (i.e., data which lie outside of the robotic cell) can be filtered out using a simple box crop filter. The resulting point cloud now only contains points in the global coordinate system which are of interest to the application on the central computer. As both the transformation matrices for all sensors and the robotic cell size are known, the same method can be applied on an arbitrary number of sensor nodes without any need for customization. Transforming all points to the global coordinate system before compression also ensures that the voxel grid of the octree used by the compressor will overlap with the grid of the GPU Voxels map used by the central computer.

D.2.3 Data Representation

The resulting point cloud format after processing on the embedded computer was preferred to be voxel- or grid-based, as the application running on the central computer uses this kind of representation. By building on the result found by [D2], the ROS node performs octree compression and encoding on the preprocessed point cloud stream before publishing the encoded stream to the network.

An octree is defined as a tree structure, where each node has zero or eight children.

A point cloud can be inserted into an octree structure by encapsulating the whole volume in a bounding box with equal side lengths. The bounding box is then divided into eight subvolumes (children) called voxels, and the process is repeated for each subvolume that contains points. A fixed octree resolution d_v can be selected such that the division process stops when the subvolumes reach a given size. The result is an octree, where all points of the cloud is encapsulated by the leaf nodes (the smallest voxels). Based on the resolution, each leaf node may encapsulate one or many points. The octree can be serialized to a binary sequence describing its structure, as shown in Figure D.2.

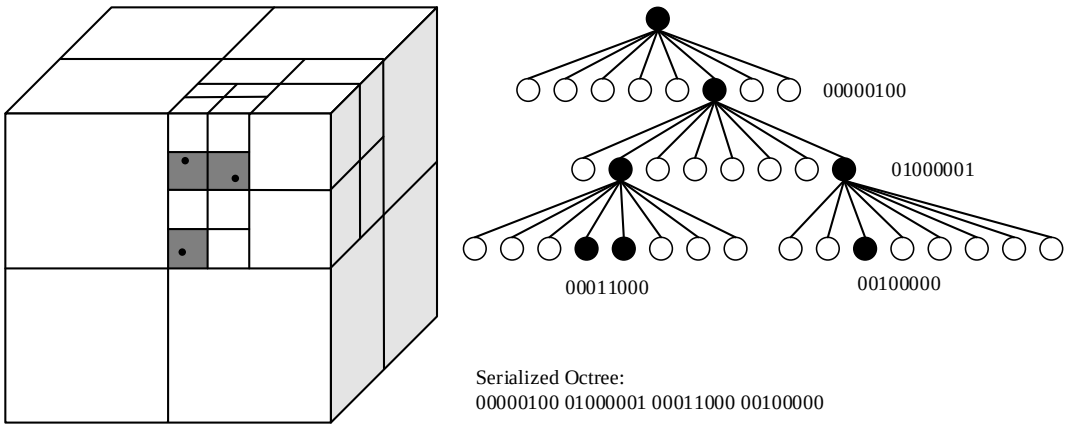


Figure D.2: Schematic overview of the octree data structure and its serialization. Nodes in the tree that encapsulate points are marked as occupied (binary 1). These are the only nodes that have children. On the finest level, the division process is stopped, and the nodes that encapsulate points are marked as occupied. Figure ©2012 IEEE. Reprinted, with permission, from [D2].

The Point Cloud Library supplies several types of octrees and octree leaf nodes, including base classes. One of them is the double-buffered octree with point detail encoding, as outlined in [D2]. To achieve the best possible compression ratios, the double-buffered octree was selected as the base octree type, and, by adapting the point detail encoding scheme, a new octree class was created. The encoding of the full point details was removed, and a new value of intensity was introduced. By creating a new leaf node type, each leaf in the octree is now capable of storing a floating point intensity value between 0 and 1. The creation of the intensity value will be described in Section D.2.5. When the point cloud is inserted into the octree, the intensity value from all XYZI points that fall within each leaf node are accumulated. Thus, each voxel in the octree gets a value of trust or intensity based on the intensity values of all points in the original point cloud that was covered by the voxel.

D.2.4 Compression

When the ROS node is started, an octree resolution d_v must be specified. The resolution of the octree is then kept constant as long as the process is running. This ensures that the tree depth is constant and that subsequent octrees can be compared using the XOR-method incorporated in the double-buffered octree. After the point cloud has been inserted into the octree, the tree structure is serialized to its binary sequence. For each leaf node, the new floating point intensity value is encoded as a separate 8-bit integer value, i.e., the floating point range 0–1 is converted into the integer range 0–255, which results in a maximum quantization error of ± 0.00195. Using a byte to represent a floating point value is done to minimize the size of the compressed point cloud and, as will be shown in Section D.2.5, the resolution is sufficient for the application at hand.

The entropy coder exploiting the double-buffered octree structure as well as the octree serialization function were used in their original and unmodified versions. An overview of the modified compression scheme is shown in Figure D.3. The point detail encoding implemented in the original method generated sets of integer x, y and z coordinates relative to the leaf node origin for each point encapsulated in the voxel. The precision of the encoded points was controlled by limiting the range of these integer values, e.g., when using an octree resolution of 9 mm, a range of $[0, 8]$ results in a precision of 1 mm. Limiting the range of the symbol set in this way allowed the point details to be efficiently encoded by the range encoder [D2].

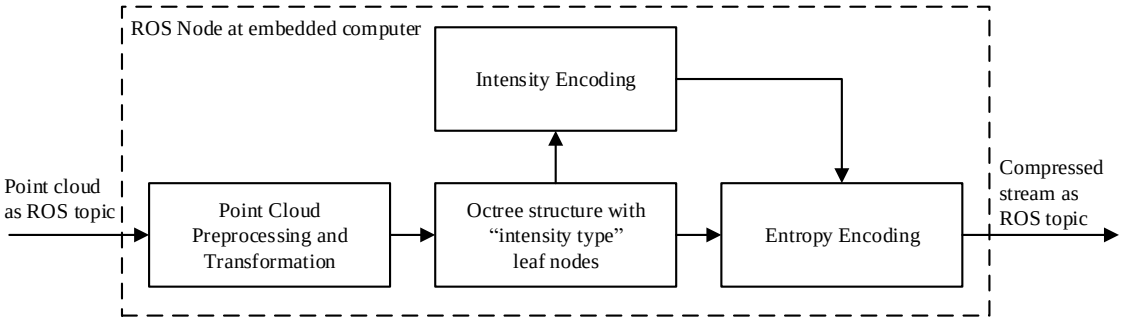


Figure D.3: Schematic overview of the compression principle implemented as an ROS node at the embedded computer.

In our work, the point detail encoding was replaced by the generated intensity value stored in a single byte with a range 0–255. The range is not further limited, but, in contrast to the original method, our method guarantees that there will only be one such number for each voxel. Depending on the octree resolution and the sensor resolution, there could possibly be a large number of points encapsulated by each voxel. The original method requires that, for each point, the point details must

be encoded, but when using an intensity value, only a single byte is needed. Both the serialized octree and the intensity values are finally passed to the range encoder, which results in a compressed byte stream that is published as an ROS topic.

D.2.5 Voxel Intensity Value Computation

Some versions of time-of-flight sensors natively provide an intensity value for each measured point. One example of such a sensor type is a lidar, where each measurement contains an intensity value based on the strength of the reflected signal. For other sensor types, e.g., RGB-D sensors such as the Kinect, such a value does not exist, and, when considering individual measurements, it is difficult to see how such a value could be constructed. However, when the point cloud has been converted to the "voxel domain" after being inserted into an octree, each voxel can contain several measurements, and that number of measurements can be seen as a measurement of trust in or intensity of the voxel.

The following paragraphs will describe different methods to construct the intensity value of a voxel, based on the measurements from a single Kinect sensor. One reason for introducing such a value is that, when using e.g., OctoMap or GPU Voxels in a later stage to fuse data from multiple sensors into a single map, the intensity values, if comparable and based on the same calculations, can be easily fused to create a level of trust for each voxel in the map. The intensity values can also be used to filter out voxels with weak measurements (typically noisy outlier points). Thus, the goal is to create an intensity value that makes sense in such scenarios.

D.2.5.1 Counting Points

When constructing a voxel intensity value, perhaps the most intuitive method is to simply count the number of points that reside in each voxel such that the intensity value becomes $I_v = n$, where v is the considered voxel and n is the number of points inside the voxel. This gives a value of strength directly based on the number of measurements, where each point is given a point intensity

$$I_p = 1. \tag{D.1}$$

While this is a simple solution, it introduces an issue when the intention is to filter out noise from RGB-D sensors such as the Kinect V2 due to the fact that the measurement points are not evenly distributed throughout the volume of interest.

D.2.5.2 Point Value Based on Quadratic Distance

The Kinect V2 has a field of view (FOV) of 70.6×60 degrees, and a resolution of 512×424 pixels for the generated depth map (a total of 217,088 pixels). As the octree resolution is constant, using the counting method to create voxel intensities means that voxels closer to the sensor will contain a much higher number of measurements than voxels farther away, when measuring the same object at different distances. Even a few noisy points (or false measurements) close to the sensor is enough to generate a much higher voxel intensity than accurate measurements farther away. Consider, for example, rain drops when using the sensors outdoors. The drops are evenly distributed in the entire volume but will trigger much higher intensity values close to the sensor, as they are "hit" by more measurements. To compensate for this behavior, a new method for generating intensity values is suggested.

Let x and y be the side lengths of the rectangle that is generated by intersecting the FOV with a plane perpendicular to the Z-axis (the depth axis) of the sensor (see Figure D.4). Using the FOV angles, the lengths at a given depth z can be calculated as

$$x(z) = 2 \cdot z \cdot \tan\left(\frac{35.3 \cdot \pi}{180}\right), \qquad y(z) = 2 \cdot z \cdot \tan\left(\frac{30 \cdot \pi}{180}\right). \tag{D.2}$$

Now, let $d_x(z)$ and $d_y(z)$ be the side length of each pixel in the depth map projected onto the plane. The area of each pixel for the Kinect V2 can be written as

$$A_p(z) = d_x(z) \cdot d_y(z) = \frac{x(z)}{512} \cdot \frac{y(z)}{424}. \tag{D.3}$$

Let d_v be the octree resolution, i.e., the side length of the cube forming the voxel, and let d_v^2 be the area of the voxel when projected onto the plane. The maximum number of measurements (pixels) that can be encapsulated by a single voxel at a given distance can then be found by

$$N_p(z) = \frac{d_v^2}{A_p(z)} = \frac{d_v^2}{z^2 \cdot c}, \tag{D.4}$$

where c is constant for a given sensor type. It should be noted that c will be different depending on the sensor type and its parameters, i.e., the FOV and the resolution. Changing any of these values will impact the calculation of the point intensity value I_p. A finer resolution (more pixels in the same FOV), will result in a lower I_p, as more points will fit into a voxel. A larger FOV with the same resolution will have the opposite effect. In addition, note that $N_p(z)$ is an approximation: The voxels have a fixed placement in the global coordinate system and, depending on the transformation between the sensor and the global coordinate system and the distance z, d_v may not accurately describe the area of the voxel when projected onto

the plane, since the voxel may be rotated or offset relative to the plane. However, since the calculation of the point intensity value is performed before the points are transformed, this approximation is not practically avoidable. The point intensity value P_v for a given measurement point can now be defined as

$$I_p(z) = \frac{1}{N_p(z)} \tag{D.5}$$

such that each point is given an intensity value inversely proportional to the maximum number of points that could fit inside the voxel. This suggested method for calculating the voxel intensity place low value in measurements close to the sensor, but, when looking at it from the voxel perspective, all voxels will have comparable intensities, and as the experimental results will show, when using the generated value to filter out "weak" voxels, the ones that are removed will be distributed over the entire volume, as opposed to the counting method where voxels closer to the sensor are prioritized, and accurate measurements far from the sensor are removed more quickly.

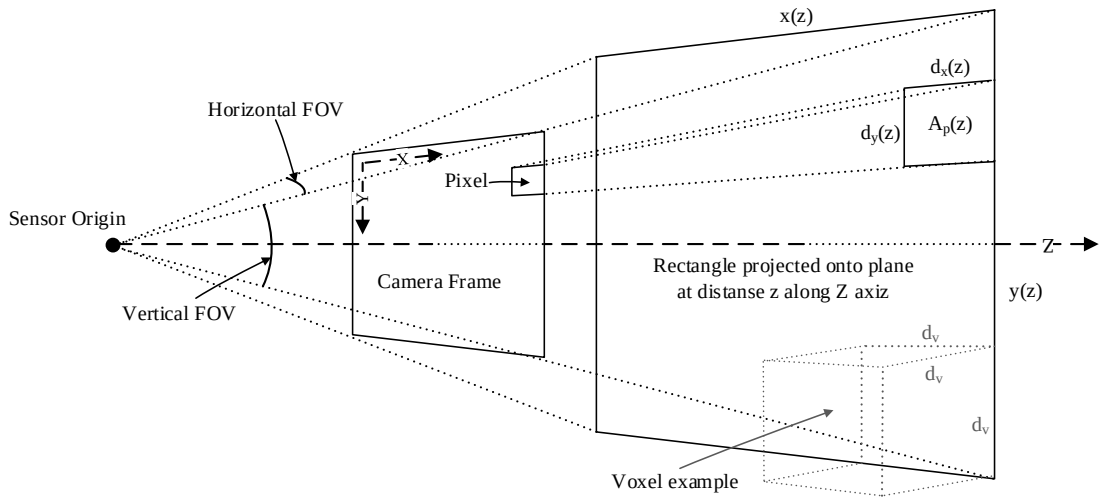


Figure D.4: Schematic view of a sensor's field of view when projected onto a plane perpendicular to its direction of view.

D.2.5.3 Point Value Based on Linear Distance

A third method for generating the voxel intensity is to use a slightly less aggressive approach when lowering the point intensities close to the sensor, so that voxels with multiple measurements are awarded a higher intensity value. To achieve this, the cutoff distance z_c at which it is impossible to get more than one measurement inside

a voxel is calculated, i.e., where $N_p(z) = 1$:

$$z_c = \sqrt{\frac{d_v^2}{c}}. \tag{D.6}$$

When inserting an octree resolution d_v of e.g., 0.04 m, this distance is 14.57 m. The point intensity value is then defined as simply

$$I_p(z) = \frac{z}{z_c}. \tag{D.7}$$

Figure D.5 shows $I_p(z)$ for the three different methods outlined above for the Kinect V2 depth sensor. The values are capped at 1.0 after the cutoff distance z_c, thus preventing single point intensities from exceeding the 0–1 range.

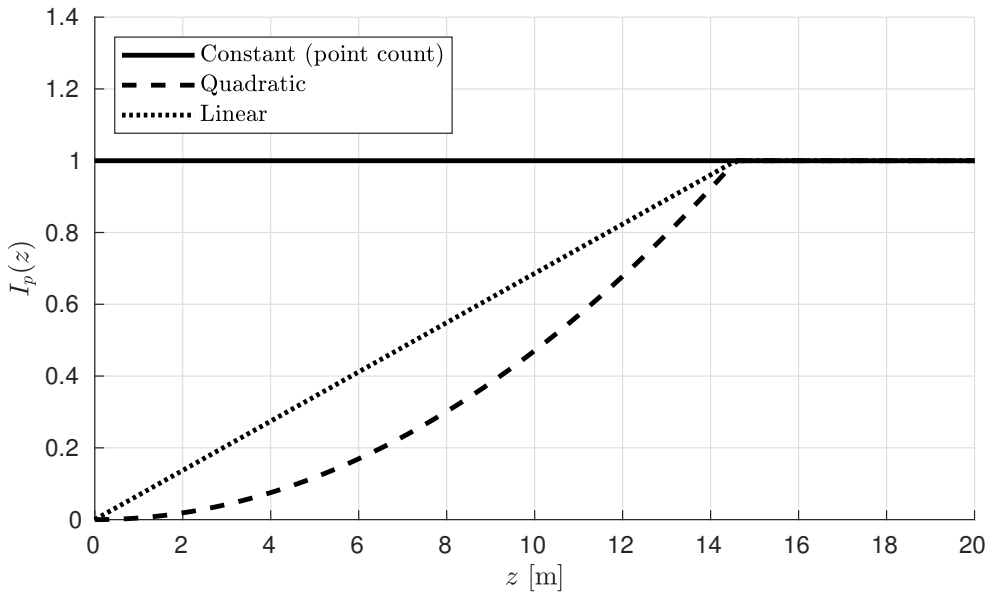


Figure D.5: Plot of the point intensities $I_p(z)$ for the three discussed calculation methods.

D.2.5.4 Voxel Intensity

In the previous paragraphs, three methods for generating point intensity were presented. The final step is to generate an intensity value for the voxel that is encapsulating the points. The voxel intensity is defined as follows:

$$I_v = \sum_{i=1}^{n} I_p(i), \tag{D.8}$$

where n is the number of points inside the voxel. When using I_p from Equation (D.1), the generated value is an integer value corresponding to the number of points. Using Equation (D.5), I_v will accumulate towards 1 when the voxel is filled up

with points. However, as $N_p(z)$ is an approximation, I_v may accumulate to slightly more than 1, hence I_v is capped in the software so that it will never exceed 1 when using Equation (D.5). The voxel intensity generated based on Equation (D.7) may accumulate to more than 1 if there are multiple measurements inside the voxel. This must be taken into account during the octree serialization process, where the intensity value is converted to an 8-bit integer value by the following method:

$$I_v^{int} = \begin{cases} I_v \cdot 255 & \text{for } I_v \leq 1, \\ 255 & \text{for } I_v > 1. \end{cases} \tag{D.9}$$

As described in Section D.2.4, the maximum quantization error when converting a floating point range 0–1 into an integer range 0–255 is ±0.00195. If only a single point is registered in a voxel, the point intensity value I_p must be larger than the quantization error for it not to be set to zero. For I_p according to Equation (D.1), this is not an issue as the value is already an integer and does not need to be converted. This means that, if there are more than 255 points in a voxel, the intensity value will be capped at 255 and not reflect the real point count. However, this can only happen when the Kinect V2 measures objects closer than approximately 1 m when using a 4 cm octree resolution, which will never happen in our scenario.

The worst case scenario happens when using Equation (D.5), where point intensities close to the sensor are very small. However, it can be shown that, when using an octree resolution of 0.04 m, which is the largest resolution used in the experiments in this paper, the smallest possible point intensity is 0.0047 at a distance 1.0 m from the sensor, which is more than double the quantization error.

The methods used in the examples in this section are only valid for the Kinect V2 or similar sensors. Other methods for generating the voxel intensity may be designed for different sensor types or depending on the application. For example, for the Velodyne PUCK lidar connected to one of the sensor nodes, the intensity value already exists and does not need to be calculated.

D.2.6 Decompression and Denoising

The ROS node which runs on the central computer subscribes to, decodes and decompresses the encoded stream into reconstructed point clouds as shown in Figure D.6. After the compressed stream has passed through the entropy decoder, the output point cloud is generated by combining the deserialized octree structure and the voxel intensity values. As mentioned in Section D.2.2, the octree resolution should be selected to match the requirement of the end application. In our scenario, the application is intended to be based on the GPU Voxels library. By matching the size of the leaf nodes in the octree used by the compressor with the resolution of the GPU

Voxels map, the reconstructed point cloud can be inserted with a one-to-one ratio. To achieve this, the center coordinate of the leaf voxels needs to be reconstructed by the decompressor.

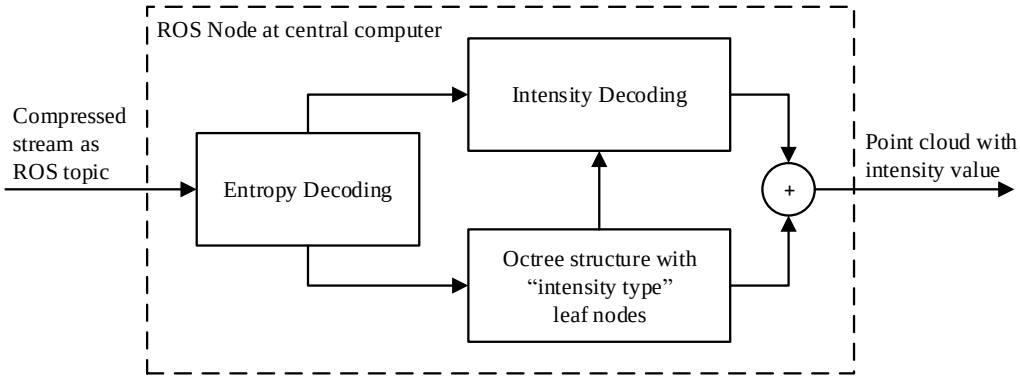


Figure D.6: Schematic overview of the decompression principle implemented as an ROS node at the central computer.

For each leaf node in the received octree, a point is generated from the voxel center coordinate and the intensity value of the voxel. The result is a new point cloud including coordinates with a precision equal to the octree resolution, similar to a voxel grid filter. Every leaf node of the decompressed octree thus results in an XYZI point in the new point cloud, including the voxel intensity value in the range 0–255 (see Figure D.7).

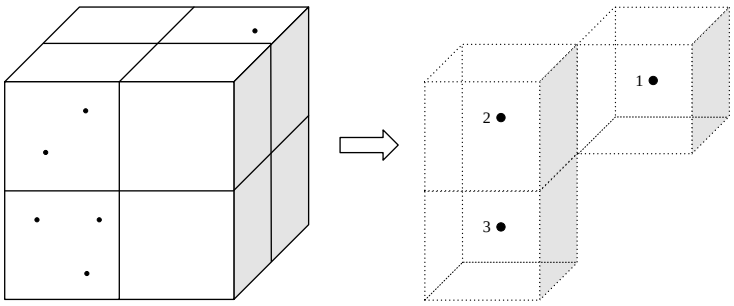


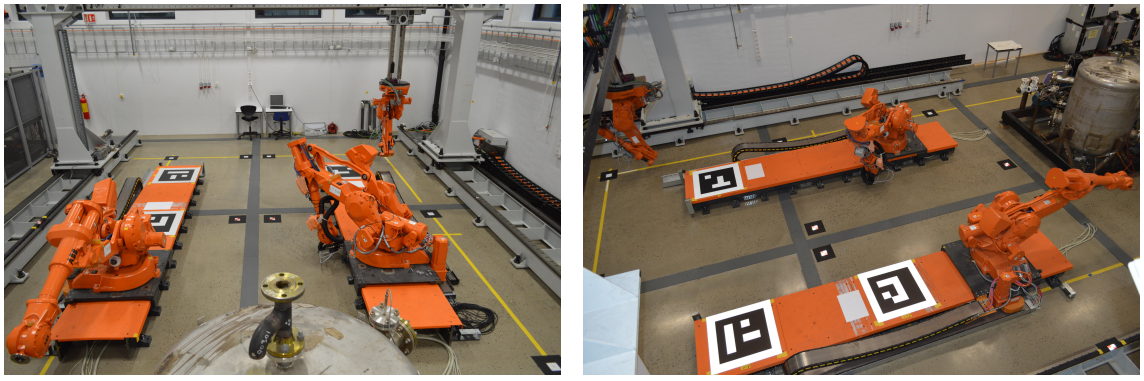
Figure D.7: (**Left**) The points of the original point cloud inserted into an octree structure. The figure shows only a subset of the octree consisting of eight leaf nodes, where three of the voxels are occupied. (**Right**) After decompression, the reconstructed point cloud contains points at the center coordinate of the occupied voxels, in addition to the intensity value. In this example, the intensity value is generated by point counting. The dashed voxel cubes on the right-hand side are only shown for reference.

After decompression, the generated intensity value can be further exploited. By using a pass-through filter, the point cloud can be filtered such that points with an intensity below a given value are removed.

D.2.7 Experimental Setup

The point cloud processing and compression scheme described in this paper is intended for use in a large scale industrial location. The system was therefore developed and tested in an indoor robotic cell consisting of two rail-mounted ABB IRB4400 robots (ABB Ltd., Zurich, Switzerland) and one gantry-mounted ABB IRB2400 robot (see Figure D.8). In addition to the robots, a processing facility was placed in the cell to introduce a more realistic environment.

The area to be covered by 3D sensors is approximately 10 m wide, 10 m long and 4 m high. To accomplish this, six Kinect V2 sensors were mounted at different locations along the walls at a height of around 4.2 m. In addition, a single Velodyne PUCK VLP-16 lidar (Velodyne LIDAR, San Jose, CA, USA) was mounted in one of the corner locations.



(a) **(b)**

Figure D.8: (**a**) overview of the robotic cell which is covered by the embedded sensor nodes; (**b**) the cell seen from a different angle, close to one of the sensor nodes.

The ROS nodes for preprocessing and compression were deployed on six NVIDIA Jetson TX2 Development Boards, each connected to their own Kinect V2 depth sensor. one node is also connected to a Velodyne VLP-16 PUCK lidar and, for future use, a Carnegie Robotics Multisense S21 stereo camera (Carnegie Robotics LLC, Pittsburgh, PA, USA), as seen in Figure D.9. The Jetson TX2 contains a Quad-core Arm A57 CPU (Arm Limited, Cambridge, UK), an NVIDIA Pascal GPGPU and 8 GB LPDDR4 memory. The ROS node for decompression and denoising was deployed on the central computer, which was equipped with an Intel Core i5-2500 CPU (Intel Corporation, Santa Clara, CA, USA) and 8 GB of system memory.

By equipping sensor nodes with different sensor technology and enclosing the electronics in a waterproof cabinet, the goal is to use the sensors in an outdoor environment, where measurements from the different sensors can compliment each other in different weather and lighting conditions. In fact, the sensor nodes have already been tested outdoors in an industrial area, and evaluating the results from

these tests is part of our future work.

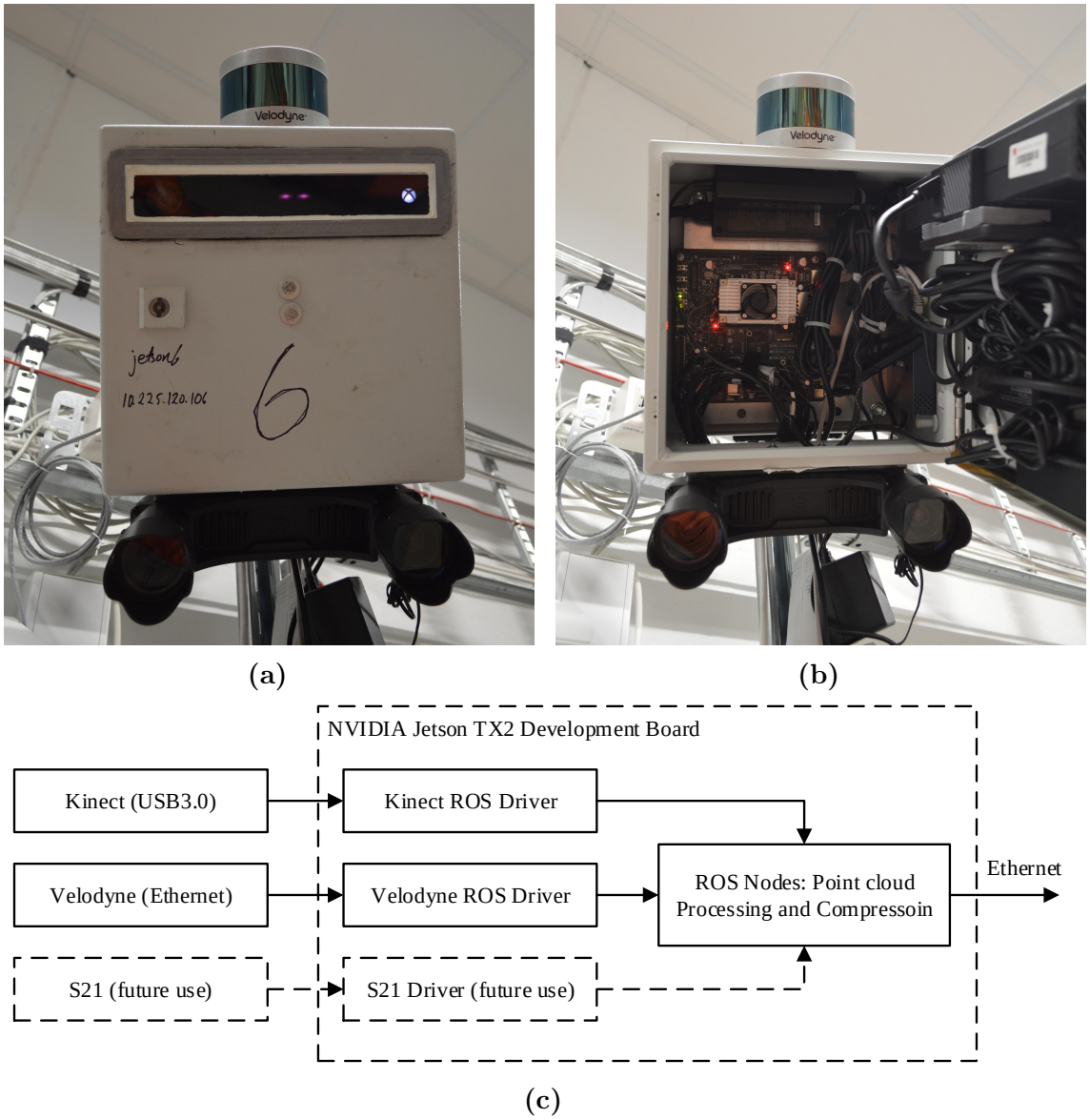


Figure D.9: (**a**,**b**): Embedded sensor node consisting of an NVIDIA Jetson TX2 Development Board, a Kinect V2 depth sensor, a Velodyne VLP-16 PUCK Lidar and a Carnegie Robotics S21 stereo camera for future experiments; (**c**) schematic overview of the NVIDIA Jetson TX2 hardware connections and software modules. The points from the Velodyne lidar and the S21 stereo camera are not part of the experimental results.

To generate the point cloud stream which is input to the compressor, an ROS driver for the Kinect V2 (IAI Kinect2, [D15]) was installed. This driver generates depth and color images at a rate of 30 frames per second, which in turn are converted and published as ROS PointCloud2 messages. The driver uses the NVIDIA GPU for depth image processing. Figure D.9c shows a block diagram of the hardware and software used in the experimental setup.

In the experimental results presented in Sections D.3.2 and D.3.3, the compression

and decompression processes from a single sensor node to the central computer was considered. Only the Kinect V2 data was processed, and the point counting method for generating voxel intensities was used.

The transformation from the sensor's coordinate system to the global coordinate system was manually measured and has an unknown accuracy. However, the transformation only affects which points are removed by the crop-filter, and thus does not directly affect the compression performance. All experiments were conducted using live point cloud streams.

D.2.8 Multisensor Setup

For the denoising results presented in Section D.3.4, all six sensor nodes were used. The sensors were placed along the outer peripheral of the lab and manually calibrated based on the method in [D16]. Calibration of the camera intrinsic parameters was performed according to [D17]. Note that, compared to the setup in [D16], some of the sensors' mounting locations have been moved such that the monitored area is smaller. Work has also been done to create an automatic calibration scheme for the same industrial lab. These results have been submitted and are currently under review in [D18]. Table D.1 shows the locations of the sensors in the global coordinate system.

Table D.1: Calibrated sensor positions in meters and orientations in degrees. The positions are in the global coordinate system. N corresponds to the sensor number.

N	**X**	**Y**	**Z**	**RotZ**	**RotY**	**RotX**
1	7.798	0.496	4.175	40.170	0.639	−136.170
2	1.729	0.501	4.135	−45.253	2.063	−141.490
3	9.522	5.275	4.353	88.439	−0.072	−143.461
4	0.553	4.995	4.353	−89.208	−0.495	−145.591
5	8.879	9.192	4.194	126.733	−1.311	−139.272
6	0.559	9.136	4.145	−121.458	−0.487	−137.562

Mounting several Kinect sensors in the same environment could possibly lead to interference problems. Indeed, the first version of the Kinect is known to have problems with interference, as they are based on structured light [D19]. The latest version that we used in our experiments (Kinect for XBOX One) is based on time-of-flight measurements using a modulated continuous wave IR signal. There is still a possibility of interference when using this version of the sensor, but it has been shown that the errors caused by interference are negligible when certain mounting constellations are avoided [D20]. Even though some of the sensor orientations in our

lab may fall within this "bad" configuration, we have not had any visible problems with interference in our experiments, and thus no further steps have been taken to manage this potential issue.

D.3 Results

The resolution of the Kinect V2 IR depth image is 512×424 pixels, thus the number of points in each point cloud generated by the IAI Kinect2 ROS driver is 217,088. When discarding the RGB color information which is not used, each point consists of 4 32 bit floating point numbers, or 16 bytes. (The PCL point type used includes x, y, z coordinates and four bytes of padding). These numbers are constant for all point clouds and give an original point cloud size of 3392 KiB.

D.3.1 Preprocessing

The first step in preprocessing was to generate the point intensity values for all points. In this experiment, Equation (D.1) was used. In the process, the points were converted from XYZ to XYZI (this does not increase the point cloud size due to the four bytes of padding in the XYZ type point). Before the point cloud was sent through the octree compressor, it was transformed and crop-filtered as previously described. This yielded a new point cloud with an average of 37,108 points when measuring 1000 consecutive point cloud frames, which was then sent through the compressor at the sensor node. Table D.2 shows the experimental results from this part of the process. By cropping the point cloud, the size was reduced by an average factor of 5.85.

Table D.2: Point cloud filtering and cropping results. Cropped results are mean values over 1000 measured point clouds.

Measurement	**Original**	**Cropped**
Number of Points	217,088	37,108 ± 293
Size (KiB)	3392	579.8 ± 4.6
Ratio	1:1	1:5.85 ± 0.05

D.3.2 Compression

After preprocessing, the point cloud was processed by the compressor. When inserting them into an octree structure with a 4 cm octree resolution, the same 1000 point clouds resulted in octrees with an average of 17,771 octree leaf nodes each. This

means that, on average, 2.09 points were inserted into each voxel. The compression ratios and sizes were logged by the compressor software. The number of bytes in the transferred encoded stream was counted each frame to measure the exact size of each compressed point cloud. Table D.3 shows the experimental results from the compression process. A compression ratio (based on size) of 40.5 was achieved.

Table D.3: Octree Compression Results, 4 cm octree resolution. The results are mean values over 1000 measured point clouds, with the number of points rounded to the nearest integer.

Measurement	**Cropped**	**Compressed**
Number of Points	37,108 ± 293	17,771 ± 118
Size (KiB)	579.8 ± 4.6	14.31 ± 0.20
Bytes per Point	16	0.82 ± 0.01
Compression Ratio	1:1	1:40.5 ± 0.5

Another experiment was done with an octree resolution of 2 cm. This was performed on another point cloud stream, hence the size of the cropped point cloud is slightly different. The results from this experiment can be seen in Table D.4. A compression ratio of 22.5 was obtained using 2 cm octree resolution.

Table D.4: Octree compression results, 2 cm octree resolution. The results are mean values over 1000 measured point clouds, with the number of points rounded to the nearest integer.

Measurement	**Cropped**	**Compressed**
Number of Points	37,574 ± 313	33,321 ± 243
Size (KiB)	587.1 ± 4.9	26.08 ± 0.17
Bytes per Point	16	0.80 ± 0.01
Compression Ratio	1:1	1:22.5 ± 0.1

The number of bytes per point in both experiments was reduced by a factor of 20 from 16 to 0.82 and 0.8, respectively. These results indicate that the encoding process presented by [D2] is providing similar performance in both experiments. The larger compressed size and lower compression ratio in the second case are due to a higher number of octree leaf nodes as a result of a more fine-grained resolution.

The IAI Kinect2 driver can deliver compressed depth images, which can limit the bandwidth required to transfer depth data. The depth images could be transferred to the central computer before being converted to point clouds. Measuring 1000 depth images from the same sensor node yields an average depth image size of 0.43 MB, and a compressed image size of 0.22 MB when using the default JPEG compression.

Compared to transferring the depth data in a raw point cloud format, using the compressed depth image would yield better results (the cropped, uncompressed point cloud has a size of approximately 0.58 MB as seen in Table D.3). However, the octree compression method far outperforms the depth image compression, yielding compressed clouds of only 14.31 KiB and 26.06 KiB when using an octree resolution of 4 and 2 cm, respectively.

D.3.3 Frequency and Bandwidth

Table D.5 shows the results obtained by logging the application and system performance. The frequency of the transferred point clouds was measured using the built-in ROS topic monitor, and the cycle time of the software was measured using the `rostime` C++ library. CPU load and memory usage were not formally measured but estimated based on the Linux process monitoring utility `htop`.

The maximum frequency and bandwidth of the compressor were measured while compressing point clouds using a 4 cm octree resolution. While the IAI Kinect2 ROS driver is able to deliver point clouds at a rate of 30 per second, the compression process is currently limited by running on a single CPU core. The developed ROS node was able to process, compress and transfer point clouds at an average rate of 26.9 Hz measured over a window of 10,000 frames. This results in an average bandwidth of 384.9 KiB/s, based on the compressed size in Table D.3.

Table D.5: Compression and decompression performance when using 4 cm octree resolution.

Measurement	**Compression (@ max. FPS [1])**	**Compression (@ 20 Hz)**	**Decompression (@ 20 Hz)**
FPS	26.9 Hz	20.0 Hz	20.0 Hz
Bandwidth	384.9 KiB/s	286.2 KiB/s	286.2 KiB/s
Cycle Time	(36 ± 4) ms	(36 ± 4) ms	(13 ± 2) ms
CPU load	100 %	80 %	14 %
Memory	63.5 MiB	60 MiB	35.7 MiB

[1] Frames per second.

By logging the processing time for the ROS node function responsible for the preprocessing and compression processes, an average time of 36 ms per frame was measured over 1000 frames. This corresponds to a maximum possible frame rate of 27.8 Hz, which is in accordance with the above results when taking into account that there is some additional overhead caused by the underlying ROS system and that the process used approximately 100% of a single CPU core.

When limiting the IAI Kinect2 driver to output point clouds at 20.0 Hz, the developed ROS node is able to process and compress all incoming point clouds, resulting in a bandwidth of 286.2 KiB/s. At this configuration, the ROS node utilizes around 80% of the processing capability of one CPU core on the Jetson TX2. In addition, the node used approximately 60 MiB of memory.

On the central computer, the average amount of time used to decompress a single frame, based on 1000 measurements, was 13 ms, as seen in Table D.5. The process utilized around 14% of a single CPU core and 35.7 MiB of memory. As a final test, point clouds from all six sensor nodes were decompressed simultaneously. This utilized around 65% of the CPU core and 51.3 MiB of memory, which indicates that there is some overhead in the ROS software and that the actual resources needed for decompression are less than the results presented in Table D.5.

Figure D.10 shows a visual comparison between uncompressed and compressed point clouds. Figure D.10a,c show the cropped point cloud with RGB color, where Figure D.10c is accumulated over one second for better visibility. Figure D.10b,d show the same region of interest after compression and decompression, using a 4 cm octree resolution. The decompressed clouds were filtered using the calculated intensity value, such that points with an intensity value lower than 2 were removed. To highlight the underlying octree structure, the points are displayed using cubes.

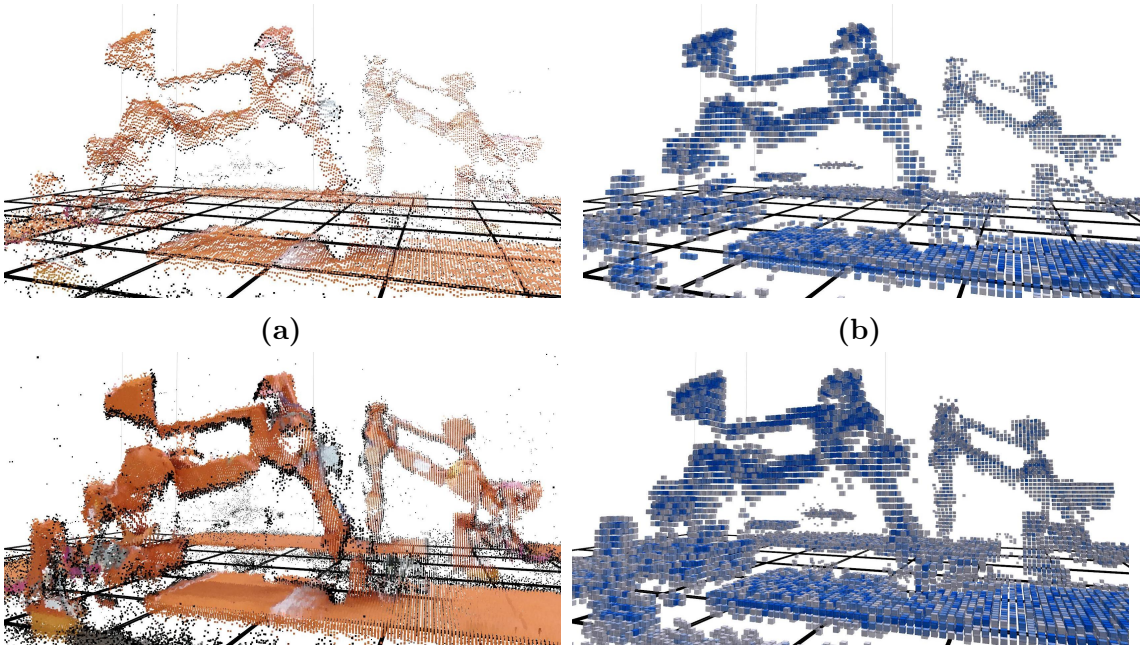


Figure D.10: Visual comparison of original and compressed point clouds using a 4 cm octree resolution. (**a**,**c**): colored point cloud generated by the Kinect V2 sensor. (**b**,**d**): point cloud generated by the decompressor at the central computer. For better visibility of the underlying octree structure, the points are shown as cubes with 3 cm sides. A stronger color indicates higher intensity.

D.3.4 Denoising

When the compressed point cloud has been reconstructed by the decompressor, the new intensity value can be used to filter the points, as suggested in Section D.2.5. To test the feasibility of this suggestion, point clouds from all six sensor nodes, including the Velodyne lidar, were compressed, transmitted, reconstructed and filtered using different intensity values. Any formal verification of the results has not been performed, but the results were visually inspected using a visualizer application from ROS. Figure D.11 presents the visualized results, when using the different point intensity generation algorithms and filtering using different values. A stronger color corresponds to a higher intensity. Points with an intensity value lower than the filter value are removed. As seen in Figure D.11a,c, there is a relatively large amount of noise in the point clouds, especially close to the sensor origins. In the filtered clouds, the floor and points outside the cell have been removed by the box crop filter.

Filtering the points on the "point count" intensity (Figure D.11d) does a decent job of cleaning up the cloud. However, as can be seen from the picture, some noisy points close to the sensors are still present, and points further away are aggressively removed. This is best seen from the yellow points in the bottom left corner, which originates from the sensor placed in the top right corner (the origin of the sensor lies just outside of the image, refer to Figure D.11b for the sensor locations).

Using the "linear" intensity value, all noisy points close to the sensor origin are removed, and more of the measurements further away are preserved, as seen in Figure D.11e. In the authors' opinion, this is the point cloud that best represents the measured environment, with the least amount of noise and strong measurements of the actual objects.

Figure D.11f shows the points filtered using the "quadratic" point intensity value. Here, it is seen that points closer to the sensor (e.g., on the blue column on the left) are given a relatively lower intensity value, and points farther from the sensor (e.g., the yellow points in the lower bottom corner and close to the floor) are given a relatively higher intensity value. The effect is, as expected, that the points removed by the filter are evenly distributed throughout the volume, and not based on the distance to the sensor.

In all the filtered clouds, almost all points generated by the Velodyne lidar (best seen as the olive-colored points on the right wall) are conserved, due to the fact that the intensity is generated by the sensor itself and not calculated, and that these values are higher than the filter values. This behavior is intended, as the Velodyne measurements are much more stable and reliable than the Kinect measurements, and thus should not be filtered out as noise. In the examples, the filter values were picked by trial and error, and different values would show different results (i.e., more

points removed by using a higher filter value). The values should therefore be tuned to the application requirements and the user's needs.

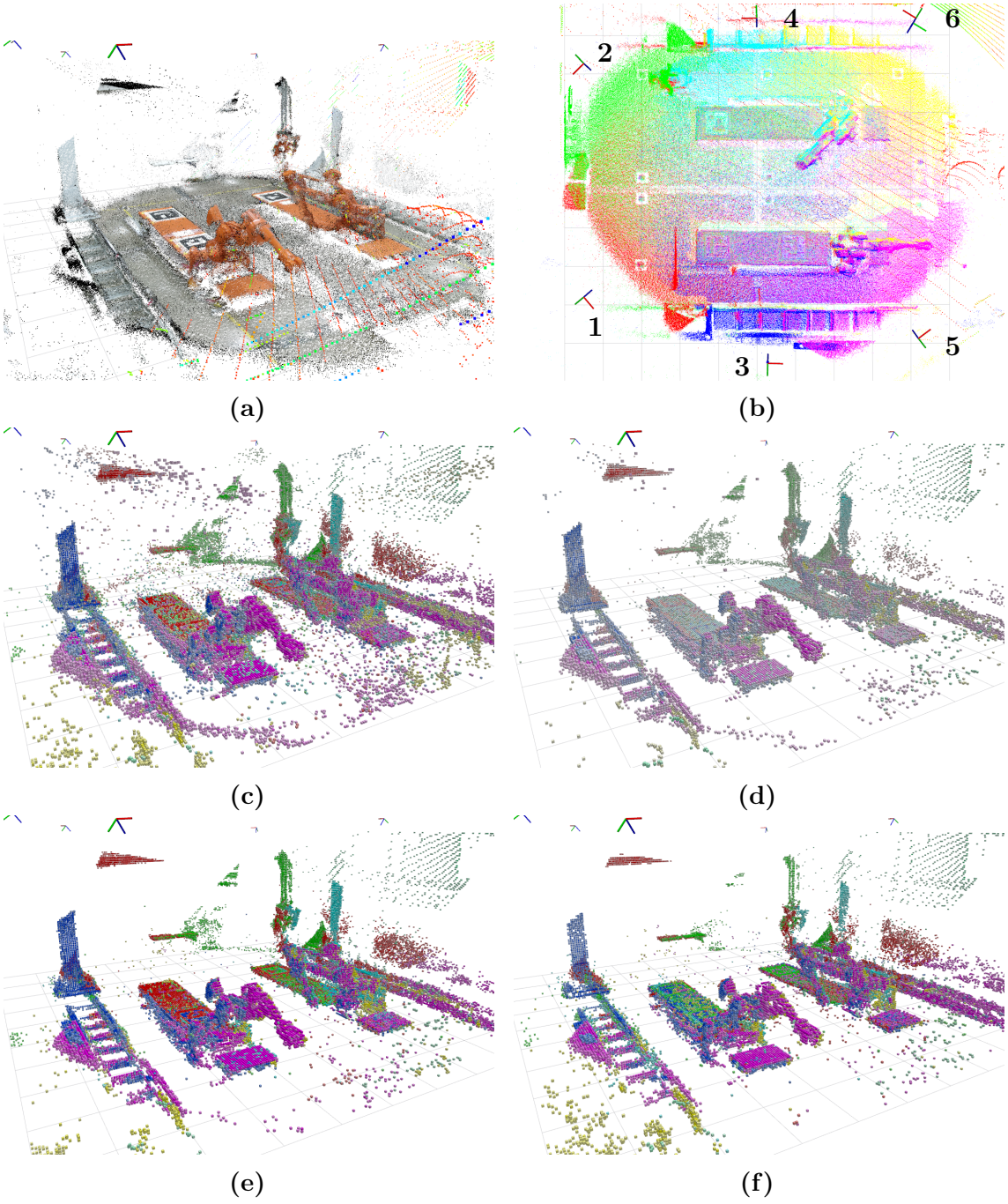


Figure D.11: (**a**) original point clouds from all sensors, including color information for the Kinect sensors; (**b**) top down view of original point clouds, where the different sensors' points are color coded; (**c**) reconstructed, unfiltered point cloud; (**d**) cloud with "point count" voxel intensities, filter value 2; (**e**) cloud with "linear" intensities, filter value 64; (**f**) cloud with "quadratic" intensities, filter level 64.

D.4 Discussion

In this paper, a scalable solution for 3D sensing in a large volume consisting of multiple sensors was presented. A constant frame rate of 20 Hz was achieved on the sensor nodes containing an embedded processing unit. The bandwidth requirement on each local node was 286.2 KiB/s, which means that the proposed solution could be scaled up to an order of 440 sensors (assuming that the central node has a dedicated 1 Gbit/s network input and "unlimited" processing power). Compared to transferring the cropped, but uncompressed point cloud, which would result in a bandwidth of 11 596 KiB/s (579.8 KiB $\times$ 20 Hz), the compression leaves room for significantly more point clouds streams on the network. It was also shown that the octree compression outperforms the default depth image compression performed by the IAI Kinect2 driver.

In [D2], the number of bytes per point (BPP) when using an octree resolution of 9 mm and a fixed point precision of 9 mm was 0.30. When using a point precision of 1 mm, the BPP was 0.87, which is closer to our results. The lower values achieved in [D2] is most likely due to the fact that the range of the symbol set used to encode the point details is greatly limited (the range $[0, 8]$ was used), which makes the range encoder more efficient. The results also show that, while in the original method the BPP increases with finer precision, in our method, the BPP is practically unaffected when the octree resolution is changed.

It should also be noted that since point details for all points are encoded in the original method, the total compressed size of the point cloud would be larger. Even with a BPP of 0.3, the resulting compressed cloud would have a size of 63.6 KiB, corresponding to a bandwidth of 1272 KiB/s. Thus, when comparing the results, one should also consider the application at hand. In our scenario, it was more important to lower the size of the compressed point cloud and to generate the intensity values which are to be used by the end application than to encode point details for all points in the original point cloud.

In [D4], a frame rate of 5.86 Hz was achieved with a powerful CPU (Intel i7), while 20 Hz was achieved in this paper using only a much less powerful ARM processor. The reason for the improved performance achieved in our work results from (1) a dedicated local network with no background transmission and (2) all the data points inside one voxel are in our work filtered and described by only one coordinate and one intensity value.

Different algorithms for generating voxel intensity values based on measurements where no such value exists were proposed. The experiments showed promising results when using the generated values to filter out false or noisy measurements. By adapting the algorithms, this method can be used to generate intensity values for

different sensor types, not only RGB-D sensors as described in this paper.

Future work includes evaluating the benefit of filtering based on intensity values on data from outdoor testing (e.g., removing noise from rain and other unwanted reflections). In the future, measurements from the Carnegie Robotics stereo camera will also be included. In addition, an effort will be made to optimize the software. More specifically, the possibility of limiting the symbol range used when encoding the intensity value should be explored, as this could make the range compressor more efficient. Parts of the process can also be parallelized in order to utilize multiple CPU or even GPU cores on both the Jetson TX2 module and the central computer. This could make it possible to compress streams at higher frame rates and reduce latency.

D.5 Acknowledgments

The research presented in this paper has received funding from the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896.

References – Paper D

[D1] J. Dybedal and G. Hovland. Optimal placement of 3D sensors considering range and field of view. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1588–1593, July 2017.

[D2] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach. Real-time compression of point cloud streams. In *2012 IEEE International Conference on Robotics and Automation*, pages 778–785, May 2012.

[D3] G Nigel N Martin. Range encoding: an algorithm for removing redundancy from a digitised message. *Video and Data Recording Conference*, pages 24–27, March 1979.

[D4] C. Moreno, Yilin Chen, and M. Li. A dynamic compression technique for streaming kinect-based Point Cloud data. In *2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 550–555, Jan 2017.

[D5] Siheng Chen, Dong Tian, Chen Feng, Anthony Vetro, and Jelena Kovačević. Fast Resampling of Three-Dimensional Point Clouds via Graphs. *IEEE Transactions on Signal Processingl*, 66(3):666–681, 2018.

[D6] Dorina Thanou, Philip A. Chou, and Pascal Frossard. Graph-based compression of dynamic 3D point cloud sequences. *IEEE Transactions on Image Processing*, 25(4):1765–1778, 2016.

[D7] Y. Schoenenberger, J. Paratte, and P. Vandergheynst. Graph-based denoising for time-varying point clouds. *arXiv e-prints*, nov 2015.

[D8] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot*, 34:189–206, 2013.

[D9] Knut B. Kaldestad, Geir Hovland, and David A. Anisi. 3D Sensor-Based Obstacle Detection Comparing Octrees and Point clouds Using CUDA. *Modeling, Identification and Control*, 33(4):123–130, 2012.

[D10] S. Ueki, T. Mouri, and H. Kawasaki. Collision avoidance method for hand-arm robot using both structural model and 3D point cloud. In *2015 IEEE/SICE International Symposium on System Integration (SII)*, pages 193–198, Dec 2015.

[D11] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[D12] Joacim Dybedal. SFI-Mechatronics/wp3_compressor and SFI-Mechatronics/wp3_decompressor: First Release. https://doi.org/10.5281/zenodo.2554855, February 2019.

[D13] A. Hermann, F. Drews, J. Bauer, S. Klemm, A. Roennau, and R. Dillmann. Unified GPU voxel collision detection for mobile manipulation planning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4154–4160, Sept 2014.

[D14] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[D15] Thiemo Wiedemeyer. IAI Kinect2. https://github.com/code-iai/iai_kinect2, 2014 – 2015. [Online; accessed Jan. 23, 2018].

[D16] Atle Aalerud, Joacim Dybedal, Erind Ujkani, and Geir Hovland. Industrial Environment Mapping Using Distributed Static 3D Sensor Nodes. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6. IEEE, jul 2018.

[D17] Erind Ujkani, Joacim Dybedal, Atle Aalerud, Knut Berg Kaldestad, and Geir Hovland. Visual Marker Guided Point Cloud Registration in a Large Multi-Sensor Industrial Robot Cell. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6. IEEE, jul 2018.

[D18] Atle Aalerud, Joacim Dybedal, and Geir Hovland. Automatic calibration of an industrial rgb-d camera network using retroreflective fiducial markers. *Submitted to Sensors*, 2018.

[D19] Andrew Maimone and Henry Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *Proceedings - IEEE Virtual Reality*, pages 51–54. IEEE, mar 2012.

[D20] Andreas Kunz, Luzius Brogli, and Ali Alavi. Interference measurement of kinect for xbox one. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology - VRST '16*, pages 345–346, New York, New York, USA, 2016. ACM Press.

Paper E

CNN-based People Detection in Voxel Space using Intensity Measurements and Point Cluster Flattening

Joacim Dybedal and Geir Hovland

This paper has been published as:

Joacim Dybedal and Geir Hovland. CNN-based People Detection in Voxel Space using Intensity Measurements and Point Cluster Flattening. *Modeling, Identification and Control*, 42(2):37–46, 2021.
doi: 10.4173/mic.2021.2.1.

CNN-based People Detection in Voxel Space using Intensity Measurements and Point Cluster Flattening

Joacim Dybedal* and Geir Hovland*

*University of Agder
Faculty of Engineering and Science
Jon Lilletunsvei 9, 4879 Grimstad, Norway

Abstract **In this paper real-time people detection is demonstrated in a relatively large indoor industrial robot cell as well as in an outdoor environment. Six depth sensors mounted at the ceiling are used to generate a merged point cloud of the cell. The merged point cloud is segmented into clusters and flattened into gray-scale 2D images in the xy and xz planes. These images are then used as input to a classifier based on convolutional neural networks (CNNs). The final output is the 3D position (x, y, z) and bounding box representing the human. The system is able to detect and track multiple humans in real-time, both indoors and outdoors. The positional accuracy of the proposed method has been verified against several ground truth positions, and was found to be within the point-cloud voxel-size used, i.e. $0.04\,\mathrm{m}$. Tests on outdoor datasets yielded a detection recall of $76.9\,\%$ and an F1 score of 0.87.**

E.1 Introduction

The ability to detect the presence of people or other objects in three dimensional data is an important factor in enabling autonomy and automation in environments where machinery and humans are both present. A natural example is in the automotive industry, where autonomous vehicles must be able to accurately perceive their surroundings. Another example is in any industrial environment where robotic machinery must coexist with personnel, whether it is on a large offshore platform or in a small indoor robotic cell.

Several approaches for people detection exist, where the problem of detecting people and other objects in 2D images is well documented in the academic literature, especially methods composing different types of machine learning such as Histograms of Oriented Gradients (HOG) [E1] and convolutional neural networks (CNNs). The problem of detecting people in 3D space, e.g. in point clouds, is also a hot topic.

Much research has been done on methods for detection in data from 2D and 3D lidars typically found on autonomous vehicles and mobile robots. In [E2], an approach based on implicit shape models (ISM) was used to detect people in a 3D lidar scan, and in [E3] person detection and tracking was done using a laser scan of the person's hip area and a Sample-based Joint Probabilistic Data Association Filter (SJPDAF). A 3D lidar was also used in [E4], where a bottom-up, top-down detector was used to select hypothetical candidates and perform validation on a tessellated volume, respectively.

For RGB-D images, the authors of [E5] used a tessellation boosting approach for human feature classification, e.g. different types of clothes and hairstyles. This approach, combined with the feature-based detection method from [E6], was used in [E7] to estimate human poses based on performant features extracted from colored point clouds. The detection method used a layer-based approach to calculate feature descriptors for each layer in a point cluster and concatenated the histograms to form feature vectors. In [E8] both 2D laser scans and RGB-D images were used for human detection and tracking, including keeping track of multiple humans in groups.

Human pose estimation has also been heavily researched and CNN-based methods such as OpenPose [E9] for 2D and [E10] for 3D pose estimations from 2D images. In [E11], RGB-D data was used for 3D pose estimation, using depth information in addition to the color image. Vehicle detection in lidar point clouds using neural networks such as VoxelNet [E12], 3D YOLO [E13] and [E14] also show promising results.

A common denominator of the methods mentioned above is that they are either tailor-made for lidar scans or that they depend on RGB images as well as any depth data. While the availability of RGB images is a justifiable assumption, there are scenarios in which redundancy is crucial, such as in the red-zone of oil-rigs. Such areas would typically be monitored by both RGB and depth-sensors (see [E15]), and while combining the measurements could yield the best detection results, in a scenario where the RGB information becomes unavailable a fall-back solution based on depth-information only should be available. In this paper, we therefore aim to remove the need for using RGB images when detecting people.

In addition, while most of the literature concentrates on single mobile sensors mounted on vehicles or robots, the method proposed in this paper utilizes a point cloud generated by several statically mounted 3D sensors as described by [E16], which is a just as likely scenario in industrial environments.

In [E17] Complex-YOLO is introduced, which is an extension of YOLOv2, a fast 2D standard object detector for RGB images, by a specific complex regression strategy to estimate multi-class 3D boxes in Cartesian space. A specific Euler-Region-Proposal Network (E-RPN) is proposed to estimate the pose of the object by adding

an imaginary and a real fraction to the regression network. The result is a closed complex space which avoids singularities, which can occur by single angle estimations. In our work multiple depth sensors are used and the data is merged into a single point cloud used for detection. Hence, the single angle problem mentioned in [E17] is not a problem in the work presented here.

In [E18] OpenPTrack is presented, which is an open source software for multi-camera calibration and people tracking in RGB-D camera networks. People detection is executed locally, in the machines connected to each sensor, while tracking is performed by a single node which takes into account detections from all over the network. For Kinect v1 and stereo cameras, which can produce color images, the HOG technique for people detection is applied to these images in correspondence of the clusters extracted from the point cloud. For the Kinect v2 the infrared images are used, since they are invariant to visible lighting.

In [E19] the proposed algorithms have been demonstrated further using aligned color and depth data in industrial environments. The algorithms have been released as open source as part of the ROS-Industrial project. The work presented in this paper is different from [E18, E19] in that a CNN-based approached is used and that the people detection is performed on the merged point cloud on the central node using only depth measurements, as opposed to detection locally on each node using color or infrared images in addition to depth information. The authors believe that people detection and tracking on a central node will be more robust, since the central node has access to the full point cloud, merged from all the sensors in real-time, however this has not been demonstrated experimentally in this work.

In [E20] an object detection method based on 3D information extraction of laser point clouds is proposed. Similar to the approach taken in this paper, in [E20] the point cloud is flattened to 2D images which are used as a basis for learning using the AdaBoost algorithm.

In [E21] the proposed method maps the three-dimensional point cloud to the two-dimensional plane by a distance-aware expansion approach. The corresponding 2D contour and its associated 2D features are then extracted. A radial basis function (RBF) kernel support vector machine (SVM) is employed with the extracted features for classification. A selective binary and Gaussian filtering regularized level set (SBGFRLS) algorithm is utilized for contour detection in the 2D images. In addition, other popular feature descriptors from the projected 2D images, such as HOG, LBP and Haar-like features are extracted.

[E22] present recent work on human detection and a classification scheme based on 3D Lidar data and an algorithm using a standard Support Vector Machine (SVM). The 3D indoor data used in [E22] has been made publicly available. In the conclusions the authors write: *Future work should look at other classification methods such as*

deep neural networks. The suggestion to use a CNN-approach is addressed in this paper, and the method developed here should fit such a system well, as it is designed to classify humans in point clouds containing intensity measurements, similar to the ones generated by a Lidar-based system.

This paper is organized as follows: Section 2 presents the overall methodology and the different modules that were developed. Section 3 contains several experimental results, while discussion and conclusions are given in Section 4.

E.2 Methodology

The main inspiration for our work was the '3D YOLO' type detectors where laser point clouds were used as inputs to image classifiers. To tackle the problem of human detection using only depth information, a scheme based on scene classification using convolutional neural networks (CNNs) was developed. To generate images for classification, a point cloud flattening approach was applied.

The input to the detector is a stream of compressed, voxelised point clouds, as described in [E16]. These streams are published as ROS (Robot Operating System) topics. Six Microsoft Kinect V2 3D sensors were used to generate the depth measurements, corresponding to six point cloud streams. Although the Microsoft Kinect sensors can supply RGB images, the purpose of this study was to become independent from RGB sensors, thus only the depth measurements were used.

The methodology is demonstrated in this paper by real-time multiple people detection in both indoor and outdoor environments.

Figure E.1 shows the structure of the developed system and the different steps will be further outlined in the following sections. The source code is available at [E23].

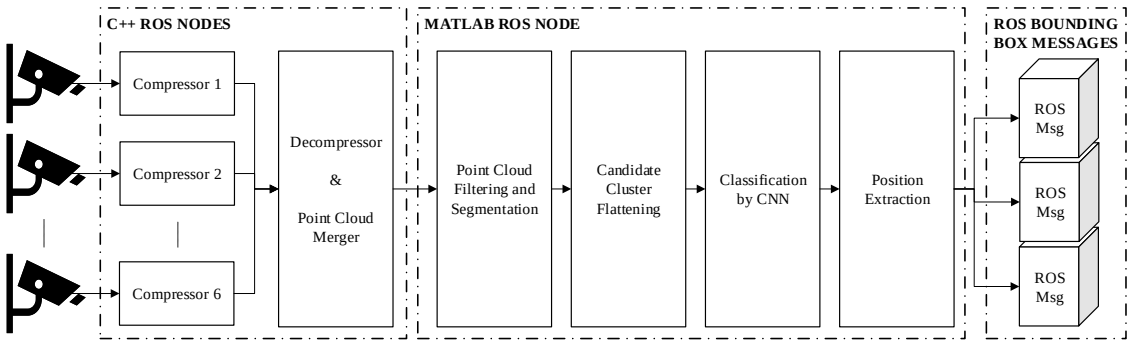


Figure E.1: Flowchart of the people detection system. The point clouds from each sensor is compressed by an edge computer, and received by a centralized computer before being decompressed and processed.

E.2.1 Point Cloud Pre-processing

Each of the six Kinect sensor nodes publishes compressed point cloud streams at up to 20 Hz. As described in [E16], the points in the compressed point clouds contain no colors, but an additional intensity value corresponding to the amount of measured points inside a single voxel, compensated for the distance to the sensor. The purpose of this addition was to add a measure of strength or confidence to each voxel as the point clouds were compressed, while accounting for the fact that objects close to the sensor will have a much higher point density than similar objects further away.

It should be noted that in addition to not including colors, the point clouds used in this paper are heavily compressed and down-sampled by voxelization. When using a voxel size of 4 cm × 4 cm × 4 cm and cropping to the volume of interest, [E16] found that each point cloud was typically reduced from 217 088 points to $17\,771 \pm 118$, with a reduction in required storage space of $1 : 40.5 \pm 0.5$.

In this paper, a ROS node was created which receives, synchronizes and merges the point clouds into a single cloud. The sensor nodes are time synchronized against a server, which ensures that the different point clouds are as close to each other as possible in the temporal space. Calibration to ensure optimized transfer functions between the sensors and the global coordinate system was performed according to the method described in [E24]. When merging the point clouds, the intensity values are accumulated such that the points in the merged point cloud contain the sum of all intensity values corresponding to the same voxel. The result and output of the developed ROS node is thus a single voxelised point cloud stream, where each point contains an intensity value in addition to x, y and z coordinates.

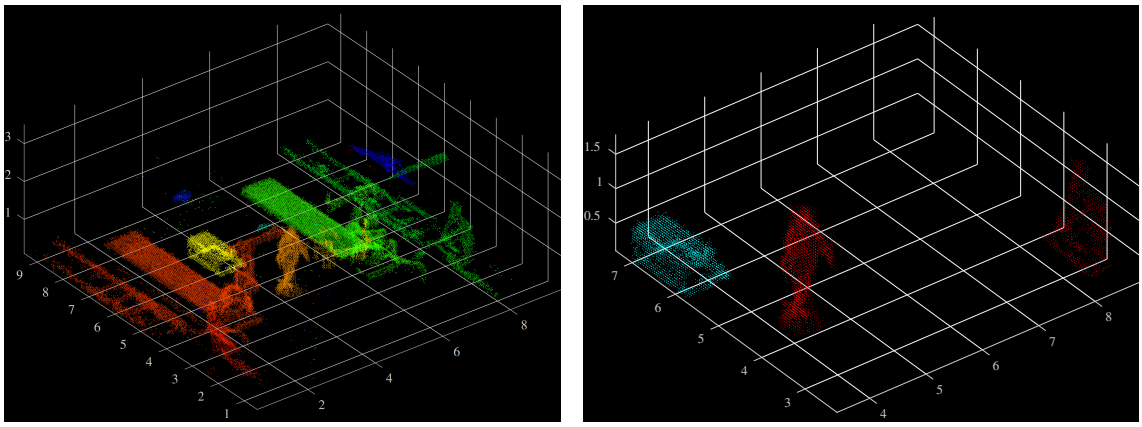
When the single, merged point cloud was obtained, it was segmented into clusters based on a minimum Euclidean distance between points in the clusters. However, most of the resulting clusters could be discarded, and only clusters defined to be a human candidate were used further in the detection scheme. To select the candidate clusters, the following constraints based on normal human poses were applied:

- Min, Max height (z dimension): 0.5 m and 2.0 m
- Min, Max width (x,y dimensions): 0.2 m and 2.0 m
- Max. distance from floor: 0.2 m

As seen in Figure E.2, this filtering resulted in a set of candidate clusters, which were used to generate gray-scale images for CNN-based training and classification.

E.2.2 Point Cluster Flattening

Several methods for calculating the positions of humans in point clouds were considered, including slicing the entire point cloud into segments in the xz and yz planes



(a) **(b)**

Figure E.2: (**a**) Candidate clusters after segmentation, but before dimension constraints; (**b**) Candidate clusters after dimension constraints have been applied.

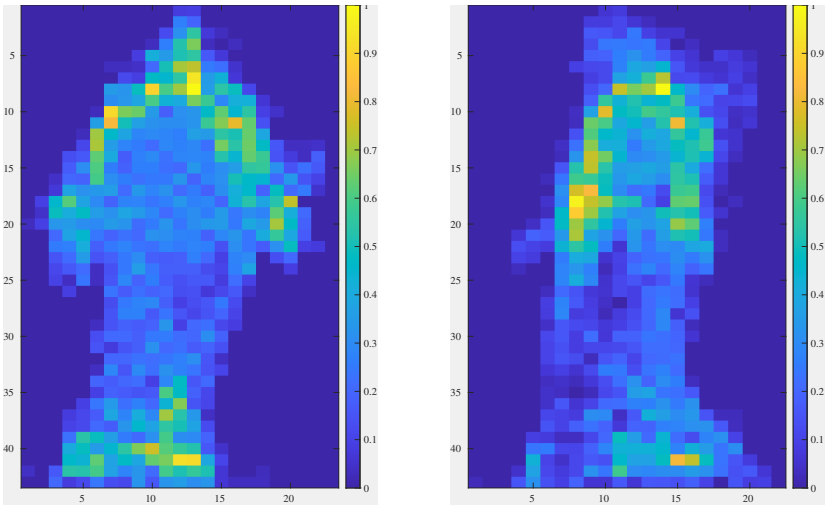
and classify/detect humans in each slice. However, due to the fact that humans would normally be present in only a fraction of the points, this was deemed too computationally expensive. In addition to the inevitable trade-off between accuracy (slice thickness) and speed, the same human, or parts of it, could be detected in multiple slices. Hence, the method developed in this paper was to flatten each candidate point cluster in the xz and yz planes, resulting in just two small slices per candidate.

As the input point cloud has already passed through a compressor based on a voxel grid, where each point has a fixed coordinate in the voxel center, and where the voxels are aligned to the global coordinate system, the process of cluster flattening could be implemented by iterating over the voxels in each dimension. Eq. E.1 shows the process of generating one pixel in the xz plane.

$$P(i,j) = \sum_{y=y_{min}}^{y_{max}} I(x_i, y, z_j), \tag{E.1}$$

where $P(i,j)$ is one pixel in the flattened image and $I(x_i, y, z_j)$ is the intensity value for the voxel at (x_i, y, z_j). The result is two gray-scaled images, where each pixel corresponds to the sum of the intensity values of the flattened points, as seen in Figure E.3. Using the intensity values to create a gray-scale image instead of just counting points results in images with a larger dynamic range. As the human body is solid, points can only exist on the exterior, which greatly limits the amount of points that would result in a single pixel in the flattened image.

Due to the coarse resolution of the voxelised point cloud, a human is typically represented by an image of only 25 times 45 pixels when a voxel grid resolution of 4 cm is used.



(a) **(b)**

Figure E.3: (**a**) Candidate cluster after flattening in the y dimension (xz plane); (**b**) Candidate cluster after flattening in the x dimension (yz plane). Color represents the accumulated intensity values, where yellow is higher.

E.2.3 Scene Classification

As the images are already cropped to the candidate point cluster, and the positions in the global coordinate system are known, there is no need for additional segmentation or bounding boxes. The classification of humans in the generated images could therefore be performed as a scene classification, i.e. the whole image was classified as a single class. As a proof of concept, a simple convolutional neural network was trained using images generated from captured datasets. The images corresponding to each candidate cluster in the training datasets were manually labeled as either 'Human' or 'Not Human'. A total of 1105 and 1956 images were used for the 'Human' and 'Not Human' classes, respectively. The two classes were chosen as the purpose here was to detect the presence of humans in a robotic environment. It would be possible to extend the number of classes to be able to classify any other object, such as the robots, other machines, equipment, etc.

The labeled images were then randomly binned into training images (60 %), validation images (30 %) and testing images (10 %). In addition, the images were augmented with random rotation ($\pm 90 deg$), reflection (around the z axis) and shear.

The structure of the neural network, including three convolution blocks, and the parameters used is shown in Table E.1. The structure was inspired by a simple example by [E25], but tuned to maximize the validation accuracy. The input layer is three-dimensional (RGB) with an input size of 224 times 224 pixels. Thus, in addition to resizing, the images needed to be augmented such that the gray-scale content was replicated across all RGB channels. As many image classifiers, including pre-trained networks, are expecting color images, converting from gray-scale to color

allows for a modular application where the classifier could be replaced by another without adding extra complexity. In contrast to many other scenarios, the images to be classified are very small, as described in the previous section. As a consequence, most images would need to be scaled up, not down, before being evaluated by the classifier.

Layer	**Parameter**	**Value**
Input Layer	Size	$224 \times 224 \times 3$
Convolution Layer	Filter Size	3
	No. of filters	16
	Padding	1
Batch Norm. Layer		
ReLu Layer		
Max-pooling layer	Pool Size	2
	Stride	2
Convolution Layer	Filter Size	3
	No. of filters	32
	Padding	1
Batch Norm. Layer		
ReLu Layer		
Max-pooling layer	No. of pools	2
	Stride	2
Convolution Layer	Filter Size	3
	No. of filters	64
	Padding	1
Batch Norm. Layer		
ReLu Layer		
Fully-connected layer	Output Size	2
Softmax Layer		
2-class Classifier		

Table E.1: CNN Structure and layer parameters.

Training of the network was performed in Matlab by the stochastic gradient descent with momentum (SGDM) optimizer. Using 100 epochs and an initial learning rate of 0.00001, the final validation accuracy was 95.75%.

E.2.4 Labeling and Position Extraction

The nature of the detection mechanism developed in this paper includes a coarse confidence measure, i.e. a human can be classified in either zero, one, or both of the flattened images. For the testing described in this paper, only the scenarios where both images are classified as humans were considered a detection.

In the event of a detection, the developed ROS program publishes a "marker" message that includes the position and extent of the detected human. The position used is the center of the bounding box surrounding the candidate point cluster, and the bounding box itself is used to describe the area where the person was detected.

E.3 Experimental Results

Four different test cases were used to evaluate the system. First, the scheme was tested in the Industrial Robotics Lab (IRL) at the University of Agder [E16]. In this environment, six 3D sensor nodes consisting of Microsoft Kinect V2s and NVIDIA Jetson TX2s are mounted on the walls around a robotic cell. Inside the area monitored by the sensors, there are three ABB robots, where two are track mounted and one is mounted on a Güdel gantry crane. The test served as a proof-of-concept, verifying that the detector worked on live data different from the datasets used for training and validation.

Second, the accuracy of the system was measured by comparing detected coordinates with ground truth coordinates measured by a Leica Laser Tracker. The same setup as in the first experiment was used.

Third, the scheme was tested using a datasets recorded at an outdoors test facility for offshore pipe handling equipment, without altering the algorithms. The datasets were recorded in August 2018, using the same sensor nodes as in the IRL lab, and includes recordings with multiple people in different weather conditions.

Lastly, the CNN was re-trained using outdoor data, and a detection capability test was performed, measuring possible detections versus actual detections.

All experiments were performed on a stationary computer running the detector in Matlab. The computer was running Ubuntu 16.04 with ROS Kinetic and contained an Intel Core i7 7820X 3,6 GHz processor, an NVIDIA GTX 1080Ti GPU and 32 GB of RAM.

The following subsections present the results from the three different test cases. The outdoor dataset, training data and a demonstrative video are made publicly available at [E26].

E.3.1 Indoor Single Person Detection

After the detector had been trained, it was tested on a single person in the IRL lab. The aim was to verify that the detector would yield good results on data different to the training data.

Figure E.4 shows the detected bounding boxes around a person in different poses. In (a), the detected position coordinates were x =4.42 m, y =5.22 m, z =0.94 m,

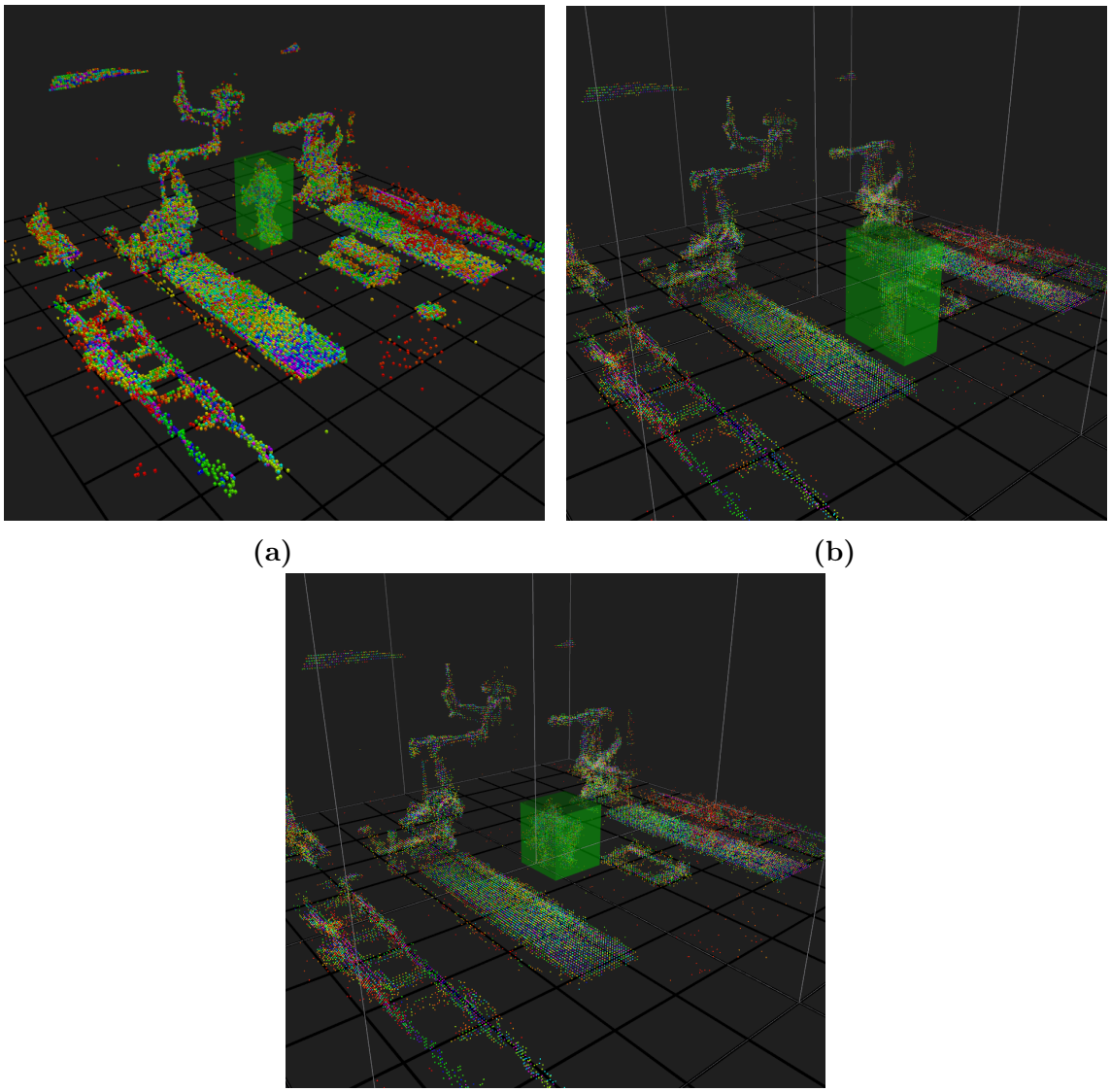


Figure E.4: (**a**) Example detection, walking; (**b**) Example detection, arms out; (**c**) Example detection, crouching.

where this point is the center of the displayed bounding box. The detection algorithm functioned as expected, marking the person with correctly sized bounding boxes for all the tested poses.

In addition, a small set of 160 'Human' images and 70 'Not Human', where the subject had different outfits than used in the training, was used to test the CNN. The accuracy in this test was 88.2%, which as expected is a little lower than the training accuracy, but still close to 90%.

E.3.2 Indoor Accuracy Validation

To verify the accuracy of the detector, the output X, Y coordinates were compared to coordinates that had been previously measured to sub-millimeter accuracy by

a Leica laser tracker and marked on the floor. A person would walk around the monitored area, stopping at the marked coordinates, before moving on to the next. Performing the test in such a way would introduce some inaccuracy, as it is not possible to guarantee that the person is standing exactly above the ground truth coordinate. However, since the resolution of the input point cloud was as coarse as 0.04 m, the test would still yield useful results.

A total of 21 coordinates were pre-measured. During the test, a total of 94 detections were performed at the ground truth positions, and a subset is shown in Table E.2. The result is shown in Table E.3, and it can be seen that the mean absolute deviation of the detector was within the point cloud resolution of 0.04 m.

X	Y	X meas.	Y meas.	Z meas.
5.0	8.0	4.96	8.06	0.94
5.0	7.0	5.02	7.02	0.94
5.0	6.0	4.96	6.02	0.92
5.0	5.0	5.02	4.98	0.94
4.69	5.3	4.72	5.23	0.92
4.0	3.0	4.02	2.98	0.94
2.0	7.0	5.02	7.02	0.94
5.42	9.24	5.5	9.2	0.94

Table E.2: A subset of results obtained during the accuracy validation.

	X	Y	Total
Mean absolute deviation (m)	0.035	0.039	0.037
Std.dev absolute deviation (m)	0.041	0.035	0.038
Mean deviation (m)	−0.001	−0.004	−0.003
Std.dev deviation (m)	0.054	0.053	0.053

Table E.3: Results from accuracy test using 94 detections compared to ground truth coordinates.

The X and Y coordinates are the ones that have been used for verification, as these are the only ones with an accurate ground truth. However, the bounding box also contains the width, depth and height of the detected person. While the width and depth can vary greatly due to different poses, the measured height was compared to the height of the test subject. Using 56 of the detections performed while the subject was standing, the measured height was (1.76 ± 0.06) m, and the real height of the test subject was 1.75 m.

E.3.3 Testing on Outdoor Datasets

To test the detection algorithm on more challenging data, datasets recorded on an outdoors facility were used. The datasets were recorded in August 2018, using the same sensors as used for the two first experiments, and contains a variety of weather conditions and multiple persons. Due to the lack of ARUCO codes used for the automatic calibration performed in the indoor experiments, the sensors' translation and rotation were calibrated manually using other features in the point clouds.

Figures E.5 (a) and (b) show a snapshot of the first test, where four persons are present. As seen in Figure E.5 (a), the weather conditions were challenging, with a low sun shining on wet concrete. The persons were all wearing different outfits, in addition to helmets, which were not part of the original training data. However, without altering the algorithm, i.e. only using training data from the indoor experiments, it was able to detect all four persons.

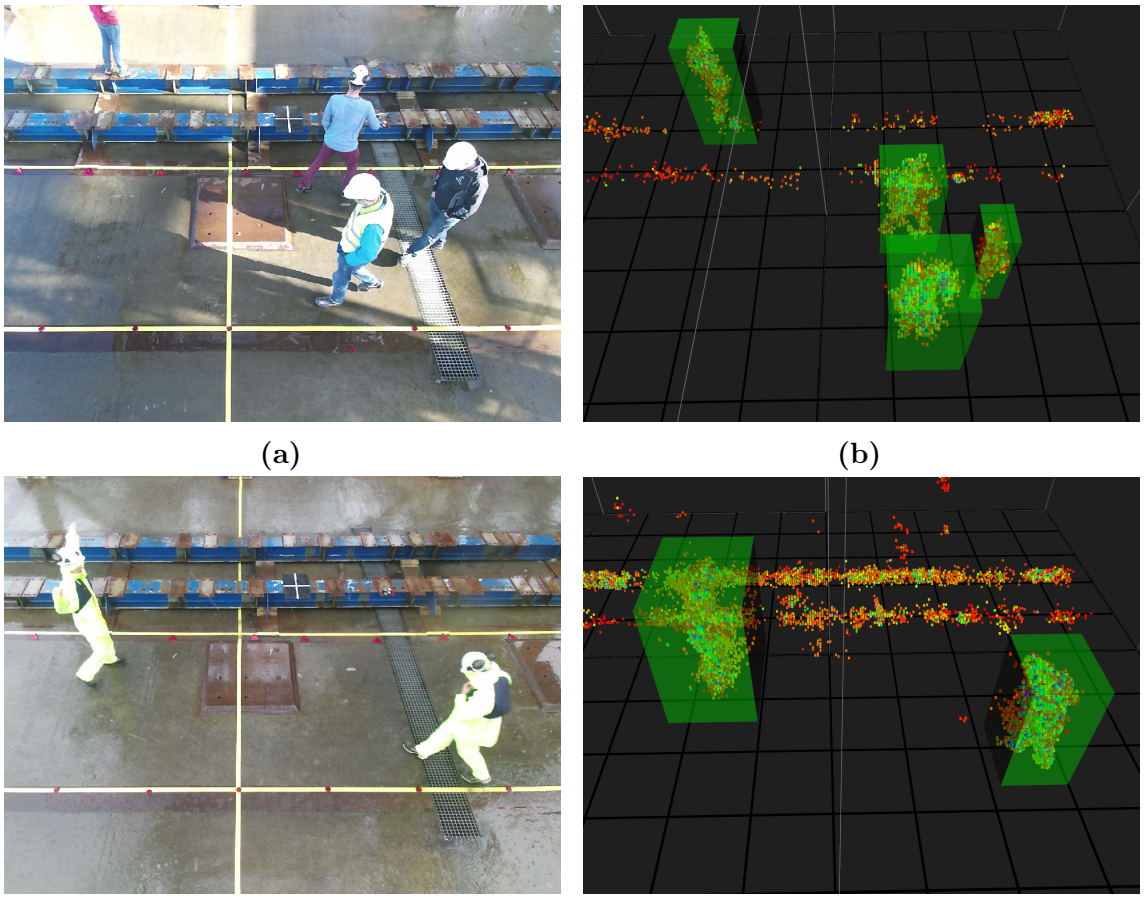


Figure E.5: Detection test on Outdoor Data - **(a)** and **(b)** Four people present and detected at the test site, in low sun on wet ground; **(c)** and **(d)** Two people present and detected, in heavy rain.

Another dataset was recorded in heavy rain. While the rain was causing an increased amount of noise in the point clouds, the system was still able to correctly

classify humans, as seen in Figures E.5 (c) and (d).

E.3.3.1 Human Detection Performance

To determine the detection capability of the system, the network was trained again, this time including data from the outdoor datasets. Six datasets, each consisting of approximately 1500 merged point cloud frames with 0-2 persons present, were added to the training and validation data. After image extraction and manual labeling, the final training and validation set now consisted of 4715 images for the 'Human class', and 6805 images for the 'Not Human class'.

A seventh dataset was used for testing. This dataset contained a 300 s recording where up to four people were present at the same time. The persons were walking randomly, resulting in many different poses, as well as entering and leaving the area. From this set, a total of 949 merged point clouds were analyzed, where 21 contained one person, 83 contained two persons, 324 contained three persons and 521 contained four persons, i.e. there were a total of 3243 clusters that should be classified as humans. The detection accuracy (recall) was calculated by dividing the number of Humans actually detected by the total number of possible detections. This yielded a recall of 76.9 % (2495 out of 3243 humans were correctly detected). The calculated F1 score was 0.87, due to a high precision of 99.9 %.

During the testing, while the point clouds were streamed at around 5 Hz, the detector was outputting results at about 1.2 Hz. The relatively low rate is due to the fact that the detection system is run in Matlab and that temporary images are written to disk before they are classified.

E.4 Discussion and Conclusions

This paper has demonstrated a novel approach to people detection on very sparse point clouds using only depth information. The proof of concept and accuracy experiments show promising results, with an absolute error that is approximately the same as the resolution of the voxelised point cloud at ±4 cm.

During the accuracy test, the test subject was wearing different clothing with different colors than used in the training data. This implies that the use of intensity values and not colors in the point clouds allows the classifier to distinguish objects based primarily on shape, without being biased by certain outfits.

The tests on outdoor datasets yielded a detection recall of 76.9 % and an F1 score of 0.87. While this is not optimal, further investigation has led to the conclusion that this is mostly due to the filtering and segmentation process, and not the classification. Firstly, work should be done to optimize the segmentation to better distinguish

persons from the environment. Typical scenarios are where persons are standing on or close to constructions, or carrying large items. In these scenarios, the Euclidean distance segmentation approach struggles to create the correct candidate clusters, and the clusters are discarded immediately and never passed to the classifier.

Another improvement would be to classify parts of persons instead of whole persons. In some scenarios, e.g. when wearing very dark clothes or when occluded by objects, parts of the body may not be visible in the point clouds. In an example where only the torso is visible, the current candidate cluster processing would discard these clusters as they would not be within the constraints. The same problem occurs when a person is entering or exiting the monitored area, leading to partial point clouds of the body.

As our approach is using a scene classifier, the approach could be extended to detect any class of items by adding more classes during training or by training on other classes entirely. This makes it possible augment the system to detect human body parts or any other objects such as robots and equipment. If needed, the system could also be extended by including the (x,y) plane in the flattening process, thus generating a third image for classification at the cost of increased computational load.

Future work should also include testing the classifier on other, published datasets, to compare the performance to other people detection techniques. Specifically, the precision of the presented solution was found to be 99.9 %, however there were few foreign objects present in the data sets which led to very few false positives. In addition, work should be done to port the classifier and detector from MATLAB to a more efficient environment, e.g. a native ROS node, to achieve a higher maximum detection rate than the 1.2 Hz achieved in this paper.

E.5 Acknowledgments

The research presented in this paper has received funding from the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896.

References – Paper E

[E1] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proc. 2005 IEEE Comp. Soc. Conf. on Comp. Vision and Pattern Recognition, CVPR 2005*, volume I, pages 886–893, 2005.

[E2] B Borgmann, M Hebel, M Arens, and U Stilla. Detection of Persons in MLS Point Clouds using Implicit Shape Models. *pf.bgu.tum.de*, 2017.

[E3] Ju Min Kim, Young-Joo Kim, and Chang-Bae Moon. Human Target Tracking using a 3D Laser Range Finder based on SJPDAF by Filtering the Laser Scanned Point Clouds. *Intl. J. Control, Automation and Systems*, 18(X):1–11, 2020.

[E4] Luciano Spinello, Matthias Luber, and Kai O. Arras. Tracking people in 3D using a bottom-up top-down detector. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 1304–1310, 2011.

[E5] Timm Linder and Kai O. Arras. Real-time full-body human attribute classification in RGB-D using a tessellation boosting approach. In *IEEE Intl. Conf. Intelligent Robots and Systems*, pages 1335–1341, 12 2015.

[E6] Benjamin Lewandowski, Jonathan Liebner, Tim Wengefeld, Steffen Muller, and Horst Michael Gross. Fast and robust 3D person detector and posture estimator for mobile robotic applications. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 4869–4875, 5 2019.

[E7] Tim Wengefeld, Benjamin Lewandowski, Daniel Seichter, Lennard Pfennig, and Horst Michael Gross. Real-time person orientation estimation using colored pointclouds. In *Proc. 2019 European Conf. Mobile Robots*, 9 2019.

[E8] Timm Linder and Kai O. Arras. People detection, tracking and visualization using ROS on a mobile service robot. *Studies in Computational Intelligence*, 625:187–213, 2 2016.

[E9] Zhe Cao, Tomas Simon, Shih En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proc. 30th IEEE Conf. Comp. Vision and Pattern Rec., CVPR 2017*, pages 1302–1310, 11 2017.

[E10] Denis Tome, Chris Russell, and Lourdes Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2500–2509, 2017.

[E11] Christian Zimmermann, Tim Welschehold, Christian Dornhege, Wolfram Burgard, and Thomas Brox. 3D Human Pose Estimation in RGBD Images for Robotic Task Learning. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 1986–1992, 9 2018.

[E12] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Proc. IEEE Comp. Soc. Conf. Comp. Vision and Pattern Rec.*, pages 4490–4499, 11 2017.

[E13] Ezeddin AL Hakim. 3D YOLO: End-to-End 3D Object Detection Using Point Clouds. Technical report, Dissertation, 2018.

[E14] Martin Simon, Karl Amende, Andrea Kraus, Jens Honer, Timo Samann, Hauke Kaulbersch, Stefan Milz, and Horst Michael Gross. Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[E15] Velodyne Lidar. Automated with Velodyne | The Marsden Group | Velodyne Lidar, 2020.

[E16] Joacim Dybedal, Atle Aalerud, and Geir Hovland. Embedded Processing and Compression of 3D Sensor Data for Large Scale Industrial Environments. *Sensors*, 19(3):636, 2 2019.

[E17] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[E18] Matteo Munaro, Filippo Basso, and Emanuele Menegatti. OpenPTrack: Open source multi-camera calibration and people tracking for RGB-D camera networks. *Robotics and Autonomous Systems*, 75:525–538, 1 2016.

[E19] Matteo Munaro, Christopher Lewis, David Chambers, Paul Hvass, and Emanuele Menegatti. RGB-D human detection and tracking for industrial environments. In *Adv. Intell. Systems and Computing*, pages 1655–1668, 2016.

[E20] Li Hui, Liu Yun, Qian Meiyi, and Pei Shujuan. Object detection method based on three-dimension information extraction of laser point cloud. In *ACM Intl. Conf. Proc. Series*, pages 208–213, New York, USA, 1 2019. Association for Comp. Mach.

[E21] Hsueh Ling Tang, Shih Che Chien, Wen Huang Cheng, Yung Yao Chen, and Kai Lung Hua. Multi-cue pedestrian detection from 3D point cloud data. In *Proc. IEEE Intl. Conf. Multimedia and Expo*, pages 1279–1284, 8 2017.

[E22] Zhi Yan, Tom Duckett, and Nicola Bellotto. Online learning for 3D LiDAR-based human detection: experimental analysis of point cloud clustering and classification methods. *Autonomous Robots*, 44:147–164, 2020.

[E23] Joacim Dybedal. Human detector for point clouds using point cloud flattening an CNN scene classifier. https://github.com/dybedal/wp3-human-voxel-detector, 2021.

[E24] Atle Aalerud, Joacim Dybedal, and Geir Hovland. Automatic Calibration of an Industrial RGB-D Camera Network Using Retroreflective Fiducial Markers. *Sensors*, 19(7):1561, 3 2019.

[E25] The MathWorks, Inc. Scene Classification Using Deep Learning. https://blogs.mathworks.com/deep-learning/2019/11/25/scene-classification-using-deep-learning/, 2021. Accessed: 2021-04-28.

[E26] Joacim Dybedal. Replication Data for: CNN-based People Detection in Voxel Space using Intensity Measurements and Point Cluster Flattening. *DataverseNO*, 2021.