FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Miiro Kuosmanen
Samu Majabacka
Kalle-Oskari Suvanto

REAL-TIME HUMAN DETECTION FROM DEPTH IMAGES WITH HEURISTIC APPROACH

Bachelor's Thesis
Degree Programme in Computer Science and Engineering
June 2020

Kuosmanen M., Majabacka S., Suvanto K. (2020) Real-Time Human Detection from Depth Images with Heuristic Approach. University of Oulu, Degree Programme in Computer Science and Engineering, 35 p.

ABSTRACT

The first industrial robot was built in the mid-20th century. The idea of the industrial robots was to replace humans in assembly lines, where the tasks were repetitive and easy to do. The benefits of these robots are that they are able to work around the clock and only need electricity as compensation. Over the years, robots capable of only doing repetitive tasks have evolved to operate fully autonomously in challenging environments. Some examples of these are self-driving cars and service robots that can work as customer servants. This is mainly accomplished through advancements in artificial intelligence, machine vision, and depth camera technologies. With machine vision and depth perception, robots are able to construct a fully structured environment around them and this allows them to properly react to sudden changes in their surroundings.

In this project, a naive detection algorithm was implemented to separate humans from depth images. The algorithm works by removing the ground plane, after which the floating objects can be separated more easily. The floating objects are further processed, and the human detection part is then achieved using a heuristic approach. The proposed algorithm works in real time and reliably detects people standing in a relatively open environment. However, because of the naive approach, human-sized items are wrongly detected as humans in some scenarios.

Keywords: robotics, service robot, stereo vision, point cloud, Intel RealSense

Kuosmanen M., Majabacka S., Suvanto K. (2020) Reaaliaikainen ihmisten havainnointi syvyyskuvista heuristisella menetelmällä. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 35 s.

TIIVISTELMÄ

Ensimmäinen teollisuusrobotti rakennettiin 1900-luvun puolivälissä. Teollisuusrobottien tarkoitus oli korvata ihmiset tehtaiden kokoonpanolinjoilla, joissa työtehtävät olivat pääsääntöisesti yksinkertaisia ja itseään toistavia. Näiden robottien etuna on, että ne kykenevät työskentelemään kellon ympäri pelkän sähkön varassa. Vuosien mittaan robotit ovat kehittyneet yksinkertaisista koneista roboteiksi, jotka kykenevät toimimaan täysin itsenäisesti haastavissakin olosuhteissa. Itseajavat autot ja asiakaspalvelijana toimivat palvelurobotit ovat näistä hyviä esimerkkejä. Tällaiset saavutukset ovat olleet mahdollisia tekoälyn, konenäön ja syvyyskameroiden kehityksen myötä. Kone- ja syvyysnäön avulla robotit pystyvät muodostamaan itselleen selkeän kuvan ympäristöstään, mikä mahdollistaa nopean reagoinnin yllättäviinkin muutoksiin ympäristössä.

Tässä työssä toteutettiin naiivi havaitsemisalgoritmi erottelemaan ihmiset syvyyskuvista. Algoritmi poistaa maatason, jonka jälkeen ilmassa leijuvat esineet voidaan erotella toisistaan. Erotetut esineet jatkokäsitellään, jonka jälkeen ihmisten havaitseminen toteutetaan heuristisella menetelmällä. Työssä esitelty algoritmi toimii reaaliajassa ja pystyy luotettavasti havaitsemaan ihmiset suhteellisen avoimessa ympäristössä, vaikkakin joissain tapauksissa ihmisen kokoiset esineet luokitellaan väärin ihmisiksi naiivin lähestymistavan vuoksi.

Avainsanat: robotiikka, palvelurobotti, stereonäkö, pistepilvi, Intel RealSense

TABLE OF CONTENTS

FOREWORD

We want to thank Teemu Tokola and Juha Röning for supervising and inspecting this work. In addition, we want to thank the University of Oulu's IT department for providing the depth camera (Intel RealSense D435) used in this project.

Oulu, June 13th, 2020

Miiro Kuosmanen
Samu Majabacka
Kalle-Oskari Suvanto

LIST OF ABBREVIATIONS AND SYMBOLS

IFR	International Federation of Robotics
ISO	International Organization for Standardization
LED	Light-Emitting Diode
LiDAR	Light Detection and Ranging
Radar	Radio Detection and Ranging
ToF	Time of Flight
IR	Infrared
DK	Development Kit
IMU	Inertial Measurement Unit
SDK	Software Development Kit
PCL	Point Cloud Library
OpenCV	Open source Computer Vision Library
RANSAC	Random sample consensus
RGB	Red, Green and Blue
RGB-D	Red, Green, Blue and Depth
FPS	Frames Per Second
AI	Artificial Intelligence

1. INTRODUCTION

When someone says the word "robot" the image that comes to people's mind can differ greatly. It can be anything between a very simple industrial robot and a futuristic "Terminator" type android. 60 years ago, robots used to be more like the former type assigned with a simple repetitive task, but with the help of computationally more powerful computers, machine learning, and more advanced algorithms, the field of robotics is aiming to move towards the latter.

This computation power correlates with the number of transistors in an integrated circuit, and since the transistor count in these integrated circuits has doubled every two years [1], computational power has also been steadily increasing for the past 50 years. For example, in 1996, IBM developed a chess-playing computer Deep Blue, which managed to beat chess world champion Garri Kasparov with an average of 50-100 million searched positions per second [2]. If we compare Deep Blue to a modern off-the-shelf laptop, the laptop can achieve up to 60 times more calculations per second than the Deep Blue was able to. In 2015, AlphaGo was the first computer program that was able to beat Fan Hui, a professional Go player, and this required more than just raw computational power. During a game of Go, there are more possible moves than there are atoms in the universe. Therefore, AlphaGo used a combination of machine learning and tree search techniques to achieve this task [3].

With the growth of computing power, better algorithms and machine learning, human-like robots have started to develop from mechanical machines that are not able to walk [4] to robots that are able to cross obstacles, run and even jump while swinging their arms the same way human athletes do to propel themselves forward [5]. Even though a lot has been achieved in regard to robot movement and behavior those two still tend to be the topic of many research projects. These incredible feats would not be possible for a robot without vision and some kind of understanding about their surroundings. Nowadays, stereo cameras have become much more affordable for everyday consumers and they are also included in many mobile phones. This proliferation of stereo cameras has increased interest in them, and consequently, more research on stereo vision is being done. In addition to what 2D cameras already provide, stereo cameras offer depth information which has enabled industrial robots to be able to grab and lift objects reliably and perform tasks that require high precision. Depth information is also used in various other applications as well, for example it allows face spoof detection, which results in more accurate and reliable face recognition.

The purpose of this work was to implement an real-time algorithm capable of detecting humans with depth information alone. This was done by experimenting and comparing functionality, accuracy, and processing time of different depth information processing methods.

2. ROBOTICS

The first idea of a robot comes from the Greek mythology more than 2500 years ago. In this ancient story, Hephaestus, the god of invention, created the first android named Talos to defend the island of Crete from unwanted visitors. Talos was eventually defeated by a woman named Medea, who managed to trick Talos by promising immortality to him, after which Medea quickly pulled out a vitally important bronze nail behind Talos's ankle, and Talos collapsed to his death. Even though the story of Talos is just fiction, it still shows the idea of human-robot interaction and the concerns of too human-like robots. [6]

The first designed and possibly constructed robot was by Leonardo da Vinci in the year 1495, but it was not made public before the 1950's when the design of the robot was found from Leonardo da Vinci's sketchbook [7]. There have been ideas and designs of robots for a long time, but the term "robot", from the czech word "robota" meaning forced labor [8], was first time used in a 1920 play of R.U.R. "Rossum's Universal Robots" by Karel Capek.

In the 1950's, George Devol invented the first industrial robot Unimate, which was installed to General Motors assembly line in New Jersey in 1961. After the installation of Unimate, large car manufacturing companies noticed the importance of industrial robots and so, one year later, the world's first robotics company called Unimation was founded by Joseph F. Engelberger and George Devol [9]. Before the 1990's, research and development focused mainly on industrial robots, because they reduced manufacturing costs immensely, and investing in robotics cost so much that neither households nor smaller companies could afford it. After the 1990's, the price of the components used in robotics started to drop and service robots started to find their way into households. This meant that the amount of service robots started to compete with industrial robots. After the change of millennium, the general interest in robots has continued to steadily increase, but service robots are starting to take the upper hand as seen in Figures 1 and 2. In 2018, the estimated amount of annual installations of industrial robots was only around 414 000 units compared to 16.3 million installations of service robots [10].

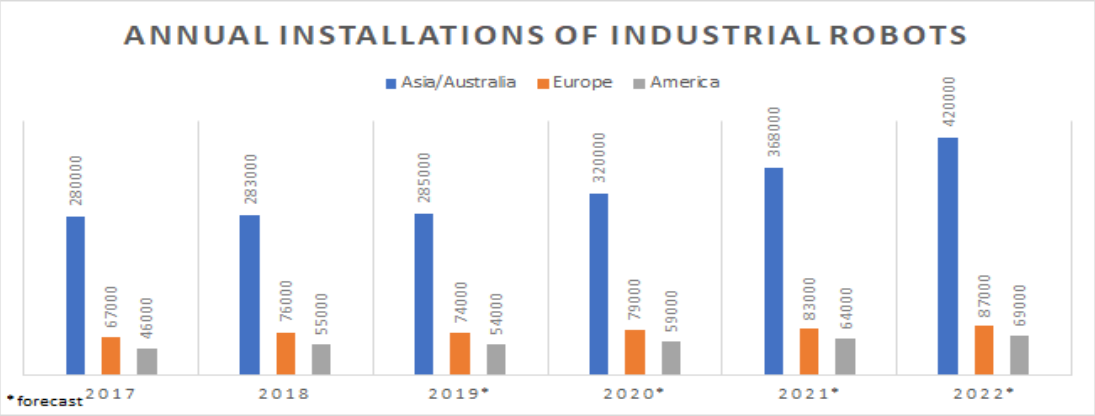

Figure 1. Forecast of the growth of industrial robots. [1]

[1]Data was adapted from: https://ifr.org/downloads/press2018/IFR%20World%20Robotics%20Presentation%20-%2018%20Sept%202019.pdf

2.1. Service Robots

Thanks to improvements in sensor technology and machine vision, robots are no longer restricted to operate only in the industrial field. They are now able to operate among humans and perform tasks that serve human needs [11]. Due to these accomplishments, the industry has started to develop and produce more service robots instead of only focusing on the industrial ones. According to a forecast made by the IFR (International Federation of Robotics), the value of service robots in both professional and personal fields will increase immensely in the upcoming years as seen in Figure 2.

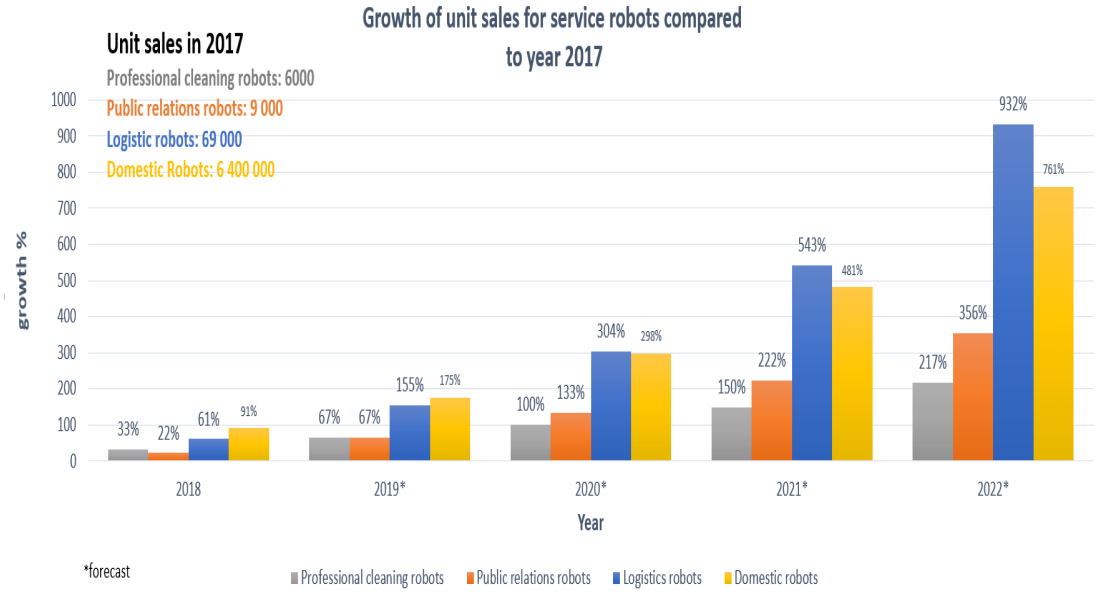

Figure 2. Forecast of the growth of service robots.[2]

A service robot is defined as a "robot that performs useful tasks for humans or equipment excluding industrial automation applications" [12]. These robots can be subcategorized in different ways. One of them is presented in Table 1, where the classification is done by their user type. This classification was presented by ISO (International Organization for Standardization) [12]. Even though the classification in Table 1 is just one of many, service robots' main purpose always remains the same: they are designed to help people in repetitive, time-consuming or dangerous tasks.

Since service robots have to operate in human-oriented environment, it is important that they can react to sudden changes in their working environment in real time. This requires either computationally efficient algorithms or a lot of computing power. The former seems to be preferred because computing power usually costs a lot more and also requires more space, which is often limited. For example, an autonomously working robot vacuum has to be affordable, small enough to fit in tight spaces and aware of its surroundings at all times, otherwise there is a chance that it could collide with a human.

Both professional and personal service robots often have to be able to work autonomously in a natural or unstructured environment because their working

[2]Data was adapted from: `https://ifr.org/downloads/press2018/IFR%20World%20Robotics%20Presentation%20-%2018%20Sept%202019.pdf`

Table 1. Usage of service robots

	Personal Service Robots (used for non-commercial tasks)	**Professional Service Robots** (used for commercial tasks)
Used by	Lay person	Properly trained operator
Examples	Domestic servant robots Automated wheelchairs Personal mobility assist robots Pet fitness robots	Customer service robots Medical robots Defence robots Logistic robots

environment is seldom predefined when manufacturing the robot. On the contrary, when designing an industrial robot, its working environment and intended task is usually known. For example, they might work in an assembly line and their only purpose can be picking an object from the assembly line or screwing a screw into an object. This task is then repeated around the clock. In contrast, service robots have to be designed in such a way that they can work in vastly different environments. For example, a robot vacuum cleaner has to be able to operate in different interiors. This means that when designing a service robot, all of its components, sensors, and software must be thoroughly considered.

2.2. Robot Behaviour and Appearance

When designing a service robot, one must of course think about the technology behind the metal surface, but there is also importance in the looks and the actions of the robot [13]. For us people, it seems that it is not enough to just have a service robot that does its job, but we actually prefer to have the service robot behave differently and look different depending on its task [14]. For example, having a robot whose behavior can be interpreted as happy and excited (like in a study done by Martin Cooney and Stefan M. Karlsson [15]) can be beneficial for a companion robot but useless for a customs robot.

Human-like behavior, in particular, would be a surprisingly beneficial topic to master. In a study [16] done by Chien-Ming Huang and Bilge Mutlu, a robot gives instructions to a test subject to do certain tasks. The results stated that when robots use human-like guiding gaze while teaching or giving instructions, humans tend to learn faster and much more efficiently. From this, one can speculate that having a robot behave with human-like gestures would be very beneficial in the field of robotics since just a simple guiding gaze can be this effective.

Perfecting this gaze requires a deep understanding of how robots see the environment around them. Robots need to be able to recognize objects, humans, and everything related to their surroundings. Machine vision has made some remarkable achievements in object recognition, and with the help of stereo cameras, it is possible to give robots a much more descriptive image of the environment they are in.

2.2.1. Humanoid Looks

Interacting with robots that almost look like humans but not quite, tends to give us an eerie and uncanny feeling. We can tell something is wrong, but we cannot exactly point our finger at it. This is called the Uncanny Valley effect and it demonstrates how the robot needs to either look exactly like a human or not even close [17]. Uncanny Valley effect is an important factor to consider in human-robot interaction and is illustrated in Figure 3. Another example of this can be seen in Figures 4 and 5. Even though the robot in Figure 4 does not look human, it still looks pleasant unlike the robot in Figure 5 [18].

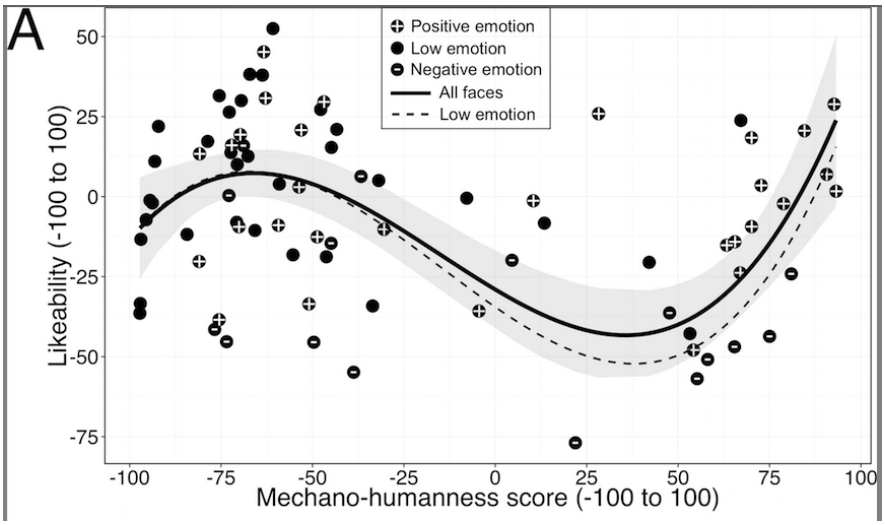

Figure 3. Uncanny Valley effect[3]

Figure 4. Robot from a well-known animated movie[4]

Figure 5. Humanoid robot for children with autism[5]

[3]Maya B. Mathur and David B. Reichling (2016) CC: https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode

[4]Lenin Estrada (2019) CC: https://creativecommons.org/publicdomain/zero/1.0/legalcode

[5]Luke J. Wood, Abolfazl Zaraki, Ben Robins and Kerstin Dautenhahn (2019) CC: https://creativecommons.org/licenses/by/4.0/legalcode

2.2.2. Speech

Speech is one of the most important factors to consider when designing a human-like robot. Without this ability, robots would not be able to make a leap from humanoid to human-like. While communication with a robot can be achieved by attaching a screen onto the robot's chest, this kind of communication does not fulfil the criteria of human-like conversation. Creating a robot with exceptional comprehension and speaking capabilities is a challenging task because the meaning of a sentence can completely change with just the tone of one's voice [19]. On top of this, even if you had a robot that could answer and talk to you with a perfectly human-like voice and sentence structures, it would still be far from perfect if the whole spectrum of human communication was considered.

2.2.3. Movement

We humans use the motion of our hands and head to support what we say and mean [20, 21]. This means that a robot's communication capabilities can be greatly enhanced by the movement of its limbs [16]. Designing these movement patterns can be difficult. Exaggerated or sluggish movements give an unnatural image of the robot and it can even feel irritating. Also, the length of the robot's motions has to be considered. In a study with a playful robot, an inverted u-shaped correlation between motion length and degree of perceived playfulness was observed [22]. This means that the robot's movements should be neither too long nor too short.

2.2.4. Eye Movement

If one wants to design a robot that feels natural and human-like as a conversational partner, in addition to speech and the movement of the limbs, realistic eye-movement is required. During conversations, we humans unconsciously use our eyes surprisingly often. Eye movements are used to maintain and begin conversations, to express emotions, and as guidance [23, 24, 25]. Implementing this into a robot requires a deep understanding of machine vision and image processing. Thankfully, both of these fields have advanced far, and nowadays it is even possible to use a regular laptop to develop software that can recognize objects or people. However, recognition in real time and with 100% accuracy has proven to be difficult [26, 27, 28].

3. TECHNOLOGIES AND EQUIPMENT

In order for robots to be able to operate safely among humans, they need to have the ability to hear and see, as humans do. These abilities allow the robots to properly react to sudden changes in their surroundings. For example, autonomous cars have to be able to react properly when a child crosses the road in front of them.

The ability to detect the child can be obtained through a standard RGB (Red, Green and Blue) camera, but in order to know how far the child is, robots need depth perception. This can be achieved through different depth cameras, which use various measurement technologies to gather depth information about the environment.

This chapter introduces the 3D techniques for acquiring depth information, as well as the main depth technologies and the key differences between them. In addition, some of the most used depth data processing libraries are also presented.

3.1. Depth Measurement Technologies and Techniques

For humans, depth perception was naturally evolved, but for machines it had to be developed. Designing a machine that can perceive depth can be done using depth measurement technologies or 3D imaging techniques. However, using multiple technologies and techniques together to get the most accurate and precise result is advised [29]. For example, most of the currently designed autonomous cars use ultrasonic for close range, LiDAR (Light Detection and Ranging) for medium range and Radar (Radio Detection and Ranging) for long range detection. This use of different technologies in unison, instead of just using one, gives much more accuracy at different distances.

3.1.1. Ultrasound

Ultrasonic range finding works by using short wavelength signals (also known as pulses) with high frequency. If there is an object in the pulse's path, it gets reflected back to the transmitter, and the distance is calculated from the time difference between the transmitted pulse and the received echo. The benefits of ultrasonic range finding are that measurements are not affected by light, the color or the transparency of an object. Sound travels nearly five times slower in air than it does in water. Because of this, ultrasonic range finding is mostly used underwater. There it can measure distances up to hundreds or even thousands of meters [30]. However, in air, it works relatively poorly compared to other methods. In air its usability is weakened by limited range and slow refresh rate, which affects its ability to detect fast-moving objects [31]. The speed of sound also varies depending on humidity and temperature, and those two attributes have to be factored in to get more accurate measurements. In air, ultrasonic range finding is commonly used in parking navigation.

3.1.2. Infrared Triangulation

In infrared triangulation, the distance to an object is calculated by using an IR (infrared) LED (light-emitting diodes) and a position-sensible photodetector. When an IR beam from the LED is focused on a surface, it reflects the beam in all directions. Position-sensible photodetector then captures this beam and calculates the distance from the angle of the reflected IR beam. Infrared triangulation solutions are usually small and lightweight, and they work in day and night environments. However, they are expensive, short-range and cannot be compounded with additional sensors. Infrared triangulation is commonly used in automatic doors. [32]

3.1.3. Distance Detection with Light

Light distance detection uses a light source with integrated optics to measure the time taken for the light to hit an object and reflect back to the sensor. Light distance detection is usually done with a powerful laser light source. Laser distance solutions are precise and can be used in long range distance detection; therefore, they are well suited for outdoor applications. Laser solutions have a high measurement frequency, which allows accurate tracking of moving targets. The drawbacks of laser distance sensors are their large size, costliness, possible harm to eyesight and the fact that they cannot be reliably integrated with additional sensors [33]. Laser distance sensors are commonly used in acquiring LiDAR data for broad areas.

3.1.4. Stereo Vision

Stereo vision is a technique that mimics the binocular vision of humans. A stereo camera usually has two lenses about 60 millimeters apart from each other, which results in slight image location disparity when viewing the same object. From this image disparity, a depth map can be calculated. Stereo cameras only require ambient light to work, which makes them great for outdoor usage in good light conditions. However, in poor light conditions or if the object has few distinctive features, acquiring the depth map is difficult. For real time applications, stereo cameras require great processing power to produce good resolution and instantaneous output. Stereo cameras usually work within a 2-meter range, but this varies and is determined by the distance between the camera's lenses. [29]

3.1.5. Time-Of-Flight

ToF (Time-of-flight) cameras calculate the round trip of the light emitted by a laser or an LED back to the sensor. Compared to stereo vision, ToF is an active technique, because it actively projects light to measure distance. ToF cameras are great solutions to use in robotics because these cameras can work in long range, they are lightweight, easy to use, compact, precise and they have multi-sensor integration. ToF's few drawbacks are that reflective surfaces can inhibit the range, outdoor performance can

be affected by direct sunlight and accuracy is not as good as with structured light technique. [34, 35]

3.1.6. Structured Light

In the structured light technique, the camera projects a modulated pattern to the surface of a scene and calculates the disparity between the original pattern and the observed pattern. Structured light is an active technique, which means that it works well in low light conditions. Structured light depth resolution can reach submillimeter level and it has higher accuracy in short range compared to the ToF method. Structured light solutions are only used indoor, because of sunlight interference to the projected light pattern [36]. The solutions can use both visible and invisible light. Invisible light is generally a better option since the projected pattern could otherwise interfere with other computer vision methods.

3.2. Depth Cameras

For accurate moving object recognition, ToF or structured light cameras are the best options. They also have other notable merits such as their availability, price, accuracy and eye safety.

3.2.1. Azure Kinect DK

In fall 2019, Microsoft released the Azure Kinect DK (development kit). Azure Kinect DK has a 1 Megapixel depth camera and a 12 Megapixel RGB camera, and it also provides a 360 degree sound pickup and an IMU (inertial measurement unit) for building advanced computer vision and speech models. Azure Kinect DK is integrated well with Azure cloud computing environment and offers an almost "plug and play" experience. Azure cloud computing environment provides speech and vision services, body tracking and sensor software development kit. Azure Kinect DK retail price is $399 and it is currently available in the United States, China, Japan, United Kingdom and Germany. [37]

3.2.2. Intel RealSense D435

The Intel RealSense D435 (Figure 6) is a depth camera solution from Intel RealSense D400 camera family. It provides high-quality depth information that can be used in a variety of applications. It also offers the widest field of view of all the Intel RealSense cameras, and it is ideal for fast moving applications thanks to its global shutter. Intel provides a RealSense SDK (software development kit) and cross-platform support, which has made the RealSense D400 camera family an excellent option for developing new products or prototypes. These cameras are used for example in robotics, drones,

3D scanning, people tracking, and in facial authentication. Intel RealSense D435 retail price is $199 and is easily available everywhere around the world. [38]

Figure 6. Intel RealSense D435 depth camera

3.2.3. Terabee

Terabee offers multiple depth camera solutions from LiDAR range finders to ToF cameras, which are commonly used by large companies in mobile robotics, IoT (Internet of things) devices, and industrial automation. Their ToF cameras offer multiple different features such as hand gesture recognition and object recognition. When compared to RealSense product family and Azure Kinect DK, Terabee cameras allow broader customization options by offering many different interchangeable interface boards for the camera. Terabee also offers a variety of different SDK's to make prototyping, implementation and developing easier. Terabee cameras are inexpensive compared to other depth cameras on the market. The cheapest Terabee camera is Teraranger Evo 64x starting at only $99; however, it has a small field of view and a low pixel count [39].

3.3. Libraries

Due to the contributions of the open-source community, many image processing libraries are available. Each of these libraries has its advantages and disadvantages. In this section, three commonly used libraries and one highly optimized library for 3D image processing are introduced and compared.

3.3.1. Point Cloud Library

PCL (Point Cloud Library) is a massive open-source library used for 2D and 3D image processing. The PCL framework offers multiple different features for point cloud data processing, for example, filtering, stitching multiple point clouds together, segmenting, object recognition based on the object's geometric appearance, and visualization. Even though it supports 2D image processing as well, it is mainly focused on 3D image processing [40, 41, 42].

3.3.2. OpenCV

OpenCV (Open source computer vision) library is a large open source project dedicated to 2D/3D real-time computer vision. OpenCV has currently the most comprehensive selection of algorithms compared to other available image processing libraries. The library contains more than 2500 algorithms, and these can be used, for example, to detect and recognize faces, identify objects, extract 3D models and produce 3D point cloud from stereo cameras, among other things. OpenCV's 3D usage is fairly limited and it is mainly used to transform 2D images into 3D point clouds[43].

3.3.3. Open3D

Open3D is one of the most modern open source 3D point cloud data processing libraries. It currently has fewer algorithms and tools to use than PCL. However, it is designed and developed to be highly optimized, take advantage of parallelization, and support rapid development of software that processes 3D data[40, 44].

3.3.4. Cilantro

Cilantro is currently the most barebone and fastest open source library available, that is designed to process point cloud data. However, Cilantro has rather limited documentation and due to its lean design, it does not support visualization [40].

4. IMPLEMENTATION

This work introduces a real-time program to segment people from depth frames. This program will then later be integrated as a part of a service robot, to provide information to the robot about the people around it. The service robot is going to operate in an open environment of which an example can be seen in Figure 7. Since the environment is relatively open and the robot's view is mostly going to be full of people instead of random objects, an AI based detection verification step can be omitted and the main focus of this work is on fast object separation and human detection.

PCL was selected as the depth data processing library, due to its large number of already implemented algorithms, flexibility and support for point cloud processing.

Intel RealSense Depth Camera D435 was chosen for this project, due to its affordable price, availability and compatibility with the chosen data processing library (PCL). The D435 camera provides both RGB and depth frames, which could both be used for human detection. In this work, only the depth frames were researched because the amount of research done on human detection from just the depth data does not compare to the amount of research done on human detection from RGB and RGB-D (red, green, blue and depth) images.

Figure 7. Example environment

4.1. Data Pipeline

The data was processed with various different techniques to find the optimal people segmentation method for this project. This resulted in the data processing pipeline shown in Figure 8. Other methods were also evaluated, but their disadvantages were too great compared to their benefits. For example, taking a reference frame of a people-free environment and then comparing other frames to the reference frame provided very fast and accurate results. However, this method was not suitable for this project,

since it would have meant taking a new reference frame of the people-free environment every time the camera moved.

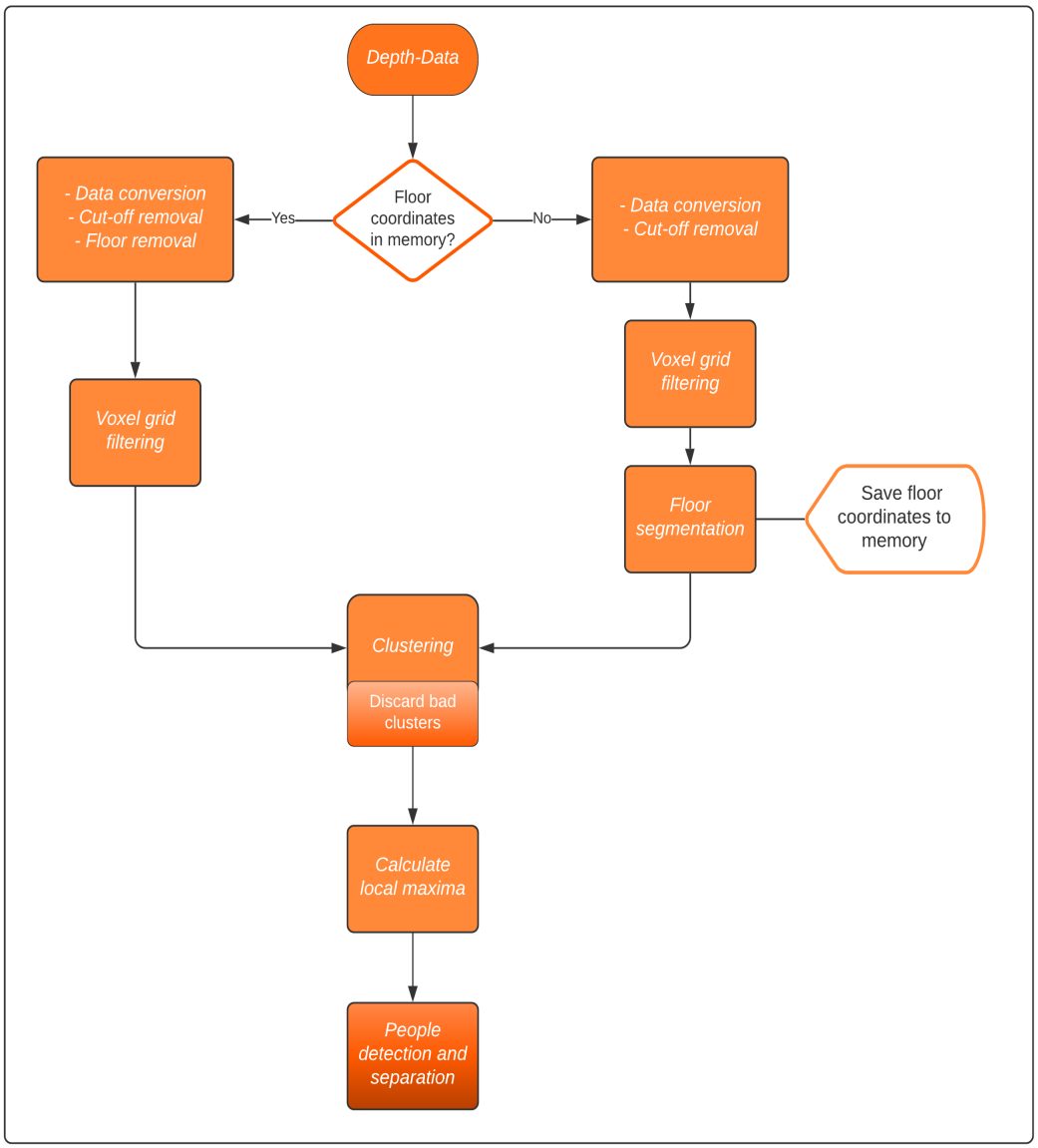

Figure 8. Data pipeline for chosen method

4.1.1. Input Data

The D435 camera allows depth streams in various resolutions and frame rates as seen in Table 2. In this work, depth frames were processed at 30 FPS (frames per second) and with a 640x480 resolution. Choosing this resolution over the highest possible one, already cuts down the amount of points in the point cloud to a one third. The raw depth values obtained from the D435 depth camera need to be scaled by a constant to obtain the distance in meters. In addition, to apply the upcoming processing steps, the metric horizontal and vertical distances need to be calculated.

Table 2. Vision Processor D4 Depth Data Stream[6]

Depth-Frame			
USB 3.1 Gen1		USB 2.0	
Resolution	Frame Rate (FPS)	Resolution	Frame Rate (FPS)
1280x720	6, 15, 30	1280x720	6
848x480	6, 15, 30, 60, 90	————	————
640x480	6, 15, 30, 60, 90	640x480	6, 15, 30
640x360	6, 15, 30, 60, 90	————	————
480x270	6, 15, 30, 60, 90	480x270	6, 15, 30, 60
424x240	6, 15, 30, 60, 90	————	————

Although the Intel RealSense SDK provides a function to convert X and Y pixel values into real world coordinates, in this work a custom conversion function was implemented, where the metric distances Pz, Px and Py of a particular point P are computed with:

$$Pz = Dx, y \cdot Scale \tag{1}$$

$$Px = Pz \cdot \frac{x - \frac{1}{2}w}{fx} \tag{2}$$

$$Py = Pz \cdot \frac{y - \frac{1}{2}h}{fy} \tag{3}$$

Where Dx, y is a raw distance at specific pixel location in the image, $Scale$ is a depth scale that converts the raw distance to meters, x and y denote the pixel location in the image, w and h are the image width and height in pixels, and fx and fy are intrinsic parameters of the camera. The resulting input point cloud can be seen in Figure 9 and the corresponding RGB frame on Figure 10.

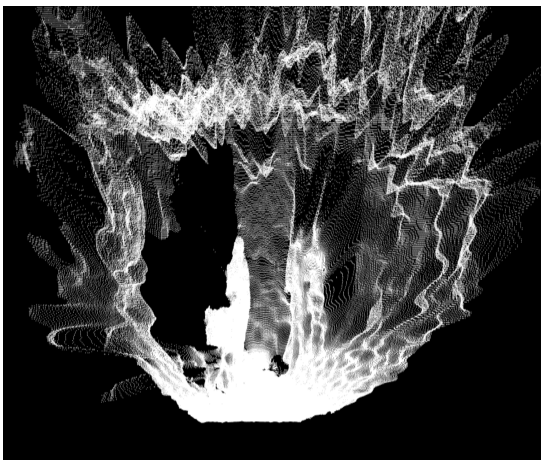

Figure 9. Point cloud after conversion

Figure 10. Input RGB image

[6]Data adabted from Intel RealSense D400 Series Product Family Datasheet: https://www.intelrealsense.com/wp-content/uploads/2019/10/Intel-RealSense-D400-Series-Datasheet-Oct-2019.pdf?_ga=2.2100848.1508440573.1585647177-331894120.1584636039

The custom function reduced the conversion time to around one third (from 16.66ms to 5.45ms as seen in Appendix 2) and provides more flexibility for this work. For example, the upcoming cut-off step can be already performed in the conversion phase and thus, even more processing time can be saved. The difference in speed might not be related to the math itself. The real reason might be that in the custom function, the data is read and processed straight from the raw data frame, instead of using pre-built data structures for the frames and their *get*-methods, which are somewhat time-consuming methods with large amounts of data points.

4.1.2. Filtering

With huge amounts of data, the processing time quickly becomes the bottleneck in the program. In this project especially, having the program run in real time is essential because the robot's movements have to be responsive. This problem can be alleviated with data filtering. In short, data filtering is the process of reducing the number of data points. The goal of data filtering is to have a data set as small as possible, that still represents the original input. This is an important step in any data processing pipeline that deals with large amounts of data.

Choosing the correct filtering method or methods is crucial and should be done case by case. In this project, data was filtered with voxel grid filtering, due to its ability to maintain the shape of the original point cloud. Voxel grid filtering creates a 3D voxel grid over the input data. Within each voxel, all the points are combined into one point by calculating the average of the points. This process drastically reduces the number of points in the point cloud, but still represents the underlying surface accurately. This accurate representation of the surface is important for future steps and it also helps to reduce the processing time because the planar component (the floor) can be separated with less effort.

With the voxel grid filter, the number of data points was decreased by around 95%. This reduction of points made the algorithm run slightly faster; however, it was still taking a considerable amount of time. After some investigation, it was noticed that the data from the D435 camera was very noisy and distorted around objects that were far away. More research on this issue was done, and it was noticed that the further the object is, the more distorted it will be. Objects, people and even the roof were stretched out. This caused problems with the upcoming floor segmentation because fitting a planar model into a noisy point cloud turned out to be difficult and slow. In Figures 11, 12, 13 and 14 the distortion is illustrated. Especially on Figure 14, where two people are standing at 2m and 6m distance away from the camera, the distortion can clearly be seen.

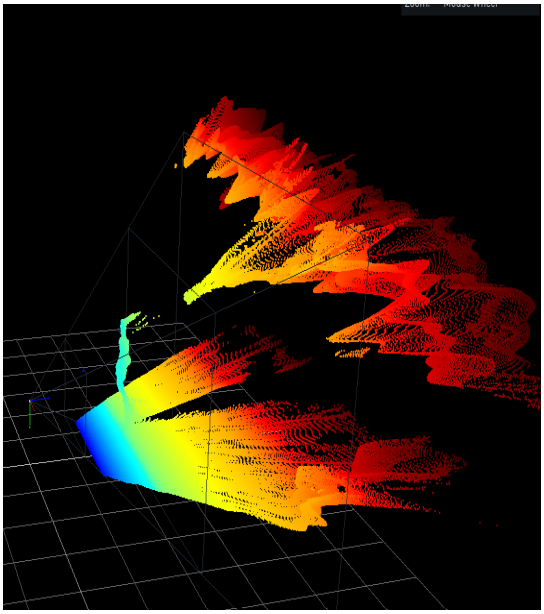

Figure 11. Person at 2m distance

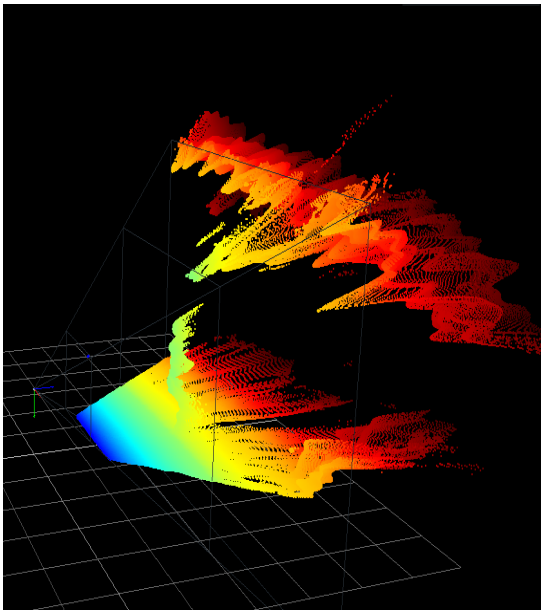

Figure 12. Person at 3m distance

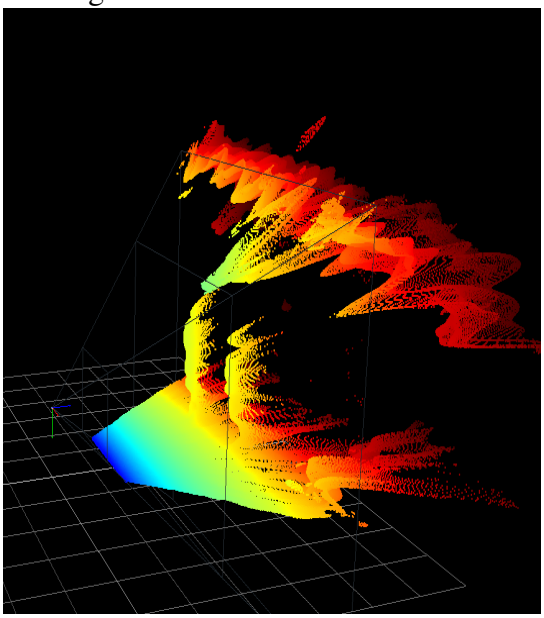

Figure 13. People at 4m distance

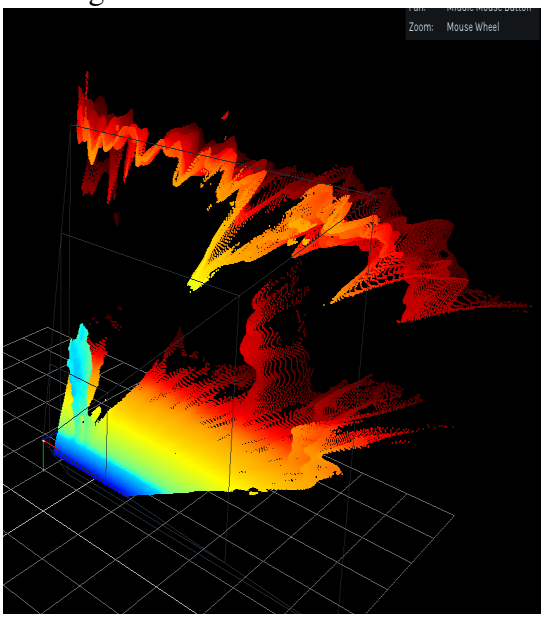

Figure 14. People at 2m and 6m distance

To counter this problem, a cut-off distance (meaning a distance after which the points of the cloud were discarded) of 6.0m was implemented. The distortion at this distance remained manageable and would allow the robot to see far enough. It has to be kept in mind that the cut-off distance was chosen specifically for this environment and can be adjusted accordingly. The resulting point cloud after each filtering step is applied can be seen in Figures 15 and 16.

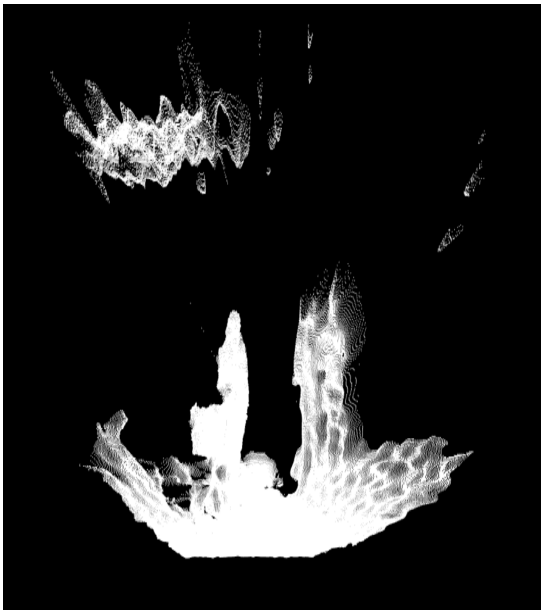

Figure 15. Result after cut-off filter

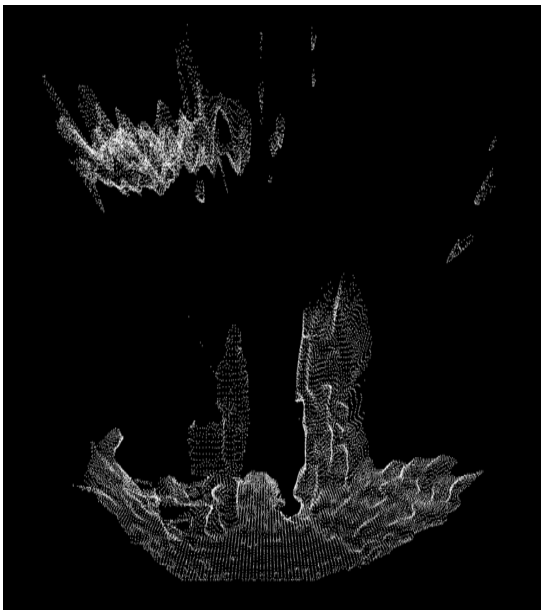

Figure 16. Result after voxel grid filter

4.1.3. Floor Segmentation

At this phase, separating people from the point cloud using the euclidean clustering algorithm would be impossible. Currently, everything that is connected to the floor would be considered to be in the same cluster. For example, the results of the clustering algorithm on the frame shown in Figure 16, would result in two clusters circled in red and green in Figure 17. From this, one can see that if the people in the frame were also flying like the noise cluster, they could also be separated. Unfortunately, every person in the view of the camera is probably not going to be flying around with a jet pack, ergo another processing step is required.

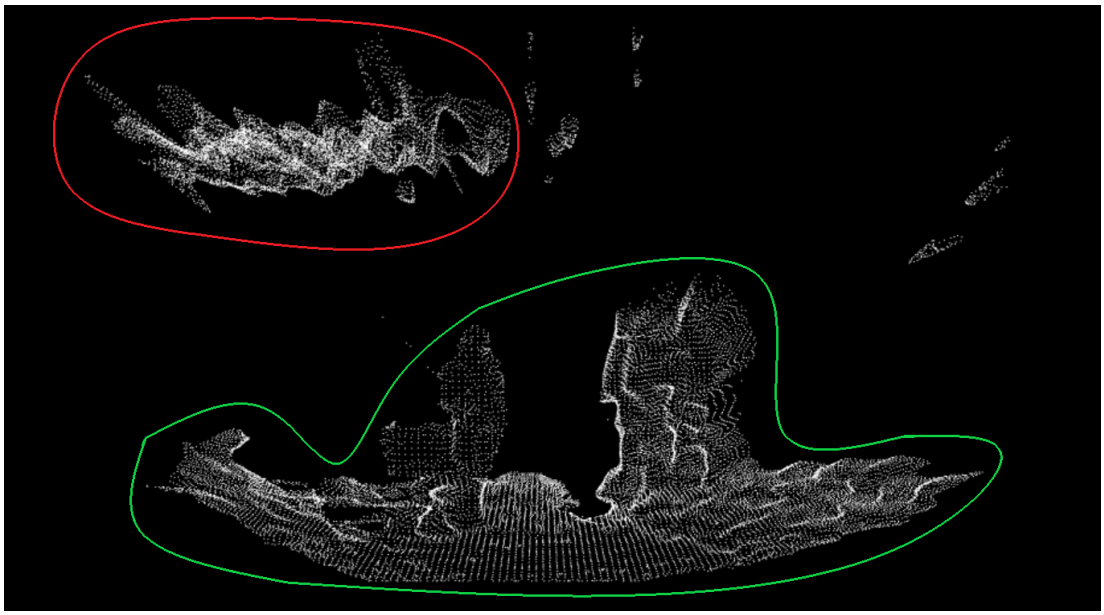

Figure 17. Clustering result before floor segmentation

The required processing step needs to remove the connection between different people and objects. This means that the floor needs to be removed. Because the floor is a flat horizontal 3D plane, it can be removed using RANSAC (random sample consensus). The RANSAC algorithm tries to fit a model into the data and group the data into inliers and outliers. Inliers are points within the model and outliers are points outside the model. In this case, the problem is solved by fitting a horizontal plane model into the point cloud and then removing the inliers, that is the points inside the horizontal plane. When fitting the horizontal plane, a ± 30 degree rotation is allowed to account for slight changes in the camera angle.

After running the main part of the floor segmentation algorithm, there is still a lot of noise left around the floor plane and way above it, as can be seen in Figure 18. However, now that the floor's coordinates are known, they can be used as a reference to know at what level people are supposed to be. This means that points near the floor plane and high above the floor plane can be removed. The better filtered result of the floor removal can be seen in Figure 19.

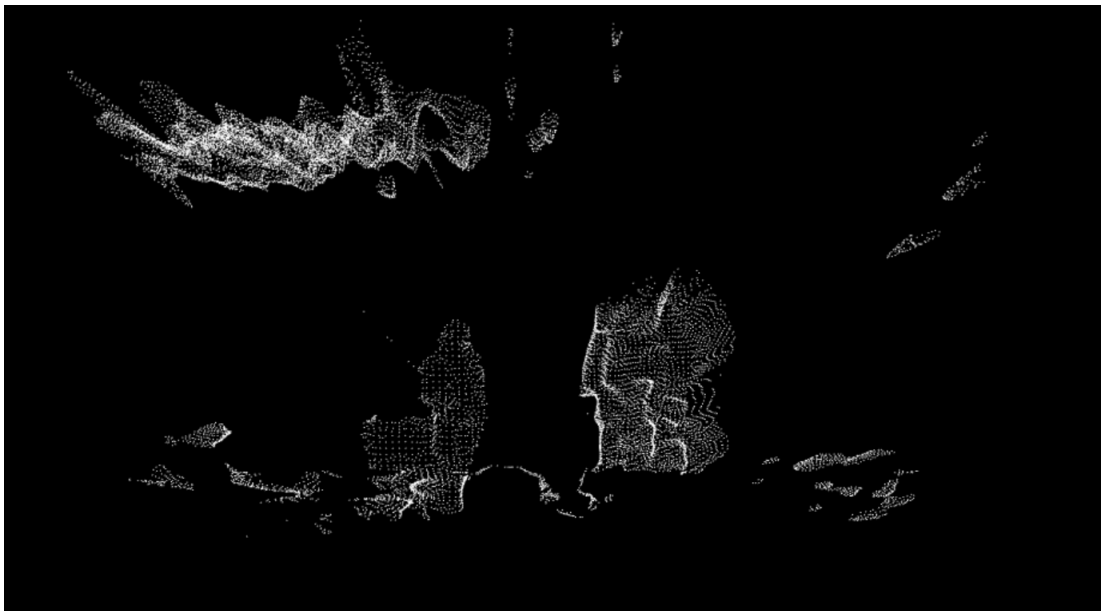

Figure 18. Result after segmenting the floor plane

The floor segmentation algorithm is the slowest part of the program (39.01ms as seen in Appendix 2) but counteracting this speed problem is fairly simple. The program only needs to run this algorithm if the camera is moved in the Y direction, because only then do the floor's coordinates change in relation to the camera. After running the algorithm once, the floor plane's coordinates are saved. When the next frame arrives, the floor plane can be removed without using the algorithm. This saves a lot of computation time because the floor plane can be removed at the same time the cut-off is applied. Furthermore, both of these and the additional filtering after the floor removal are implemented into the data conversion function, which means the data needs to be looped through only once to achieve the results seen in Figure 19.

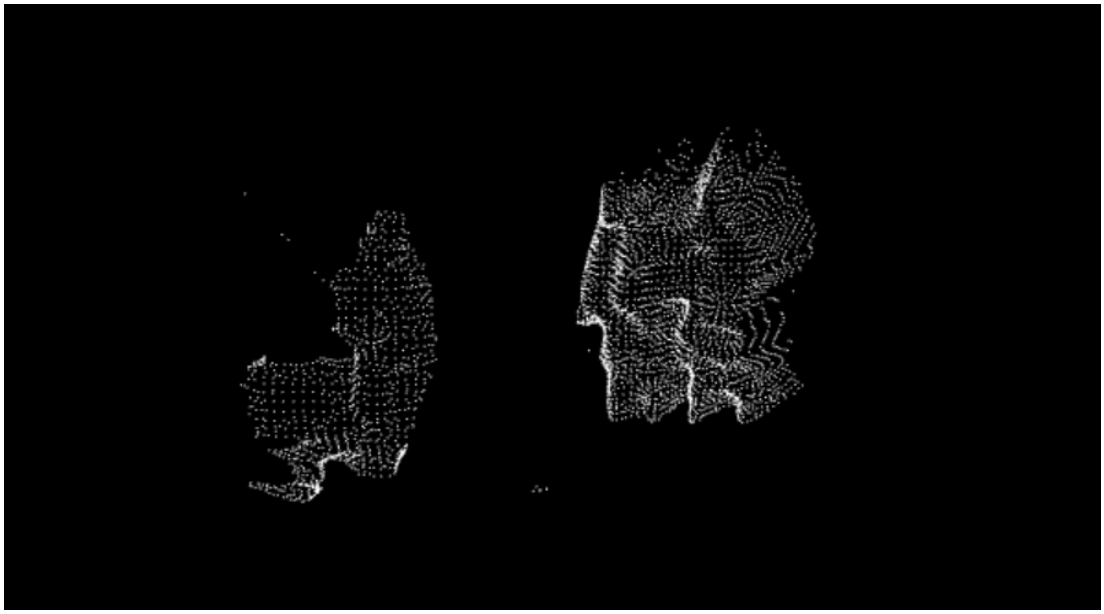

Figure 19. Result after floor segmentation and additional height-based filtering

4.1.4. Clustering

Now, as can be seen from Figure 19, the point cloud clearly consists of separate objects not connected by anything, but they are still all contained within the same point cloud. To be able to detect people and further parse the data, the clusters need to be extracted from the point cloud.

To solve this problem, a simple clustering algorithm based on euclidean distances between points is applied to the point cloud. Euclidean distance clustering simply groups points together within a certain distance threshold. This approach works well for the input point cloud because the floor plane has been removed and none of the objects are connected by anything. By setting certain requirements shown in Appendix 1 for these clusters, a lot of the noise can also be removed. These requirements include the minimum and maximum height of the cluster and the minimum number of points within the cluster.

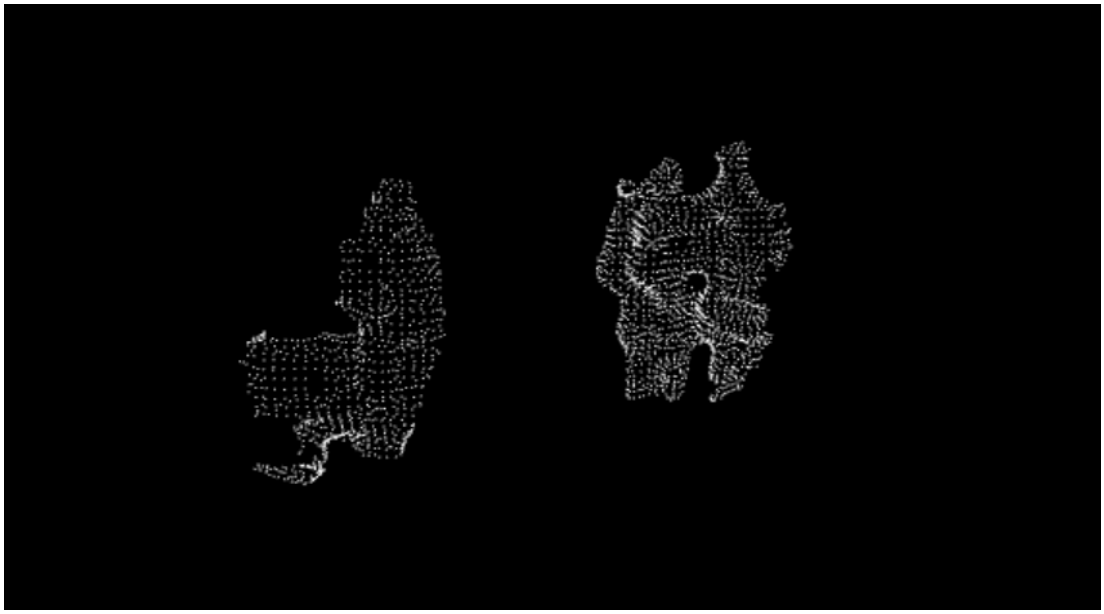

Figure 20. Result after euclidean distance clustering

As seen in Figure 20, the noise is now gone and all that is left is two clusters. One containing one person and one containing two people. Further processing steps can now be easily applied to each cluster.

4.1.5. Sub-Clustering Groups of People

Two or more people can be merged into the same cluster because they are too close to each other as seen on Figure 20. People can also be clustered in with objects next to them. This merging is further enhanced because of the camera noise, and needs to be dealt with in the data processing phase.

Separating multiple people from one cluster can be achieved by detecting the heads of the people in the cluster, since there is a one-to-one person-head correspondence. The head is usually the highest part of the human body, and a certain "uphill/downhill" pattern can be distinguished from the point cloud. By detecting these mountain tops, so to speak, a sub-clustering algorithm can be implemented.

Using the algorithm presented in [45], a sub-clustering method that detects people from a 3D point cloud and segments people into sub-clusters based on the head positions was implemented. The algorithm goes as follows:

1. Divide each cluster into bins, each containing 100 points.
2. Find the highest point in each bin that is higher than the height threshold (1.3m) from the plane floor.
3. From those points, calculate the local maxima.
4. Create a sub-cluster for each maximum by taking all the points within 15cm to its left and 15cm to its right. This creates a 30cm wide sub-cluster.

The above-mentioned algorithm reduces the possibility of a chair or a table being clustered in with a person. The 30cm width of a person is an arbitrary number based on the idea that people's heads are usually not closer than that to each other. In the paper [45], 30cm was also used, but for discarding the local maxima within 30cm euclidean distance of each other. This discarding of other maxima is not implemented in this project, because after testing, separating the data into bins and then calculating the local maxima was concluded sufficient enough to remove any false head detections. The resulting sub-clustered cloud can be seen in Figure 21, and clusters can now be used to make decisions on where to turn the robot's head.

Figure 21. Result after sub-clustering

5. FUTURE WORK

Even though the proposed algorithm works well in an open environment, there is still much that can be improved to get more reliable results and a more responsive robot, once the program is integrated into one. This can be achieved by either working on improving the program to further decrease the processing time or taking advantage of AI to verify that the detections from the algorithm are indeed human.

The processing time could be improved by using a more powerful library like Cilantro or by implementing all the algorithms used from scratch. The latter method is probably better in the long term, since it allows designing the algorithms to perfectly fit the program. Although improvements in processing time can be beneficial, the achieved processing time with the current implementation is adequate; therefore, the main focus of future work should be in improving the accuracy of the detections.

Because the processed data is available in real time, taking advantage of AI to verify the detections should be the clear next step to take for future improvements. There are two possible ways to start implementing the AI: either doing human detection on the separated point cloud or doing it on the corresponding area of the RGB image. If the working environment of the service robot is considered, projecting the separated 3D point cloud clusters into a 2D plane, and then detecting if the shape corresponds to the shape of a human, could be a fast and effective solution. This projection would decrease the distortion of the human body illustrated in Figures 11, 12, 13 and 14. However, in good lighting conditions, using the area of the RGB images that corresponds to the area of the separated point cloud for correct detection verification will probably yield more accurate results.

6. TIME USAGE

The work for this project was mostly done as a group and partly individually. Usually, the group scheduled 3 workdays per week for this project. Most of the group work was done remotely. Table 3 represents the time used by each group member in this project.

Table 3. Time usage by each group member

Name	Hours
Miiro Kuosmanen	210
Samu Majabacka	215
Kalle-Oskari Suvanto	210

7. SUMMARY

Thanks to the continuous development of technology and robotics, service robots are becoming more commonplace and can already be found in many households. This means that the number of robots working among humans is constantly increasing, and therefore, their safety for people is becoming a critical factor. The safety of the robots can be improved by giving them a better understanding about their environment, which can be achieved with the help of different camera technologies and artificial intelligence.

In this project, an algorithm capable of distinguishing people from depth data obtained from a depth camera was implemented. The algorithm works by processing the depth data in multiple steps, where the first steps are focused on noise and data reduction. This speeds up the algorithm considerably and allows real-time data processing. After this, through various other stages and algorithms, people are separated from the environment by mainly using naive and heuristics methods.

The developed algorithm worked admirably well in real time, which was one of the purposes of the algorithm. The proposed algorithm could be used in various situations, for example giving a service robot information about the people around it and as a pre-processing phase for AI solutions. However, there is a lot of room for improvement in the algorithm as it only works in a relatively open environment. Also, the algorithm uses heuristics for human detection, so this aspect could also be significantly improved with the help of artificial intelligence.

8. REFERENCES

[1] Moore G.E. (2006) Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. IEEE Solid-State Circuits Society Newsletter 11, pp. 33–35.

[2] Murray Campbell A.Joseph Hoane Jr F.h.H. (2002) Deep Blue. Artificial Intelligence 134, pp. 57–83. URL: https://www.sciencedirect.com/science/article/pii/S0004370201001291?via%3Dihub.

[3] Granter S.R., Beck A.H. & Papke D.J. (2017) Alphago, deep learning, and the future of the human microscopist. Archives of Pathology & Laboratory Medicine 141, pp. 619–621. URL: https://doi.org/10.5858/arpa.2016-0471-ED, pMID: 28447900.

[4] Illustrated London News - Saturday 15 September 1928. URL: https://www.britishnewspaperarchive.co.uk/viewer/BL/0001578/19280915/035/0011.

[5] Guizzo E. (2019) By leaps and bounds: An exclusive look at how boston dynamics is redefining robot agility. IEEE Spectrum 56, pp. 34–39.

[6] (2020), Greek mythology. URL: https://www.greekmythology.com/Myths/Creatures/Talos/talos.html.

[7] Rosheim M. (2006) Leonardo's Lost Robots. Springer, 69 p.

[8] (2012) robot, n.1. Oxford English Dictionary. URL: https://www.oed.com/view/Entry/275486?rskey=vPXfO0&result=1.

[9] O'Regan G. (2015) Unimation, Springer International Publishing, Cham. pp. 219–223. URL: https://doi.org/10.1007/978-3-319-21464-1_34.

[10] Müller D.C. (2018) Ifr press conference. IFR International federation of robotics 28. URL: https://ifr.org/downloads/press2018/IFR%20World%20Robotics%20Presentation%20-%2018%20Sept%202019.pdf.

[11] Sprenger M. & Mettler T. (2015) Service robots. Business & Information Systems Engineering 57.

[12] (2012) Robots and robotic devices – Vocabulary. Standard, International Organization for Standardization, Geneva, CH.

[13] Park E., Kong H., Lim H., Lee J., You S. & del Pobil A.P. (2011) The effect of robot's behavior vs. appearance on communication with humans. In: 2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 219–220.

[14] Goetz J., Kiesler S. & Powers A. (2003) Matching robot appearance and behavior to tasks to improve human-robot cooperation. In: The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003., pp. 55–60.

[15] Cooney M. & M. Karlsson S. (2015) Impressions of size-changing in a companion robot. In: Proceedings of the 2nd International Conference on Physiological Computing Systems, PhyCS 2015, SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT, p. 118–123. URL: https://doi.org/10.5220/0005328801180123.

[16] Huang C. & Mutlu B. (2012) Robot behavior toolkit: Generating effective social behaviors for robots. In: 2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 25–32.

[17] Mathur M.B. & Reichling D.B. (2016) Navigating a social world with robot partners: A quantitative cartography of the uncanny valley. Cognition 146, pp. 22 – 32. URL: http://www.sciencedirect.com/science/article/pii/S0010027715300640.

[18] Wood L.J., Zaraki A., Robins B. & Dautenhahn K. (2019) Developing kaspar: A humanoid robot for children with autism. International Journal of Social Robotics URL: https://doi.org/10.1007/s12369-019-00563-6.

[19] Cheang H.S. & Pell M.D. (2008) The sound of sarcasm. Speech Communication 50, pp. 366 – 381. URL: http://www.sciencedirect.com/science/article/pii/S0167639307001884.

[20] Wu Y.C. & Coulson S. (2007) How iconic gestures enhance communication: An erp study. Brain and Language 101, pp. 234 – 245. URL: http://www.sciencedirect.com/science/article/pii/S0093934X0600438X, gesture, Brain, and Language.

[21] Obermeier C., Dolk T. & Gunter T.C. (2012) The benefit of gestures during communication: Evidence from hearing and hearing-impaired individuals. Cortex 48, pp. 857 – 870. URL: http://www.sciencedirect.com/science/article/pii/S0010945211000323, language and the Motor System.

[22] Cooney M. & Sant'Anna A. (2017) Avoiding playfulness gone wrong: Exploring multi-objective reaching motion generation in a social robot. International Journal of Social Robotics 9, pp. 545–562. URL: https://doi.org/10.1007/s12369-017-0411-1.

[23] Cary M.S. (1978) The role of gaze in the initiation of conversation. Social Psychology 41, pp. 269–271. URL: http://www.jstor.org/stable/3033565.

[24] Lee D.H. & Anderson A.K. (2017) Reading what the mind thinks from how the eye sees. Psychological Science 28, pp. 494–503. URL: https://doi.org/10.1177/0956797616687364, pMID: 28406382.

[25] Hanna J.E. & Brennan S.E. (2007) Speakers' eye gaze disambiguates referring expressions early during face-to-face conversation. Journal of Memory and Language 57, pp. 596 – 615. URL: http://www.sciencedirect.com/science/article/pii/S0749596X07000174, language-Vision Interaction.

[26] Taigman Y., Yang M., Ranzato M. & Wolf L. (2014) Deepface: Closing the gap to human-level performance in face verification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[27] Song S., Zhang L. & Xiao J. (2015) Robot in a room: Toward perfect object recognition in closed environments. CoRR, abs/1507.02703 .

[28] Munaro M. & Menegatti E. (2014) Fast rgb-d people tracking for service robots. Autonomous Robots 37.

[29] Jiejie Zhu, Liang Wang, Ruigang Yang & Davis J. (2008) Fusion of time-of-flight depth and stereo for high accuracy depth maps. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8.

[30] (1946) The Radar of the Deep. Popular Science, 84-87 p. URL: https://books.google.fi/books?id=bCEDAAAAMBAJ&lpg=PA84&dq=popular%20science%20July%201946&pg=PP1#v=onepage&q&f=true.

[31] Moravec H. & Elfes A. (1985) High resolution maps from wide angle sonar. In: Proceedings. 1985 IEEE International Conference on Robotics and Automation, vol. 2, vol. 2, pp. 116–121.

[32] Pastorius W. (2013) Triangulation sensors.

[33] NOOA (2013), Lidar—light detection and ranging—is a remote sensing method used to examine the surface of the earth. URL: https://oceanservice.noaa.gov/facts/lidar.html, accessed 20.1.2013.

[34] Cui Y., Schuon S., Chan D., Thrun S. & Theobalt C. (2010) 3d shape scanning with a time-of-flight camera. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1173–1180.

[35] Ringbeck D.I.T. (2007) A 3 d time of flight camera for object detection.

[36] Scharstein D. & Szeliski R. (2003) High-accuracy stereo depth maps using structured light. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., vol. 1, vol. 1, pp. I–I.

[37] About azure kinect dk. https://docs.microsoft.com/en-us/azure/kinect-dk/about-azure-kinect-dk. Accessed: 2020-18-2.

[38] Intel realsense d435. https://www.intelrealsense.com/depth-camera-d435/. Accessed: 2020-18-2.

[39] Terabee. https://www.terabee.com/sensors-modules/. Accessed: 2020-18-2.

[40] Zampogiannis K., Fermüller C. & Aloimonos Y. (2018) cilantro: A lean, versatile, and efficient library for point cloud data processing. pp. 1364–1367.

[41] Rusu R.B. & Cousins S. (2011) 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China.

[42] Rusu R.B. & Cousins S. (2011) 3d is here: Point cloud library (pcl). In: 2011 IEEE International Conference on Robotics and Automation, pp. 1–4.

[43] Bradski G. (2000) The OpenCV Library. Dr. Dobb's Journal of Software Tools .

[44] Zhou Q.Y., Park J. & Koltun V. (2018) Open3D: A modern library for 3D data processing. arXiv:1801.09847 .

[45] Munaro M., Basso F. & Menegatti E. (2012) Tracking people within groups with rgb-d data. pp. 2101–2107.

9. APPENDIX

Appendix 1. Variables

Local maximum max height	2.3m
Local maximum min height	1.3m
Bin size	100 points
Voxel leaf size	0.05m
RANSAC distance threshold	0.10m
RANSAC maximum allowed plane rotation[7]	± 30 degrees
Euclidean cluster tolerance	0.05m
minimum cluster size	100 points
cluster max height	2.3m
cluster min height	1.3m
cut-off distance	6.0m

Appendix 2. Processing times

Intel RealSense SDK conversion function	16.66 ms
Custom conversion	5.45 ms
Floor removal	39.01 ms
Voxel grid filter	4.77 ms
Clustering	17.91 ms
Sub-clustering	0.54 ms
Total time per frame[8]	28.67 ms

Times are averages of 100 processed frames with off-the-self laptop. Its CPU and RAM can be seen in Appendix 3.

Appendix 3. Computer specifications

CPU	Intel i5-7200U @ 2.50GHz
RAM	DDR4 2133MHz 16.0GB

[7]The fitted model was a horizontal plane

[8]After running floor removal once, only the New conversion, Voxel grid filtering, clustering and sub-clustering steps need to be run