Master of Science in Informatics at Grenoble
Master Informatique
Specialization Graphics, Vision and Robotics

Detection, Classification and Tracking of Persons with a 3D LiDar sensor

Juan Gomez

June 13, 2021

Research project performed at LIG - Marvin

Under the supervision of:
Olivier Aycard

Defended before a jury composed of:
Dominique Vaufreydaz
Renaud Blanch
Michele Rombaut

June 2021

Abstract

In this work, a method for detecting, classifying and tracking persons in 3D point cloud data in an office environment is presented. This is an important aspect in applications such as social robotics and autonomous driving. Other applications such as surveillance and understanding people behavior or human-robot collaboration in industries are also pertinent. In our solution, we present a modular approach that relies on combining multiple principles: a robust implementation for voxelization of points and object segmentation, a simple classifier with local geometric descriptors and a Global Nearest Neighbor solution for tracking. Further optimization and enhancement could be done easily thanks to its modular nature. Moreover, the main merit of this work relies in achieving a real time solution in a low-performance machine, which is done by reducing the amount of points to be processed at all times by obtaining and predicting regions of interest via movement detection and motion prediction without any previous knowledge of the environment. Furthermore, even though our geometric classifier is simple, our prototype is able to successfully detect and track persons consistently even in cases of extreme pose changes like crouching, jumping and stretching. It also handles targets appearing partially cropped when standing near to the sensor due to its narrow vertical field of view of 33 degrees. This is done using the module for tracking to predict the target's location and therefore correct miss classifications. Lastly, the proposed solution is tested and evaluated in a real 3D LiDar sensor test set taken in our office and shows great potential for this proof of concept. Particularly, the results show a high confidence for positive classifications in not too cluttered environments.

Acknowledgment

I would like to express my sincere gratitude to my supervisor Olivier Aycard for his invaluable assistance and comments in reviewing this report and throughout the complete process to implement the presented solution.

Résumé

Dans ce travail, une méthode de détection, de classification et de suivi de personnes dans des données de nuages de points 3D dans un environnement de bureau est présentée. C'est un aspect important dans des applications telles que la robotique sociale et la conduite autonome. D'autres applications telles que la surveillance et la compréhension du comportement des personnes ou la collaboration homme-robot dans les industries sont également pertinentes. Dans notre solution, nous présentons une approche modulaire qui repose sur la combinaison de plusieurs principes : une implémentation robuste pour la voxélisation des points et la segmentation des objets, un classificateur simple avec des descripteurs géométriques locaux et une solution Global Nearest Neighbour pour le suivi. Une optimisation et une amélioration supplémentaires pourraient être effectuées facilement grâce à sa nature modulaire. De plus, le principal mérite de ce travail réside dans la réalisation d'une solution en temps réel sur une machine peu performante, ce qui

se fait en réduisant le nombre de points à traiter à tout moment en obtenant et en prédisant les régions d'intérêt via la détection de mouvement et la prédiction de mouvement. sans aucune connaissance préalable de l'environnement. D'ailleurs, même si notre classificateur géométrique est simple, notre prototype est capable de détecter et de suivre avec succès les personnes de manière cohérente, même en cas de changements de pose extrêmes comme s'accroupir, sauter et s'étirer. Il gère également les cibles semblant partiellement recadrées lorsqu'elles se tiennent près du capteur en raison de son ouverture verticale étroite de 33 degrés. Cela se fait à l'aide du module de suivi pour prédire l'emplacement de la cible et donc corriger les classifications des ratés. Enfin, la solution proposée est testée et évaluée dans un jeu de données réel pris avec notre capteur LiDar 3D dans notre bureau, et montre un grand potentiel pour cette preuve de concept. En particulier, les résultats montrent une confiance élevée pour les classifications positives dans des environnements pas trop encombrés.

Contents

List of Figures

— 1 —
Introduction

In the field of robot perception, detection and tracking of moving objects (DATMO) is one of the main problems in almost every robotics application. Now more than ever, several autonomous robotic applications require a robust perception module to understand the environment the robot is in. Applications like autonomous or assisted driving, environment understanding for social robots and even human behavior understanding for commercial or scientific purposes are among them. In this work, we are focused on detection and tracking of moving persons in a static environment.
The algorithm is intended for a mobile robot in an office-type environment for social robot applications such as person following, etc. The extent of the work is to successfully detect, classify and track moving persons with a static 3D LiDar sensor.
A lot of previous work has been done to tackle this particular problem in perception, most of them with classic sensors such as cameras, radar and even 2D LiDar range sensors as they became very popular due to their affordability. Even though they are affordable and their data is not computationally expensive to process, the drawbacks are evident in the overall final performance and evaluation of a system due to multiple constraints in the collected data. These sensors do a scan of the environment along only one plane, so any object outside this plane might not be detected. Many applications try to overcome this issue by placing multiple 2D LiDar sensors at different heights of the robot, depending on the target object to be detected the position of the sensors can be key to obtain an accurate representation of them. Despite this, a single row of points at each height is still often not sufficient for classification purposes, because the geometric features that can be extracted from them are limited. These issues might lead to a considerable amount of false positives or false negatives. Other approaches in the literature have used simple solutions such as stating that if a small cluster of points are moving they could belong to a moving person, which results in the same type of problems previously described.
On the other hand, RGB cameras might struggle to work correctly under precarious lighting conditions as is the case when the application is in autonomous vehicles or assisted driving and obstacle avoidance.
LiDar 3D sensors eliminate every problem previously mentioned in other sensors, but come with their challenges as well. The most immediate challenge is the amount of data provided by each scan of the sensor. In the case of the Ouster OS1-32, the number of points for each scan could go from 16384 to 65536 points depending on the horizontal resolution the sensor is configured in. Therefore, one scan of 3D LiDar is several magnitudes larger than one of a 2D one. In addition, geometrical interpretation with raw data could prove to be very difficult due

to the high amount of sparse points. For these reasons, fewer work with 3D LiDar has been done to this moment, because the majority of the applications for DATMO can not work in real time due to the high volume of data that have to be processed at each scan.
Most of these sensors have enough resolution to appropriately display objects in 3D, but to understand the geometry of the environment the points have to first be clustered and segmented into objects correctly. Then, if the object is within the sensor minimum and maximum range of detection, the results would be well defined objects with geometric properties that align with preconceived assumptions of several classes of objects. This means that objects in 3D LiDar reading can be described geometrically as they appear in the real world, with accurate representations for length, width and height up to a certain precision. Besides this, 3D LiDar sensors are not affected by extreme lighting conditions as they depend on time of flight calculation just as their 2D counterpart.
In this work we present a real time solution for multiple person detection, classification and tracking. The former is achieved by using techniques normally used in classic computer vision such as movement detection, region of interest and tracking in addition to classic techniques used in 3D point cloud non real time applications such as voxelization and segmentation. The two of which combined can result in a real time application.
The algorithm can be separated in the following sections: Voxelization, Segmentation, Movement Detection, Classification and Tracking. The focus of our work is to always process the least amount of points possible whilst never leaving a possible region of interest unprocessed.
The basis of the operations is the voxelization, as all of the other parts depend on it. The voxelization is done with a k-nn search that allows for the reduction of the total number of points to be processed.
Movement detection in sparse point cloud data can prove to be difficult due to the nature of 3D LiDar sensors. Every time the sensor does a 360 degree scan the exact point position can vary scan to scan. The former combined with possible noise data that can appear in some scans create inconsistencies between scans that should not be attributed to movement detection. This is why the movement detection is done using voxel comparison and not point to point comparison. In the movement detection stage, the voxels are computed only once and then updated with each new scan. In this way we can correctly identify new voxels that might belong to a moving object.
Segmentation is done mainly according to distance between voxels but also according to voxels parameters such as intensity, normals, etc. After segmentation, we might go from having to process 66000 points to dealing with only 10-15 objects depending on how cluttered the environment is. This method was proposed in the literature [2] but for non real time applications, as the process can have a great impact in performance.
Classification for 3D point clouds have already been studied before, with a lot of deep learning approaches dominating the research community but mainly for one-scan multi-class classification. In our case, classification is done according to geometrical features attributed to each object, like shape and the normal vectors. Even though classification with deep learning methods have proved to be effective, it remains computationally expensive to do for real time applications in DATMO.
Each of these techniques, working in combination with a GNN approach for tracking, results in a real time solution for our problem without having any prior knowledge on the environment. As simple or as computationally expensive as some of the techniques can be on their own, they complement each other in order to accomplish our real time goal.

Details on the conception and implementation of each part is explained in detail in the Methodology section of the report.

To summarize, the main contributions of our solution are:

- It is structured in a modular approach consisting of multiple stages that work together to achieve the best performance and results. The modular nature makes it easier to enhance the performance of the solution by improving any of the modules presented. Among these modules we can find: Voxelization, Segmentation, Movement Detection, Classification and Tracking.
- A real-time implementation on a low-performance machine for detecting, classifying and tracking persons in an indoors environment. This is done by minimizing the amount of points to be processed at each scan by extracting regions of interest from movement detection and motion prediction approaches without any previous knowledge of the environment. The movement detection is done based on the principle of background subtraction and motion prediction is done assuming a constant target velocity.
- A robust and consistent performance when dealing with extreme pose changes such as crouching, jumping and stretching. Our solution also successfully overcomes sensor's limitation such as a narrow vertical field of view of 33 degrees. Even though the target might get cropped when standing near to the sensor, the presented solution can maintain the track consistently.
- The prototype is validated on a real 3D LiDar data set taken with our sensor in our work office. The results presented are overall satisfactory and show a specially high confidence for positive classifications.
- This work has been implemented, tested and validated with a 3D LiDar sensor, and it represents about 1850 lines of code written in a ROS node that work with standard PointCloud2 data.

This document is organized as follows: In the next chapter we expose the context and problematic of our problem. In Chapter 3, we present and analyze the related works in this field. Then, we discuss the methodology and implementation of our solution in Chapter 4. After, we evaluate and analyze the performance of our prototype taking into account multiple classic metrics for computer vision and classification applications in Chapter 5. Lastly, we conclude and discuss possible future works in Chapter 6.

— 2 —
Context & Problematic

This chapter aims to describe our application explicitly, explain the environment in which the test data was collected and to give a clear idea of the interest of our approach. Moreover, it exposes the limitations of the sensor and some of the most difficult situations for person detection. Knowing these two things is very important in order to conceive a prototype that is able to overcome them.

Our solution aims to solve the problem of detection, classification and tracking of persons in small or medium-sized indoors environments. For the moment our solution does not deal with a moving platform. Instead, it focuses on a static sensor, which is conceived for an indoor application like in social robots for person following. Even though usually in these applications a mobile platform is necessary, a complete solution for a static platform can also be interesting because many of them could profit from a pre-movement stage in which the robots can identify their surroundings. Moreover, in future works robot movement could be introduced and the current framework could be adapted to do SLAM first and then DATMO.

Other indoor applications with a static sensor could be for industrial robots and human-robot collaboration. Using a solution like the one we propose, the robot could detect and track the person that is working with in order to avoid accidents and collisions. Future works in specific pose estimation and understanding would be very efficient in this area.

The sensor was positioned at a height that tried to mimic a head-mounted sensor on a mobile robot. The height in which the sensor was placed was about 1.3m chosen according to the height of some mobile robots available to us in our laboratory. The test set for evaluation was collected at this height in our laboratory office for experimental procedures. The office represents a good example of a small or medium-sized indoor environment, with a simple square layout that extends up to 12m by 7m. Due to office divisions causing beam occlusions, the sensor captures the entire 12m of length of the office but can only capture a zone of about 5m in width. In this space is where the data collection is done. In Figure 2.1 we can see a sensor scan of the office that illustrates our setup during the entire collection of the data. The sensor location is represented by the red arrow and it is configured to register only what is in front of it, since there is a wall behind the sensor and it is not of our interest.

In this chapter, we first go into details about the sensor specifications, operation modes and limitations in section 2.1. Then, we show some of the most difficult situations for our prototype to handle in section 2.2.

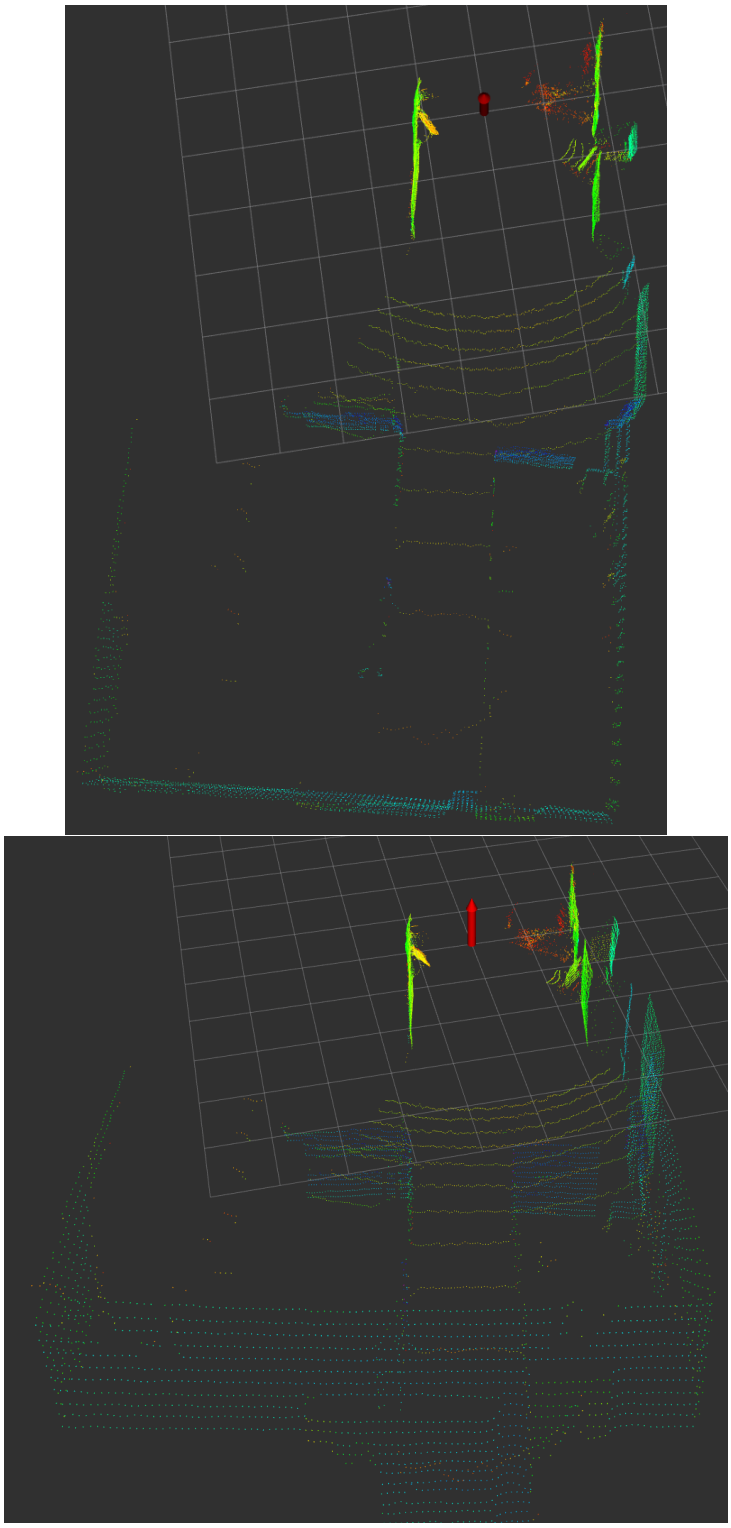

Figure 2.1: On the top, a top-down view of the office where the data was collected. On the bottom, the office viewed from another perspective. The sensor location is represented by the red arrow.

2.1 Sensor: Ouster OS1-32 specifications

The Ouster OS1-32 is the 3D LiDar sensor we used for our application. In Figure 2.2 the official specifications of the sensors are shown. The sensor has a vertical resolution of 32, which means 32 layers of point cloud rows. The sensor can do scans in 360 degrees but it is also possible to change this setting in case we only wish to capture in front of the sensor (in cases where the sensor is mounted in front of the robot/vehicle). In any case, the sensor has a customizable horizontal resolution (number of points in each row) with three different options: 512, 1024 and 2048 points per row. In the first two configurations the sensor can do up to 10 or 20 scans per second, and with a horizontal resolution of 2048 it can get up to 10 scans per second.
The maximum range of the sensor is up to 150m which is sufficient for an outdoor application and even more for an indoor one. On the other hand, the minimum sensing range is not something usually considered to be important for applications where the sensor is going to be far from the targets, but for applications such as social robotics it is important to keep in mind that a head-mounted sensor in a robot is only going to be able to sense objects that are further than 0.8m. Overall the sensor is very accurate as it is state-of-the-art technology, with a precision of 1.1 cm in a 1-20 m range, that ensures a confident measurement. Besides, the nature of the 3D point clouds and the fact that the points are going to be processed together (voxelization) and not by their own makes the application very resistant to sensor noise.
Furthermore, the sensor readings offer multiple characteristics of each point that belong to an object. These parameters include intensity, reflectivity, near infrared ambient light and range. Therefore, they provide interesting information from each point that could be used to enhance the segmentation results.
Choosing the operation modes of the sensor is part of our design process. Between the most important aspects of the design are the horizontal resolution, the frequency of operation and the Azimuth window.

Range (80% Lambertian Reflectivity)	110 m @ >90% detection probability, 100 klx sunlight 150 m @ >50% detection probability, 100 klx sunlight
Range (10% Lambertian Reflectivity)	50 m @ >90% detection probability, 100 klx sunlight 65 m @ >50% detection probability, 100 klx sunlight
Minimum Range	0.8 m for point cloud data
Range Accuracy	±5 cm for lambertian targets, ±10 cm for retroreflectors
Precision (10% Lambertian Reflectivity; 1 standard deviation)	0.8 - 1 m: ± 1 cm 1 - 20 m: ± 1.1 cm 20 - 50 m ± 3 cm >50 m: ± 5 cm
Range Resolution	0.3 cm
Vertical Resolution	16, 32, or 64 channels
Horizontal Resolution	512, 1024, or 2048 (configurable)
Field of View	Vertical: +16.6° to -16.6° (33.2°) Horizontal: 360°
Angular Sampling Accuracy	Vertical: ±0.01° / Horizontal: ±0.01°
False Positive Rate	1/10,000

Figure 2.2: Specifications of the Ouster OS1-32 sensor. Taken from the official data sheet given by the manufacturer.

2.1.1 Horizontal resolution

As mentioned before the sensor can run at three different horizontal resolutions and two different frequencies. On the resolution side, the main difference between the three is off course the number of points at each layer of the scan. The higher the resolution the higher the number of total points in a scan. In this case we have to make a trade off between complexity in computation and resolution. For our solution, we found that 1024 horizontal points per layer lands a nice trade off. In fact, the choice on selection depends as well on the environment in which the robot is going to be operated. For closed and small indoors environments a resolution of 2048 is not going to deteriorate the computation times by a significant amount. But at the same time, a jump from 1024 points to 2048 does not make too much of a difference for these environments since the furthest detection is limited by the size of the room instead of by the range of the sensor. On the contrary, for outdoors application most of the time the range of the scans could be limited by the obstacles or even by the maximum range of the sensor (150m). In these cases, a higher resolution would be ideal if the wanted sensing range is to be maximized, and in fact the maximum number of total points is the same as it would be in any environment but in outdoors spaces we are more likely to obtain more points than indoors. Again, a trade off has to be done and for every specific application a design stage has to determine the best operation mode for it.

2.1.2 Frequency

Frequency wise, at our horizontal resolution of choice 1024 we have the choice to go either for 10hz or 20hz. The end goal of our application is for it to be real-time, and according to our tests when the system is detecting and tracking a single person the prototype is able to run at full speed in 10Hz most of the time with some occasional dips when object occlusions or a track loss happen. Otherwise, when tracking multiple people the amount of process points and track handling increases computational time and the prototype runs at a frequency lower than 10Hz all the way down to 1Hz for very cluttered situations. Running at full speed in these situations remains a topic of discussion and a subject for future works because of the high amount of data in 3D point clouds. In conclusion, configuring the sensor to 20Hz of frequency will not make any difference for our application since even running at full speed more scans per second would not increase performance in any way.

2.1.3 Azimuth window

Even though the sensor can do scans in 360 degrees it can also be programmed to work in an specific "azimuth window". The default azimuth window is [0, 360] but it can be changed according to the needs of the application. In Figure 2.3 we can see the grid convention for the azimuth window parameter along with the frame of reference for the coordinate system. It is true that one of the interests of this sensor is that it can work for 360-degree scans, as it could be placed on the top of a mobile robot or an autonomous vehicle. In spite of this, in order for the sensors to be properly protected in an unpredictable environment such as the ones they could face in a vehicle they are usually placed in a more conservative place like the front part of the vehicle. On the contrary, for a mobile robot in a controlled environment the robot could benefit from having a complete perception of the environment, being able to detect and track persons

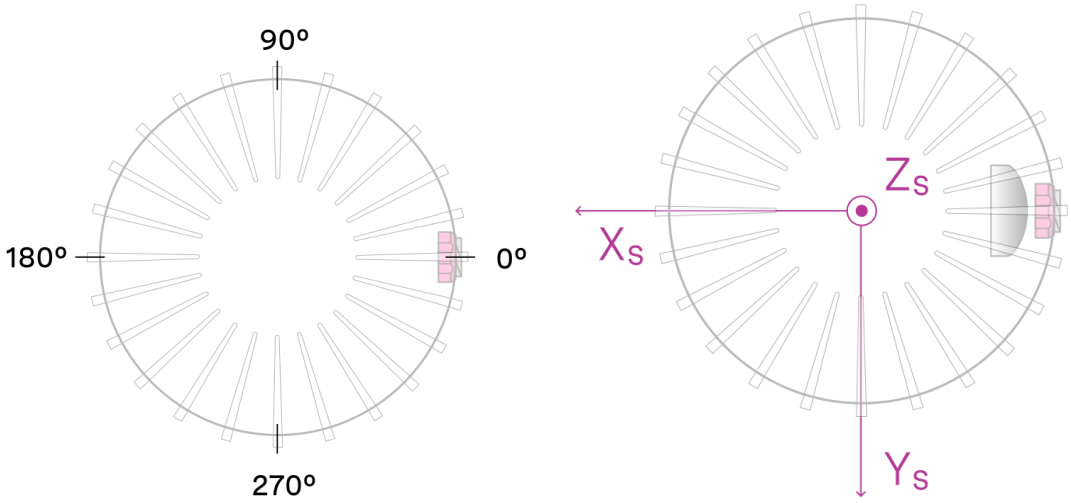

Figure 2.3: On the left, azimuth angle convention. On the right, top-down view of the coordinate system.

on his path (towards the front of the robot) and also on his back. This configuration depends on the application and our prototype works correctly in any configuration of the azimuth window. In our case, most of the tests were done with the azimuth window configured to see the front and sides of the sensor [70, 300] as it was placed in front of a wall for convenience reasons.

2.2 Difficult situations and limitations

Understanding some of the difficult situations we might encounter in an office environment is important in order to include as many techniques as possible in our prototype to overcome them. In this section, we first talk about an important sensor limitation: a restricted vertical field of view. Then, we analyze how extreme pose changes can be difficult to threat in most tracking applications, specially when working with 3D point clouds. Finally, we expose the problem with occlusions.

2.2.1 Important limitation: restricted vertical field of view

The Ouster OS1-32 that we are using for our application has an important limitation, it has a restricted vertical field of view of 33.2 degrees. This means that even though objects are detected from a minimum range of 0.8m, an object that is close to the sensor would appear cropped in the scans. Therefore, it would be hard for our classifier to identify a person when their geometric features are affected due to an incomplete scan of the body. How much the person is cropped depends on the distance to the sensor and on the height in which the sensor is located. For our application the sensor is located on what would be the height of a social robot, which is at about 1.5m.
This is in fact a limitation of our configuration since for small or medium-sized indoors environments we can expect people walking near to a social robot so it could happen multiple times

in any situation. For other applications like autonomous driving or surveillance this problem is negligible because most of the obstacles are going to be farther away from the sensor, but in our application it is an important limitation. An example of this situation can be seen in Figure 2.4.

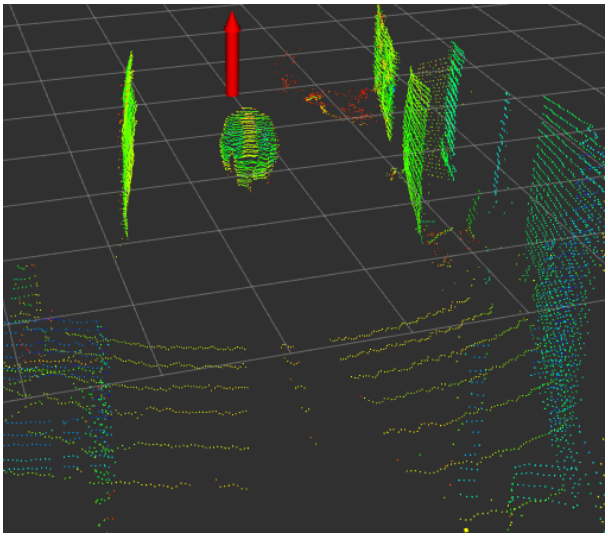

Figure 2.4: Results of classification when the person is close enough to the sensor to be not fully visible. The left is the raw data, the right shows the person in violet and the background in red.

2.2.2 Extreme pose changes and object carrying

The variation of poses that a person can undergo have always been a difficult subject to threat in classification tasks. Classifiers, whether they are done with deep learning supervised models or are done based on geometric features comparison, rely on the knowledge and modeling of the target's features.
In the case of supervised models, in order to take into account extreme pose changes, the training set should contain a balanced amount of data for both regular human poses and the most complicated ones. This can also bring generalizing problems, and could end up confusing the model. Another problem with supervised models is that at the end the model might classify incorrectly a person when they are holding an object in their hands, like when carrying something in a pick and place situation or carrying an accessory on their hands or backs.
For geometric classifiers, they rely on the comparison of pre-established thresholds. These thresholds are picked according to what we know about human geometry. For example, it is easy to say that a person should not be wider than 2 m. In this case, extreme poses like crouching can be hard for these types of classifiers, as they are intended to detect a person in their most common poses.
In Figure 2.5 we can see a variety of extreme pose changes like crouching, stretching and even a pick and place situation. In chapter 4 we explain how our prototype is capable of handling such situations.

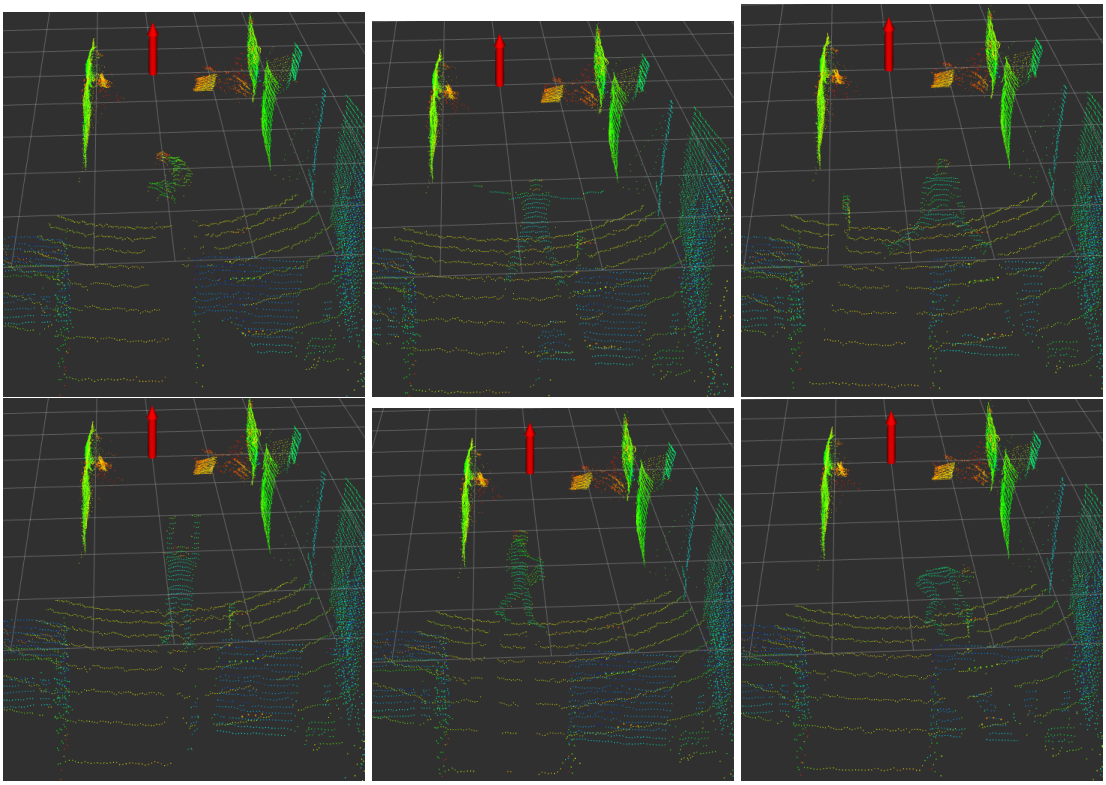

Figure 2.5: Typical pose changes a person can undergo on a day to day basis. Ranging from crouching, exercising, stretching, holding hands up to pick and place actions (in that order). The red arrow represents the position of the sensor.

2.2.3 Occlusions

When a moving object is being tracked, the occlusion problem is one of the most common ones to be encountered. Tracking applications rely on one or multiple sensors from a specific point of view to capture the environment. Having a sensor at a specific point of view generates the problem of moving objects being occluded by static objects or each other. To solve this problem, many techniques have been proposed and they depend on the type of data the sensor is providing.
Due to a lack of time, our solution does not deal implicitly with the occlusion problem. Still, it is important to acknowledge it, understand it and see how our prototype behaves when these situations happen. In Figure 5.3 we can see some examples of the occlusion problem.

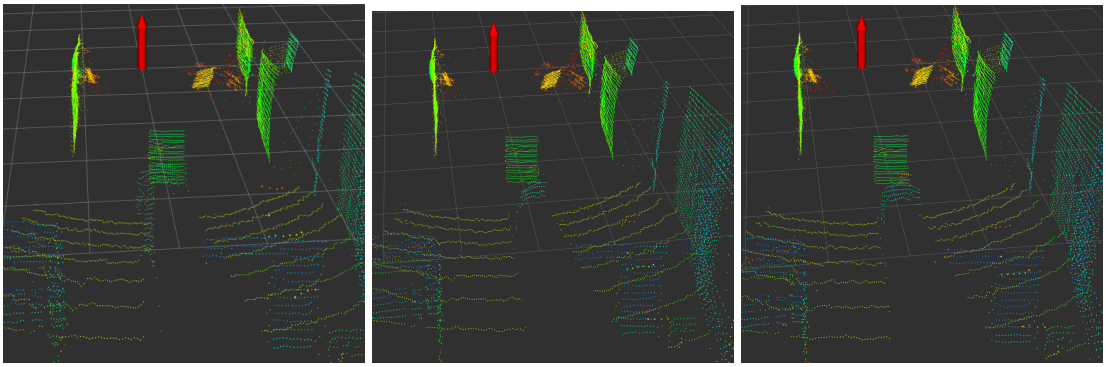

Figure 2.6: Three examples of the occlusion problem. An obstacle is placed in the middle of the office. The obstacle has the following dimensions: 1.3 m x 0.6 m. The red arrow represents the position of the sensor.

— 3 —
State-of-the-Art

The literature for laser point clouds processing in the research community is extensive. As our goal is to solve a simplified DATMO problem with the help of classification from LiDar 3D point clouds, it is important to understand the related works in those fields. In order to understand the evolution of the research over the years, it remains important to understand the main contributions from both 2D and 3D point clouds DATMO applications.
Classification in DATMO application is often regarded as an addition to determine the class of moving objects. For our application, classification is the main method used for separating background and foreground objects, as we wish to strictly determine that a moving object at any given time is in fact a person in order to begin tracking. This is why a consistent classification module is necessary. In this particular field, there are two approaches that dominate the research community: geometric features based classification [2] [1] [11] and using deep learning models [21] [29] [26] [27]. For the latter, the applications were usually reduced to single scan classification and were not used in real time applications until a few years ago. Meanwhile, classification with geometric approaches are more predominant in DATMO applications as they are simple and often work fine enough in conjunction with a proper module for tracking. For our application, we wish to stick to geometric features classification exploiting the detailed nature of 3D point clouds with high resolution that works well with a proper segmentation method.
In this chapter, we first analyze the related works on the field of DATMO with 2D point clouds and then we go into more details when exploring the previous works done with 3D point clouds. Finally, at the end of this chapter we summarize the points that make our approach innovative and we give a brief overview of the structure of our solution.

3.1 2D Point clouds

Most of the applications for DATMO with 2D point clouds were done with an autonomous driving application in mind, therefore the environment is considered as dynamic and the research has to be partitioned in two main parts: SLAM and DATMO. This is the case for [22] [23] [24]. Despite this fundamental difference, we can focus on the DATMO part of the previous works to understand the benefits and shortcomings of working with 2D LiDar sensors.
In [22] DATMO is performed based on the SLAM results. An incremental mapping approach is proposed, which consists of building a local vehicle map that is updated when new scans arrive. This new data coupled with the estimates of vehicle locations obtained from a scan

matching algorithm gives a proper representation of the environment. Therefore, by comparing new measurements with the constructed local vehicle map dynamic objects can be detected, segmented and tracked. In this approach, there is no classification stage, which leads to detection and tracking of any moving object that are presumed to be vehicles. As a result, movement models are generalized to motor vehicles and pedestrians are not considered. Furthermore, not knowing anything about the objects of interest beforehand can also result in difficulties for the algorithm. For example, since the 2D scanner only sees part of the object, the contours changes with movement result in degradation of tracking results.
To solve some of these problems, in [23] a model-based approach is implemented to interpret laser measurements. In this approach tracking is considered in a sliding window of time. Using object models, each observation generates specific object hypotheses (bus, car, bike, pedestrians) and a data driven Markov chain Monte Carlo technique is employed to traverse the solution space. Even though it worked fine for vehicles and buses, the problem of this approach was that due to the limited geometry and not descriptive measurements of the 2D LiDar sensor, there were problems in the interpretation of measurements from smaller targets like bikes and pedestrians.
Our approach is done only with the data of a 3D LiDar sensor, but many previous works [24] [16] have worked with sensor fusion to provide a more reliable and more accurate model than a single sensor data would provide. In [24] a similar approach as in [22] is implemented but with the help of measurements from a radar sensor and using Multiple Hypothesis tracker and adaptive Interacting Multiple Models filter which generate a reliable tracker. In the case of [16] fusion is done with stereo vision data. In this approach, the work of the laser processing in [23] is used to get a list of objects with position and dynamic properties for each one. Then this information is fused with the information provided by the stereo vision framework output in [18] with Bayesian fusion technique. It is important to understand the advantages of sensor fusion, as in future works our approach could be expanded to work with a similar stereo vision framework to enhance the results.
The local occupancy grid in [22] and [23] is equivalent to our background model, obtained from the calibration stage at the start of our framework. After the background model is obtained, we can also detect the inconsistencies between it and new scans to detect moving objects.
Even though [23] solves many of the issues in [22], both still rely solely on movement detection, so an object could only be detected and tracked from the first movement.

3.2 3D Point clouds

Understanding the shortcomings of 2D point clouds, the use of more accurate and descriptive 3D point clouds began to gain popularity in the research community but the literature for DATMO in 3D point clouds is not as abundant as it is for its 2D counterpart. In spite of this, many approaches have been done mainly for autonomous driving [1] [3]. All of which can be taken into consideration when comparing to our work even though the fundamental difference of a moving sensor (dynamic environment) exists.
On the other hand, an approach with a goal more similar to our own in [4] aims to solve person tracking in large public spaces with static sensors. The main difference from a conception perspective is that there are multiple 3D LiDar sensors in a top-mounted position to deal with less

occlusions.
Furthermore, there are many works in classification and segmentation that were intended for single-scan multi-class classification and segmentation [2] [29] [11] but can possibly be scaled to work appropriately for a stream of scans and DATMO with the proper process. The main concern with these approaches is that because they were conceived for a single-scan classification their application in real time for DATMO could be at first sight difficult.
Lastly, it is also important to acknowledge the works done for classification in 3D point clouds with deep learning models [20] [26] [30] [27]. These approaches gained a lot of popularity in the research community as numerous features from 3D point clouds with high resolution can be extracted to train a model. Each of these categories have something to contribute to research for DATMO in 3D point clouds, an it is important to take a more detailed look into them.
In this section, we analyze previous solutions of problems involving 3D point cloud data in the following order: First, segmentation and geometric classification. After, we take a look at DATMO in autonomous driving applications. Then, we talk about DATMO with stationary sensors, which is more in line with our own application. We discuss deep learning for 3D point cloud data later, and finally we briefly present previous works for DATMO in stereo vision and RGB-D cameras.

3.2.1 Segmentation and classification

It is important to begin with this section as in our approach segmentation and classification are the most important stages of the proposed solution. In this section, we tackle only the related works on geometric classification, this means classifiers that mostly rely on the comparison of pre-established general geometric features such as the proportions of the dimensions or minimum and maximum height, width and length. These approaches often rely on a good segmentation stage to segment objects to then classify them according to their features. It is also worth mentioning that these approaches are conceived to work with single-scan images and not for a stream of scans as it is in our application (where temporal association is necessary).
In [11], an interesting approach is presented to segment and classify pole-like objects such as lamp posts, utility poles and tree trunks. The automatic extraction of these objects could reduce mapping costs. For our application in indoors environments (such as offices) one could think of a way to adapt this approach to segment and classify persons as they are in fact pole-like objects and in an office environment the variety of these classes would not be as diverse as it is in outdoors data. The pole-like object candidates are generated from a voxel-based shape recognition. Then, according to vertical continuity the objects are detected and segmented using a circular model with adaptive radius and vertical region growing algorithm. Finally, further classification into different categories can be done according to shape features and spatial relationships. Although some aspects of their approach seem potentially beneficial for our application, the complexity of the algorithms use to segment and identify pole-like objects would make a real-time application very hard to accomplish.
On the other hand, the approach presented in [2] aims to solve the problem of segmentation and classification of every object in a 3D point cloud scan of an outdoor environment. The final result is a label for each point of the cloud with a specific class (building, tree, floor, post, cars). The points in the scan are grouped in voxels and then normalized attributes are given to each voxel to transform them into super-voxels. This approach seems very interesting since

the notion of super-voxel and their attributes can be really helpful when performing a correct segmentation of the objects. In this way, segmentation is not only done by euclidean distance but also according to the other attributes of the super-voxels, such as mean intensity, vector normals and color.
Since the voxels are made on a nearest neighbor basis their dimensions vary from each other and this is important to maintain the integral shape of each object. Once the super-voxels are done, a proper segmentation can be done using a chaining method between super-voxels that have similar attributes and are close enough to each other. What makes this approach so interesting is that the segmentation uses other attributes that point cloud sensors offer other than distance between points to ensure a robust segmentation. The result of the chaining process is multiple objects in the scan, meaning that we could go from 32000 points to only 10-13 objects in a given scan. This makes the classification easier since the classifier only has to classify each object according to geometric models built from local characteristics and descriptors from each one.
This approach is very interesting and it works well for outdoor multi-class classification. The main challenge is that the chaining process for segmentation can be computationally expensive and could be difficult to adapt to a real time application. Despite this, the super-voxel definition and the segmentation via chaining is adopted in our final approach because the robustness in the segmentation process is key for a good performance of the entire process.

3.2.2 DATMO in autonomous driving

Autonomous Driving is the most studied field for sparse point clouds. The nature of the sensors bring substantial improvement over classical vision sensors, as in the case of difficult lighting conditions and it also provides reliable depth information which is important for such applications. We have already seen how useful 2D sensors were for this application, but it is even more important in the context of our work to see how 3D point clouds can be used for real time DATMO applications in Autonomous Driving.
Being in constant movement, the first step (as it was for 2d point clouds) is to do a SLAM approach to model the environment. Despite the fact that our sensor does not move for the extension of this work, we also do a model of the environment in a different way, and techniques for DATMO can be compared as equivalents.
In [1] a very important contribution with good results was studied. Similar to the works in autonomous driving with 2D point clouds [22] [23] [24], a SLAM step is performed to create a local occupancy grid. In order to handle the larger amount of points, a voxel approach with an octree based occupancy grid is adopted. Once the local map of the environment is done, similar to [22] and [23], moving objects can be detected when new measurements arrive. The hypothesis of a moving object is based on the inconsistencies between scans, and voxels that fall into this criteria are considered to belong to possible moving objects. After having this list of dynamic voxels, segmentation is done by euclidean distance. This approach borrows the main idea from background subtraction methods in classical computer vision.
The same principle is applied to our work for the movement detection part, but the modeling of the environment is done with only a few scans at the beginning of each launch considering that the sensor does not move. On the other hand, their method for segmentation is basic and it only takes distance into account which can lead to segmentation errors grouping voxels that don't

belong to the same object when they are close together. Furthermore, they use this approach only to identify voxels that belong to moving objects and eliminate the ones that are below the threshold for number of voxels per object (noise data), but the notion of segmented individual objects is not maintained.
The effects can be seen more clearly in their method for classification. The classification is strictly based on the shape of bounding boxes created around the clusters of dynamic voxels left after filtering the noise data. Once again this method is not thorough enough and can yield incorrect results such as grouping multiple clusters that are close together generating an incorrect bounding box that contains multiple objects such as a group of pedestrians. To avoid these problems, in our approach we maintain the object representation for the voxels classified as dynamic. This means that after our segmentation/chaining process we would have only objects in the region of interest. When the segmentation process works well, the result is multiple objects that are moving, these objects are later classified as either a person or background with a voting system from different classifiers for shape, vector normals and possible shadows.
Formulating a DATMO problem in 3D point cloud data as a variant of a background subtraction problem is very popular in the research community. Another approach that relies on this computer vision technique is presented in [3], where the goal is to propose a real-time algorithm for dynamic object detection in autonomous driving applications. Tasks such as SLAM, and subsequent tasks such as detection, clustering, tracking and classification pose a challenge for real time applications. This is why in this approach a prior 3D map of the environment is created offline to provide a static background model. With this model the problem can be solved with localization and dynamic events representation which derive from the concept of the background subtraction problem.
Even though this approach presents good results for real-time applications, it does need a first stage to create the static background model, where the "training" set for its creation is supposed to not contain any moving obstacles. Similar to our own approach, once the static background model is done, moving object detection can be considered a binary classification problem with only background and foreground (in our case persons). Another similarity is that this first stage of creation of the background model is the most computationally heavy stage of them all.

3.2.3 DATMO with stationary sensor

Although DATMO with stationary sensors is far more uncommon than applications with moving sensors, some work has been done in this area. In [4] multiple person tracking using 3D range sensors is done to understand people actions in a commercial environment. Multiple sensors were placed in a shopping center to track persons and estimate their body position, height and orientation. This information could be helpful to identify people interested in a certain brand or marketing ad. In this approach, multiple range sensors were overhead mounted to avoid occlusions in clutter situations. Although in said work the goal was understanding people's reaction to their environment it could also be used for applications such as human-robot interaction which falls more in-line with our own application.
The detection of the moving object relies also on the background subtraction principle. The background model is learnt beforehand when no moving objects are present. From there, points that belong to moving objects are clustered by distance and then tracked. One of the main disadvantages of this approach is that it does not offer any way of distinguishing between persons

and other moving objects in the environment. This can be particularly challenging in an environment such as a shopping center, where many stores can change the background model by changing the objects in display in front of them. As mentioned before, our approach relies heavily on our classification module, even though we also use the principle for background subtraction, any segmented moving object has to be classified as a person first in order to begin tracking.
On the other hand, a similarity between the two approaches is that the complexity of the tracking is proportional to the number of persons being tracked as it is to be expected. For our approach, detection and tracking of one person works very well in real-time at the same frequency that the Ouster OS1-32 works (10 hz). For every other person that enters the scan at the same time computational costs increase significantly as the number of points to be processed increase. Moreover, in an indoor small environment such as an office, the moving objects can sometimes be so close to the sensor that one of them might block the laser trajectories resulting in a lot of discrepancies from the background model that also results in higher processing times.
Another problem in our application is that in general, the nature of 3D laser sensors imply that when an object stands in front of another, anything directly behind that object will be eclipsed. This will generate possible detection of movement around these eclipsed areas that we denote shadows for the purpose of explanation. This is why one of our classifiers is strictly conceived to classify these shadows as such and therefore obtaining less false positives classifications in this case.
In both cases the most computationally intensive part is the clustering/segmentation algorithm. In [4] the segmentation is performed point by point so the computational cost is high, in our case, even though we segment by voxels and then by chains, it is still the most expensive portion of our algorithm, but yield better results than traditional point by point clustering.
Other works as in [21] study people detection, classification and tracking for mobile robots and intelligent cars in populated environments. This approach is different as it does not rely on movement detection with the background subtraction principle but in classification of the whole scan each time a new measurement arrives. At each scan, each scan line (or level) of the 3D point cloud is divided into 2D segments that are then classified with an AdaBoost [7] model. This results in a set of 2D segments that, if classified as belonging to a person according to a set of 19 extracted features, together form a 3D representation of a person. This detector is single-frame and generates detection hypotheses from a single scan. It works with no temporal integration so a multi-hypothesis tracking method is introduced to integrate this information over time to ultimately smooth the detector output.
This approach does not deal implicitly with a moving sensor and tests are done with a static sensor. One similarity with our approach is that this one relies heavily on its classification module and uses tracking to smooth the output of the detector. The main difference is that in our approach our main focus is to always process the least amount of points possible which leads to better performance.
Another related work for person tracking with a stationary sensor is presented in [19], where a method is introduced to adapt the observation model of a particle filter to identify partial and full occlusions of a person. This is done by tracing rays from positions near the person to the sensor and determining if the ray hits an obstacle in that path. A voxel grid of a set constant size is used which facilitates retrieval and data reduction. Every voxel in the grid has an occupancy value which is increased at each scan if laser beams end in that voxel. The voxel grid is created at first and then updated with the arrival of each new scan.

This is similar to our approach after constructing the background model with the first few scans, the following scans are used to update and detect movement when compared to the background model. Furthermore, in their approach the supposition that everything that moves is a person is made. That means that there is no proper classification module and incorrect tracks could be created if any other moving object is present in the scan.
On the other hand, the main difference relies on the overall goal of the applications, since in their work they are more focused on how to deal with occlusions and the work is limited to one person tracking since it is more of a proof of concept for their method to deal with occlusions. Our approach aims to be a prototype of a complete modular solution, which can then be improved with more time by enhancing any of the modules in the implementation.

3.2.4 Deep learning for 3D point clouds

Deep learning for 3D point clouds is a very interesting topic of research in the scientific community. Multiple works have been done to detect and classify people and other moving obstacles in LiDar range data. As already mentioned, [21] uses Adaboost models to classify each 2D segment (or layer) of a 3D point cloud. Although this method yields good classification results, it still relies on 2D geometry and features to classify each segment at every height, therefore taking advantage of all of the previous work done for 2D point cloud classification already. On the other hand, other works have focused more in taking advantage of the descriptive nature of 3D point clouds and working directly with 3D data instead of transition to 2D first.
One example of this is in [29], where the idea is to segment and classify objects that could move in a single-scan reading. Including objects such as cars, pedestrians and bicyclists which are classes of interest in autonomous driving applications. In this approach, the methodology consists in solving a binary classification problem first (separating data into background and foreground) and then solving for the multi-class classification problem of the objects in the foreground. Clustering is done using Euclidean Minimum Spanning Tree, and the spanning tree is broken wherever an outlier is found using the RANSAC paradigm [6]. Given that the clustering is once again done only by euclidean distance, some background clutter could be present after the segmentation. This is why foreground extraction is done as a supervised classification task with Support Vector Machines before doing the clustering part.
There are also other works that rely on deconvolutional networks for their applications, which in these cases are usually for detection and tracking of vehicles and not pedestrians [26] [27]. Even though the goal of these approaches are very different to our own, it is important to acknowledge the advances in this area as they show that good results can be obtained with trained CNNs and correct pre-processing of the data. Moreover, they also include a temporal integration which shows that the work of classification with deep learning models can go beyond single-scan detection and classification.

3.2.5 DATMO with stereo vision and RGB-D cameras

Stereo vision and RGB-D cameras are also widely studied in the field of detection and tracking of objects, there are many works that aim to take advantage of these technologies.
In the case of RGB-D cameras, they present an enticing offer since they provide both RGB in-

formation and depth data. RGB information could be useful in general vision applications like detection by skin detector and also for specific applications such as person re-identification. Meanwhile, range data is helpful to have a better understanding of the environment and accurately obtain distance values as already explained. In [9] a human tracking system is done based on a 3D mean shift algorithm for mobile robots. In this work, the range data was added to the target and candidate model as a weighting function for human prediction. Other works as in [8] implemented a person detection and tracking solution for service robots using a RGB-D sensor. But in this approach, because the robots were low leveled they were only interested in segmented and classifying the legs of the persons. In [14] both types of data provided by a RGB-D sensor are taken advantage of when a person detector and tracker is done using a combination of histograms for height difference and color. In [13] a prior knowledge guided random sample consensus fitting algorithm is used to detect ground plane and ceiling points to reduce the amount of data to be processed, this is similar to the goal of our application but we do not have at any point any prior knowledge of the environment. Also, plan view maps are used to identify person candidates from point cloud clusters and a weighted histogram is extracted to feature a person candidate. In [10] they implement a real time solution capable of running up to 20fps without the use of the GPU in most cases. Similar to our approach, they rely heavily in regions of interest processing and extraction to concentrate the efforts so overall performance is better. They present two detectors that depend on the distance to the object provided by the depth sensors. For a close range an upper body detector which is depth based is used, whereas for longer distance a full body detector performs better. In [5], a real time RGB-D SLAM method is proposed. This method is fast and memory efficient and it is conceived for dynamic scenes. To gain memory efficiency, they assume that most robotics applications do not require a high resolution level of the environment, so a simpler but less expensive map could be useful. They also present a simple and fast way to detect moving objects.
On the other hand, in the field of stereo vision we have works such as [28] where the aim is to present a solution for moving obstacle detection and tracking in autonomous driving applications. In this work a method based on an illuminant invariant image is done to detect free road space, then a ROI is extracted using a convex hull. And then a particle filter is used for tracking multiple objects. By using range data, other information such as target size verification and introduction of redundancy are implemented to improve accuracy. In our work, target size also plays an important role in classification and can be more reliable due to the accurate nature of the 3D LiDar sensor, another similarity is the introduction of redundancy for improved results. For a similar application, in [15] an algorithm for detection of moving objects is presented. The proposed solution is based in dense stereo matching and optical flow estimation where a fast algorithm based on Lucas-Kanade paradigm is used. In a more recent work [12], a deep learning solution of a R-CNN model is used with stereo vision data for object detection in an autonomous driving application. Using data from stereo images, the sparse, dense, semantic and geometric information is exploited. The R-CNN model extends from Faster R-CNN for stereo inputs to detect associated objects in left and right images simultaneously. This method does not require depth input and 3D position supervision but the results are state-of-the-art for supervised image-based methods. This is another example of the potential of deep learning models for DATMO applications.

3.3 Summary of our solution

In this section we explain some of the innovative features of our approach that makes it stand out from other previous works already explained in Section 3.1 and Section 3.2. Moreover, we give a brief overview of the final structure of our solution.
Firstly, in contrast to [23] and [22] , our work is not limited to a first movement detection, since it can detect a person even if it is static from the beginning with our geometric classifier. This is particularly important in our application since the calibration stage for obtaining a background model can be done accurately even if there is a person on the first scan. The former can be achieved by doing a proper segmentation and classification of the person in the first scan and removing these points from the final background model. Moreover, our approach relies heavily on our classification stage, as we don't track any moving object but only the ones classified as persons which decreases computation costs.
In order to achieve real-time performance, our solution relies on creating a model of the background at launch and then using a background subtraction algorithm that isolates moving points that can then be segmented and classified as either a person or not. This means that our approach is less computationally expensive than other approaches like [21] where the whole frame is used for the classification, as it only has to deal with every point in the scan (which is the worst case scenario) on the background modeling step. In conclusion, our approach relies heavily on the creation and processing of only regions of interest extracted either by background subtraction algorithm or by prediction which is a result of the temporal integration done with our tracking method.
As already explained, we find the approach for voxelization and segmentation done in [2] to be very robust as it takes advantage of the descriptive nature of the readings of the sensor instead of only using distance as the main criteria for clustering points like it is done in [1]. This is why we implement it for our own solution. The contribution of our work relies on its adaptation to a real-time DATMO application instead of being limited to single-scan segmentation. To overcome the difficulties of this adaptation and the computational cost, several methods for reducing the amount of points to be processed are done in our approach. The goal of which is to find regions of interest (ROIs), either by movement detection or with the help of tracking and prediction of moving object states when introducing temporal association between scans and objects. Another aspect that makes this possible is that because of the simplified nature of our problem, instead of multi-class classification we can focus on a binary classifier, the segmented objects are classified as either background or foreground (person).
In [3], a prior map of the environment is taken in order to get their solution running in real-time. This solution relies on the assumption that at the moment of taking the map, no moving objects are present in the environment and therefore everything in the map can be modeled as the background. Contrary to this, in our approach, we do not make any assumptions about the environment. Meaning that even if there are people already in the frame, we are able to segment and eliminate them at the background model creation stage so we can remove them from the final model.
This has plenty of advantages as our approach can perform a good estimation of the background model at the start in any given unknown environment no matter if target obstacles are already present in the firsts scans. This is done by performing a complete scan classification of the environment at the first scans, which implies having to do voxelization, segmentation and classification for every point in the scan. This is why the creation and calibration of the background

model stage done in the first few scans is more computationally expensive as it has to process the highest amount of points possible in the whole stream of scans.
Regarding classification, we chose for our approach a simple geometric method that relies heavily on the robust segmentation of moving objects of [2] to extract proper geometric features such as shape and vector normals. For our application, our simple classifier works fine and coupled with the GNN [17] tracking solution it is robust to pose changes in the person. Our choice to not use a deep learning model for our classification module relies heavily on time restrictions, as a proper understanding of the architecture and training of a CNN model could (on its own) take more time that we had to conceive a complete prototype (4 months in total).
For future works, the nature of our modular solution allows the replacement or enhancement of any particular module to obtain better results or even expand its original goal of tracking only persons in an indoors environment. With enough time to work, and given that our solution relies heavily in classification, it would be interesting to implement a more robust solution for classification similar to those in [26] and [27] to potentially see better results.
Overall, our approach accomplishes all of the above thanks to the integration of our main modules for Voxelization, Segmentation, Movement Detection, Classification and Tracking which can be simple in implementation by their own but working together they result in a cohesive framework with good results. In the next chapter, we will talk more in detail about the conceptualization and implementation of these modules.

— 4 —
Methodology

To understand the structure of our solution, this section is layout to explain each of the modules or stages previously mentioned that in combination make our approach viable within our goals. Firstly, an explanation for the pre-processing stage of voxelization is done, then for segmentation, movement detection and classification. Each of these modules are explained firstly in a scan to scan basis and at the end the temporal integration of everything is detailed in the tracking section. But first, it is important to talk about the characteristics, functionality and operating modes of our 3D LiDar sensor.
In Figure 4.1 the pipeline of our solution divided in each of its modules is shown. This chapter aims to explain the details and implementation of each one of these in the following order: The Voxelization module in Section 4.1. Movement Detection and Segmentation in Section 4.2 and 4.3 respectively. Classification method in section 4.4, and lastly the temporal integration and tracking in Section 4.5. Even though the module for voxelization is shown after the movement detection module, it is important to explain the concept first as it is used for explaining the implementation of the movement detection module.

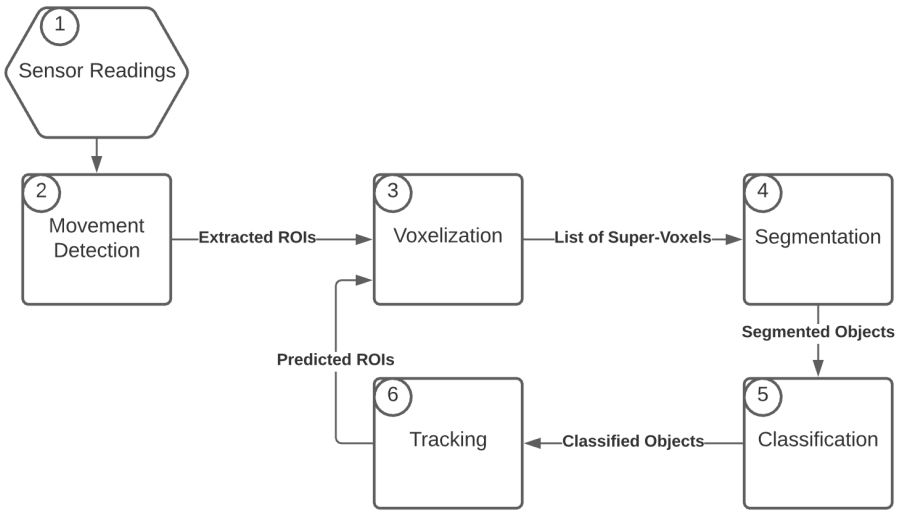

Figure 4.1: Pipeline of our solution. Here we show the different modules implemented for our prototype.

4.1 Voxelization

One of the difficulties of working with point clouds is the high amount of data in each scan, and our goal is to have a framework that works in real time and consistently no matter the duration of the application. This means that our solution has to be structured to handle memory constraints and make sure it is working as optimally as possible at all times. To ensure this, a proper pre-processing of the high amount of data is to be done if our goal is to always reduce the total amount of data to be processed. This is why we use the movement detection module to extract ROIs and therefore process the least amount of points possible at each scan since it is clear that the more points in the scan the slower the computation will be. Additionally, another approach that helps reduce the amount of points to be processed even more by grouping them in 3D point clouds is Voxelization.

In our approach, similar to that of [2] the voxelization is a two-part process. The first part consists in creating the voxels with a r-nearest neighbors approach. Where "2r" denotes the maximum length of the voxels. It is important to say that the value of the maximum voxel size "2r" is to be determined for a specific application. Multiple values were tested but at the end for our application in an indoors small office environment a maximum size of 0.4m lands a good balance between resolution (detail of shape) and performance.

The result is a list of voxels with different shapes and sizes only bounded by a maximum dimension of 0.4m. This is a fundamental distinction from other works that generally establish a fixed grid of voxels with fixed and regular sizes. This approach aims to retain the geometrical fidelity of each object due to the fact that classification is done mainly according to the shape of the objects, and a voxel would be the most basic element from which an object is composed. For better understanding, the voxelization process is presented in Algorithm 1.

Algorithm 1: Voxelization

Result: Super-Voxel list

1 **while** *there is a point not included in a voxel* **do**
2 Select search point;
3 Find neighbor points within a maximum spherical distance of $2r$ (maximum voxel size);
4 Create voxel with neighbor points and assign search point as voxel center;
5 Find voxel dimensions: Width, length and height;
6 Find super-voxel attributes: mean intensity, mean reflectivity, variance of the intensity, variance of the reflectivity and its normal vector;
7 **end**

Once the voxel list is ready, the second part of the process consists in turning our voxels into super-voxels. The term and definition of super-voxel also originated in [2] and it is a fundamental step to generate a more descriptive representation of a voxel that eventually leads to a more capable segmentation method. Overall, a super-voxel is a regular voxel with other intrinsic characteristics inherited from the group of points that are part of the voxel. These characteristics aim to add more descriptive features to each voxel other than just their shape and size. At this point, the different parameters obtained from the sensor data become useful for our application.

In Figure 4.2 we can see the class diagram of the voxels and super-voxels. All in all, the super-voxels could have any combination of the parameters provided by the sensor but the ones

that we ultimately kept are those which mattered the most when doing the segmentation, this part will be addressed in more detail in the segmentation section. Moreover, another important parameter is the calculated surface normal of each voxel that will help in the classification stage.

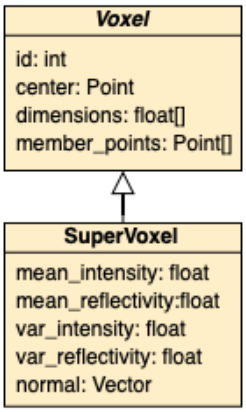

Figure 4.2: Class diagram for the voxels and the super-voxels

4.2 Movement detection

Now that we have introduced the voxelization method in the next section, we can explain how the movement detection is done. As our goal is to detect only people in the environment, we do not care about the background objects, which incidentally include the great majority of points at each scan. With this in mind, we aim to reduce the amount of processed points by applying our solution only to specific regions of interests that are generally several times smaller than the entire scans in not too cluttered situations (with a lot of people at the office). This is done in one of two ways that work in conjunction to optimize the processing at each frame: movement detection and motion prediction. In this section, we are going to focus on the movement detection module. In Figure 4.3, the pipeline for the movement detection module is shown.
The idea is simple: in an office environment most of the scans are going to be points belonging to background objects, and only a little part of them are going to belong to persons. This module aims to segregate the points that do not belong to the background to process them later. To do this, a model of the background should be created in order to compare it to consecutive scans and segregate the points that were not there previously, those points must certainly belong to moving objects and are what we call a region of interest (ROI).
Therefore, the movement detection module has two parts: creation and calibration of the background model and a method similar to adaptive background subtraction to extract new voxels that might belong to moving obstacles that will form our regions of interest. The extraction of these ROIs by determining the dynamic voxels is done at all times (even when someone is already being tracked) to detect other possible persons coming into the scan range.

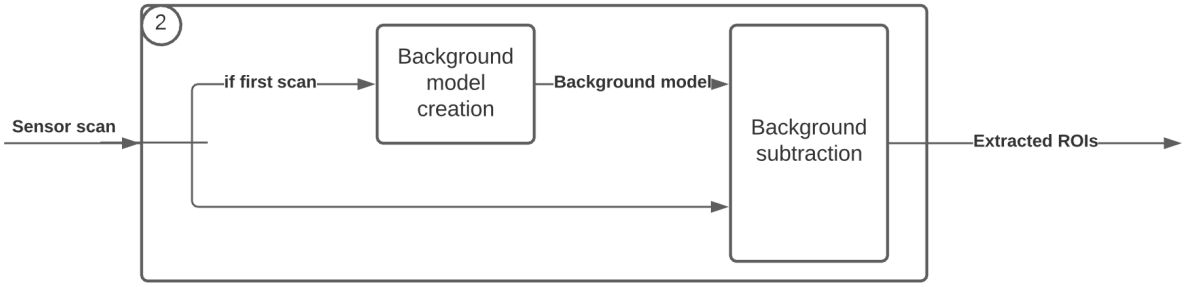

Figure 4.3: Pipeline of the movement detection module, it consists of the creation of the background model in the first scan and the subsequent background subtraction to extract ROIs.

4.2.1 Creation and calibration of background model

The creation and calibration of the background model has to be done online every time the framework is launched. The reason for this is that we do not work with any previous knowledge of the environment and our solution should work in any situation. Another design choice and functionality we want to have is not having to depend on a first movement to detect a person in the environment. Therefore, in our solution we do not assume that at launch the environment has no people in it so for the background modeling we first have to identify the background and the foreground (if any).
The identification of the background requires a full-scan classification, when the first scan arrives, the process for voxelization, segmentation and classification is done to the full scan. In this way we can have a first idea of the background, by eliminating the objects that were classified as persons and saving the rest of the voxels as our first model for the background. A typical result for a background model can be seen in Figure 4.4.
With this method, our approach is indifferent to any initial conditions for the environment, as

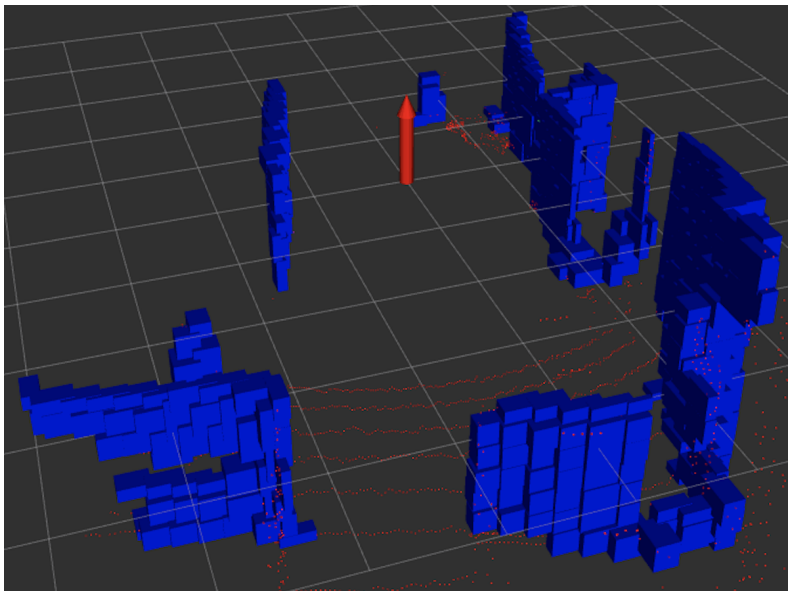

Figure 4.4: An example of a final background model

it should be able to create a first model of the background that does not contain any people on it.
In order to get the most accurate model as possible, after the first creation the background model enters a stage of calibration. The calibration allows for the model to be updated with subsequent scans. In our approach we use four scans after the creation to do the calibration. In this way, redundancy is introduced and possible errors for classification can be corrected in the calibration stage.
In the event of an error in the final background model like a person being miss classified as background, we found that our approach is still able to correctly detect the person once they start moving away from their original position.

4.2.2 Background subtraction: extracting ROIs

Once the calibration stage is done, we have created the background model and now we can extract the ROIs that correspond to the moving objects at each scan. As our approach does not rely on a fixed voxel occupancy grid, the voxels are created by grouping points in the near neighborhood of each center point (which are chosen at random). In this case, the same technique applied in [1] and many other approaches that use an occupancy grid is not possible for our application. Normally, once an occupancy grid was established, the movement detection would be done only by comparing and analyzing the changes of states in the grid.
In our case, the technique would be similar to the classical background subtraction method used with standard RGB cameras. The classic technique consists in getting a background model and then comparing it pixel by pixel (or group of pixels) to the new frame, the result would be the parts of the frame that do not belong to the background model and therefore are considered new moving objects introduced into the sequence.
The first challenge for this method is the nature of the measurements with a 3D LiDar sensor. At each 360-degree turn, the sensor shoots the beams of laser with a precision that depends on the distance between the object to the sensor, for objects in the 1m to 20m range the precision is of 1.1cm. This means that at each scan the position of the points may vary significantly, and some noise could be introduced that was not originally in the background model. For this reason, it is not possible to do background subtraction point by point.
The principle of the background subtraction process is summarized in Algorithm 2. As it can be seen, our approach relies on the voxelization process to overcome the issues caused by the sensors precision. This approach is therefore robust against the sensor imperfections and allows to segregate the new points in a reliable way.

Algorithm 2: Background subtraction

inputs : Voxel list from background model, point cloud from new scan
output: Moving voxel list MV (ROIs)

```
for voxel in background model do
    Get voxel center v_c;
    Find neighbor points of v_c within a maximum spherical distance of 2r in the new
      scan and save them;
end
Find points that did not fall in the vicinity of any of the voxels in the background
  model;
Apply the Voxelization method on them and save them on moving voxel list MV;
for each voxel V in MV do
    if number of points in V < minimum-density then
        erase V from MV
end
```

In conclusion, the worst case scenario (performance wise) is having to process every voxel in the scan at each time, but this is only necessary in the first scan to create the background model. For every subsequent scan, movement detection is done to obtain the list of moving voxels at each frame. This is done in order to process the least amount of points possible. Despite this, a high amount of moving obstacles might end up in an unavoidable decrease in performance. In Figure 4.5 we can see the result of the movement detection stage, when the person first appears in the scan range. The result is a region of interest and it is one input of our processing pipeline.

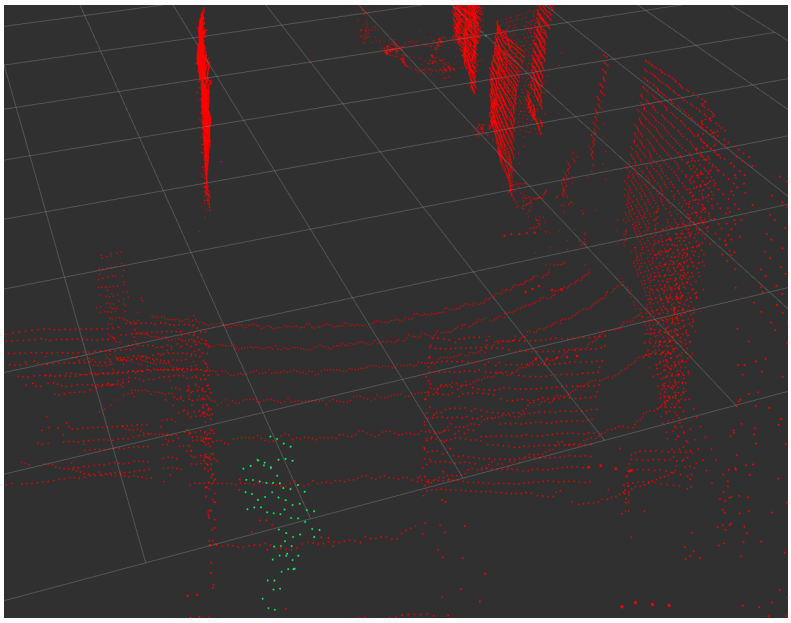

Figure 4.5: The result of the movement detection, in green the points resulting from the background subtraction method and in red the static points.

4.3 Segmentation

The segmentation part consists in obtaining multiple objects from the voxel list created from the regions of interest. In this way, we go from having around 32000 points in the worst case scenario to having less voxels and to finally having only 10-13 objects maximum. The accuracy of this process is fundamental to our overall solution since the classifier will classify each object according to their characteristics. Meaning that if an object is not correctly segmented, for example merge with another object the result of the classification will be incorrect.
To ensure a good solution for segmentation we can take advantage of the descriptive nature of the sensor readings instead of doing it based only in the distance between points.
In order to have the best segmentation possible, we use the robust segmentation method proposed in [2]. This is a method that relies on the chaining of super-voxels according to distance and their other descriptive parameters (refer to Figure 4.2). The final objects are the result of merging the resulting chains that share any links (voxels) between them. The method was conceptualized for single-scan processing as it could be computationally expensive to do with every voxel in the frame. In our approach we manage to make it work in real time by reducing the processed area to regions of interest whilst keeping the benefits of the robust segmentation. The result of the segmentation process can be seen in Figure 4.6.
The first step of the segmentation is to construct chains. Each chain is formed by many links (super-voxels) but usually are small in comparison to a full object. Chains are constructed by linking super-voxels with similar characteristics and that are close enough together. One chain is constructed of a principal and multiple secondary links. The criteria for linking one super-voxel to another is done the same as presented in [2]. In their work, they had access to RGB data as well which was very useful for the segmentation process. In our case we do not have access to RGB but we used reflectivity as another criteria. To reiterate, any super-voxel is chosen as the principal link of a chain V_p and then multiple conditions are checked to find suitable contenders for secondary links V_n as follows:

$$|V_{pC} - V_{nC}| \leq w_D \tag{4.1}$$

$$|V_{pI} - V_{nI}| \leq 3\sqrt{w_I} \tag{4.2}$$

$$|V_{pR} - V_{nR}| \leq 3\sqrt{w_R} \tag{4.3}$$

Where, for each voxel:

- V_{pC} and V_{nC} are the geometric centers
- V_{pI} and V_{nI} are the mean values for intensity
- V_{pR} and V_{nR} are the mean values for reflectivity
- w_I and w_R are the average coefficient of the intensity and reflectivity respectively, equal to the maximum value of the variances $Var(I)$ and $Var(R)$
- w_D is the distance weight equal to $\frac{(V_{p_{S_{X,Y,Z}}} + V_{n_{S_{X,Y,Z}}})}{2}$ where $S_{X,Y,Z}$ represents the voxel dimension in each axis

This is done until every super-voxel in the region of interest is taken into account and therefore is not necessary to pick a seed or a starting point as is often common in classical clustering region growing algorithms and there is no preference at all in the choice of principal links.
After every super-voxel is taken into consideration we would have a list of chains. Principal links are then linked together if they share any links between them whilst removing redundant links. Therefore forming our final segmented objects as we can see in Figure 4.6. As seen, this method is much more robust than regular region growing algorithms for clustering as it utilizes some of the parameters given by the sensor's reading. Furthermore, it is completely autonomous and does not need any previous knowledge or set up to be done.

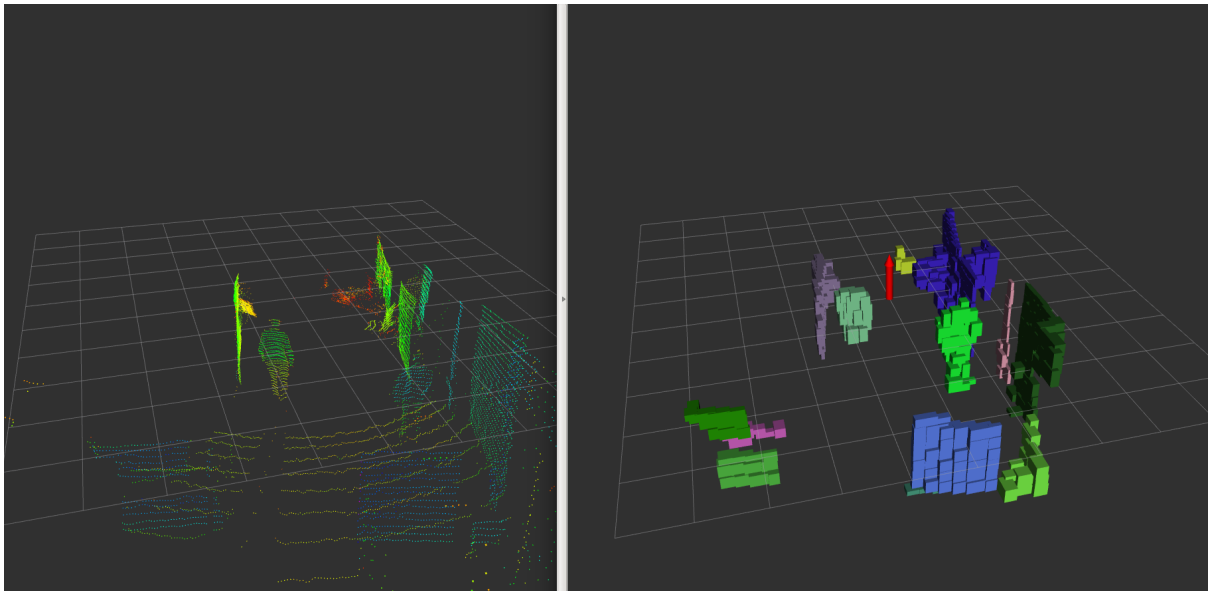

Figure 4.6: Result of the segmentation process

4.4 Classification

After the movement detection process, the list of dynamic voxels are the ones to be segmented into distinct objects. For the classification stage, we would have a list of objects that were present in the regions of interests. The pipeline for the classification module can be seen in Figure 4.7.
As already explained, many previous works in the literature rely solely on movement detection or changes in the occupancy grid to cluster and segment the objects and begin the tracking. On the other hand, classification is one of the most important steps in our solution as due to noise and beam occlusion some objects segmented from the dynamic voxel list could not be a person. By classifying the object first, we are reducing the noise to a minimum and dealing with possible inconsistencies in the background model due to a possible imperfection in the creation of this model.
This is why we only create a track when the object is classified as a person three consecutive times. By introducing this redundancy we overcome possible false positives, since our classification model is simple and works fine most of the time, but beam occlusion can eventually

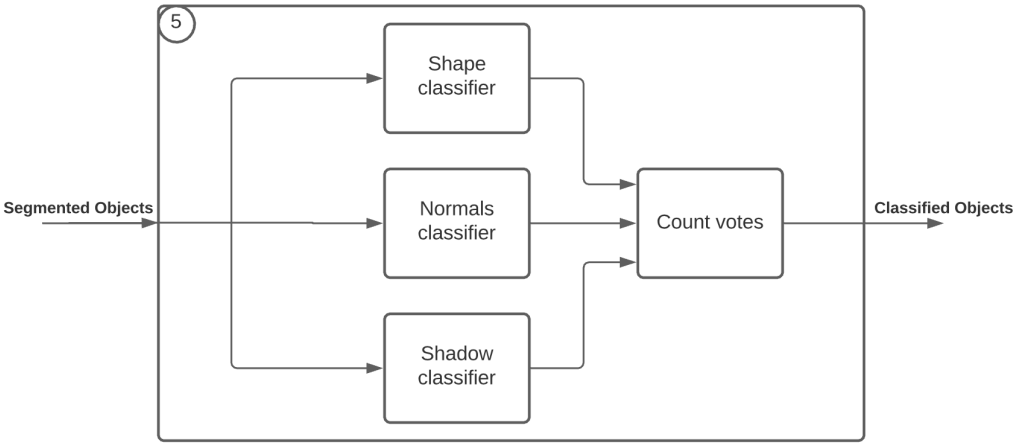

Figure 4.7: Pipeline of the classification module. It is divided into three weak classifiers. The final classification result depends on the votes of each one of them.

lead to a false positive classification. On the contrary, three consecutive positive classifications are considered to be redundant enough and lead to the creation of a track. Tracks and temporal integration will be explained in more detailed in the next section.
As mentioned before, for our solution we decided to work with a simple geometric classifier instead of using a supervised deep learning method. This decision was made based on time restrictions, as we wanted to complete a working prototype in the short amount of time we had. Overall, our classifier relies on simple geometric features comparisons to pre-established thresholds for characteristics such as shape and vector normal orientation. These thresholds are set according to pre-conceived notions and previous knowledge of the target object, which for our task consists only of persons and background. Despite its simplicity of coding, our classifier combined with the introduced redundancy is very reliable for positive classifications. On the other hand, false negatives within a track (for example in two consecutive scans) are the weakest point of the classifier by its own, but this is fixed using tracking techniques.

4.4.1 Voting system

As seen in Figure 4.7, the classification process is divided into three weak classifiers, one for shape, another for vector normals and the last one to discard mistakes due to beam occlusions or "shadows".
Each object goes through each one of them, and they assign a vote for either person or background according to their own criteria for classifying. The different criteria are explained in the following sections. After the list of segmented objects have gone through each one of the weak classifiers, each object in the list has a certain amount of votes assigned by the classifiers for either person or background. At the end, the votes for each object are counted, and an object is classified as a person only if it has more votes for person than for background. It is worth mentioning that some of the weak classifiers only have criteria to vote for one of the classes, so it is possible to have an object with the same amount of votes for person and background. The object is finally classified as background if it has less votes for person than for background or

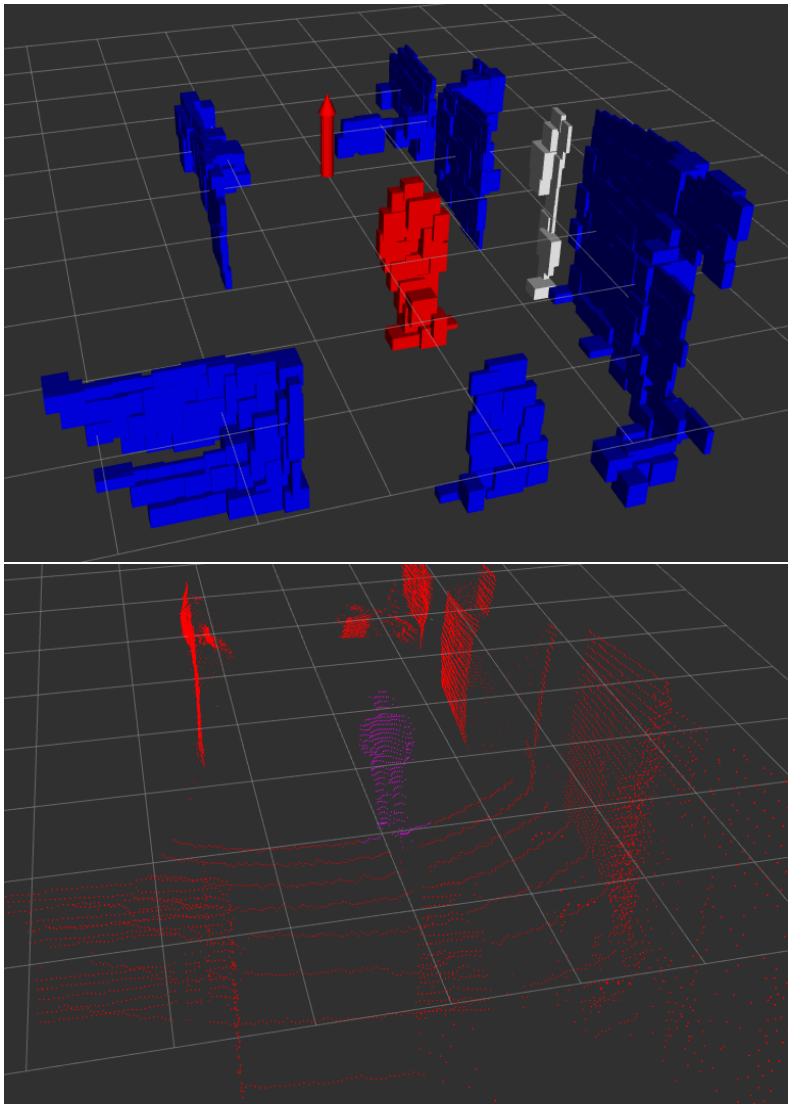

Figure 4.8: On the top, the result of classification of the objects composed of voxels (Red for persons, blue for background and white for unclassified). On the bottom the result of the classification in point clouds (violet points were classified as persons).

the same amount of votes for both classes.
The voting system is interesting since it allows for simple modular enhancements as it would be simple to add other weak classifiers if new data parameters become available. Moreover, an introduction for weighted votes is possible if the classification of each weak classifier is considered to have a different impact on the final result. For example, we might consider according to the results that shape classification is more important or gives more crucial information than the vector normals. In our prototype, every classifier has the same weight, but further customization with more time to analyze the importance of each one is possible and easy to implement.
This section is subdivided to explain each of the weak classifiers in the following order: The shape classifier, the vector normals classifier and the shadow classifier at the end. An example of the results can be seen in Figure 4.8.

4.4.2 Shape classifier

Once the object list is created the next step is to determine the dimensions of them. In our solution, we find the limit dimensions of each object. This means that we have for each object the maximum width, length and height. Although 3D point clouds allow for a more refined geometric description of the objects it would require a layered approach to get the dimensions at different parts of a person. For our application in not too cluttered environments and indoors our limited representation of the dimensions give good results when determining between only two classes: person and background while keeping the processing time to a minimum.
Using the shape of the segmented objects, it is easier to see when an object is not a person than to decide it is. Simple predefined thresholds are set to determine an object is not a person by shape, like for example that if the width or length of an object are larger than the height the object would certainly not be a person but a wall or other big horizontal object. These clear and simple thresholds give good results for discarding an object as background.
To determine if an object is a person is more complicated because the variability in poses could get to be overwhelming when dealing with predefined binary thresholds. This is why we consider our shape classifier to have the minimum conditions for an object to be recognized as a person, but could fail in the event of extreme poses like crouching or doing exercise like jumping-jacks. On the other hand, we find that for these same reasons our classifier is less prone to fail in cases when the person is carrying a suitcase or bag in their hands or when wearing backpacks or hats. This is a problem that usually affects approaches that use supervised models for classification due to generalizing problems when training the model. Between the predefined thresholds to be voted as a person are minimum and maximum values for height, width and length. Such as a person can not be taller than 2.3m normally, can not be wider than 1.8m (completely extended arms) and the other side must not be very long at all compared to the other dimensions.
After the shape classifier, each object has one vote for either background or person, as it is the only weak classifier that always votes for every object.

4.4.3 Vector normals classifier

Another way to classify objects is by doing an analysis on their normal vectors, as explained in the Voxelization section, a normal vector was calculated for each super-voxel and an example of the results can be seen in Figure 4.9. With this information, once the objects are segmented into a group of super-voxels we can calculate the average contributions of each component of the normal vectors. In this way we can compare the proportion of the contributions.
In an office environment, most of the objects we can find are walls, or objects with the same type of characteristics like divisions or lockers. In these objects, the contributions of the X or Y components of the normal vector are often predominant. For our classifier, we consider that any object with an average X or Y component of their normals greater than 72 percent corresponds to an object with normals parallel to the floor and therefore walls/divisions/lockers and are voted as background. This classifier also helps to correct some of the issues that may occur with the shape classifier. For example, it is often conceivable that a wall corner or small division could appear as a person to the shape classifier as it only deals with limit dimensions, but it would be voted as background from the normal classifier.

Another interesting application for the normal vectors is to remove the super-voxels that belong to the ground. Since eliminating the ground helps to reduce the total amount of points to be processed and it is not of our interest it is eliminated at the time of the construction of the super-voxel. Ground removal could also help in segmentation, since it can be used as separation between objects.
Finally, it is important to say that this classifier only has criteria to classify an object as background but there is not enough clear and descriptive information for classifying people with their normals. Due to different poses, positions and orientations of the different limbs of a person the average components of the normals do not follow a proper distinguishable distribution. Overall, we have three weak classifiers in total among which the only one that always votes is the shape classifier. This means that we might face a situation where instead of three votes, an object could have only two. In this case, if there is one vote for background and another for person (could happen when shape classifier classifies as person and normals classifier classifies as background) in the final result that particular object would not be considered as a person as there is not enough information and would have to wait for the following scan to try for a more precise classification.

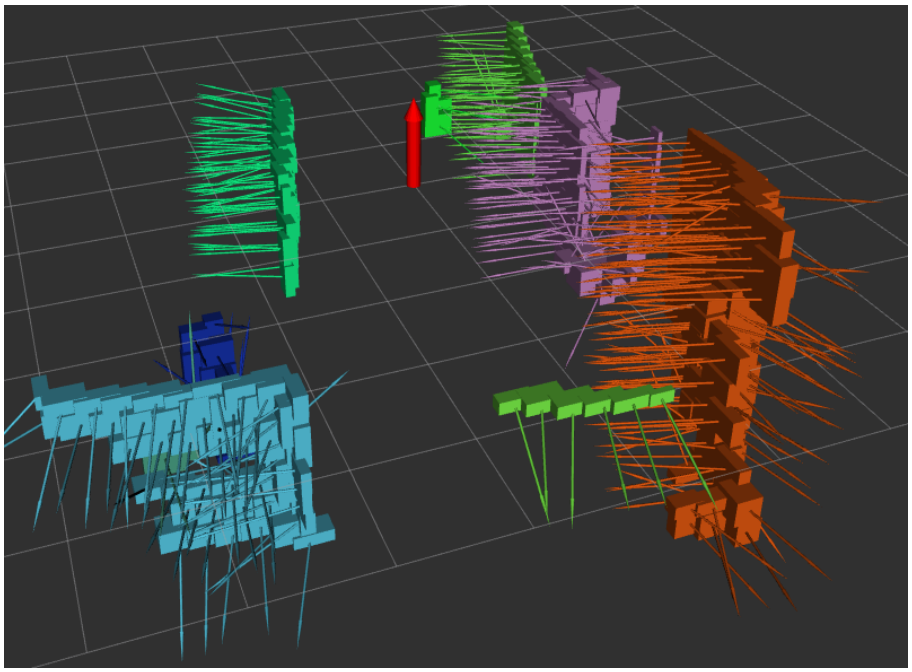

Figure 4.9: Super-voxels with their vector normals

4.4.4 Shadow classifier

The last two weak classifiers do most of the classification work, the shadow classifier is implemented to solve a problem that became evident in the testing stage. But first let us define what are considered shadows for us and how they can affect the results. Following that, we explain the process for the classification.

Shadow definition

In the context of any vision and range sensor we have to deal with occlusion problems. In the case of LiDar 3D sensors, when a beam hits an object it can not continue traveling so everything behind that object in a projection is going to be occluded and it will affect the apparent shape of the background, this is what we denote as shadows. Normally this wouldn't pose a big problem in outdoors or big indoors environments if the sensor is far enough from the targets. However, in our situation we noticed some strange false positives created from the distortion caused by the shadows of a moving person.
Despite the fact that the shadows themselves can not generate miss classifications since they are the absence of points behind a moving obstacle, they could generate distortions on the background like separating two formerly clearly defined background objects into objects that might be confused as people. This particular issue happens more often when doing entire scan classification before creating the background model and could result in a miss classification of a background object as a person and might disturb the background model calibration. Furthermore, when doing movement detection, if the shadow is big enough (depends on how close the target is from the sensor) it could generate sufficient discrepancies to create apparent dynamic voxels that will clutter the segmentation and classification algorithm leading to a decrease in performance.

The classifier

To solve this issue we implemented another weak classifier specifically for determining if an object classified as a person is in reality the product of a shadow disturbance in the background. Therefore, if an object has more votes for person or the same amount for both classes we check if there was another moving object directly in front of it in the occlusion line. For any object in the scan that is in front of other objects, the occlusion line for that object would be the projection the beams would follow if the object was not there. If the classifier determines that in fact the object falls in another moving object's direct occlusion line then it is considered the result of a shadow distortion and it gets one vote as background. This method is similar to the explicit occlusion adaptation method done in [19] but in that case it was intended to not lose track of moving objects partially or totally occluded by others. Whereas for us is to deal with discrepancies that the shadows create in the background. Overall, this classifier corrects some possible errors that could happen with these types of sensors specifically for an application indoors where the sensor is close to the moving targets and it helps the general implementation to have better performance. In future works an implementation with this principal for dealing with target occlusions could also be implemented.

4.5 Temporal integration and tracking

Until now we have gone through all of the stages of our prototype mostly on a scan to scan basis. But it is important to transform everything to work progressively over time. When a person is detected, identifying them as the same person in the next series of scans can help us improve performance and the quality of the results as we are able to correct possible mistakes from the classification process. As mentioned before, the most common mistake in classification are the

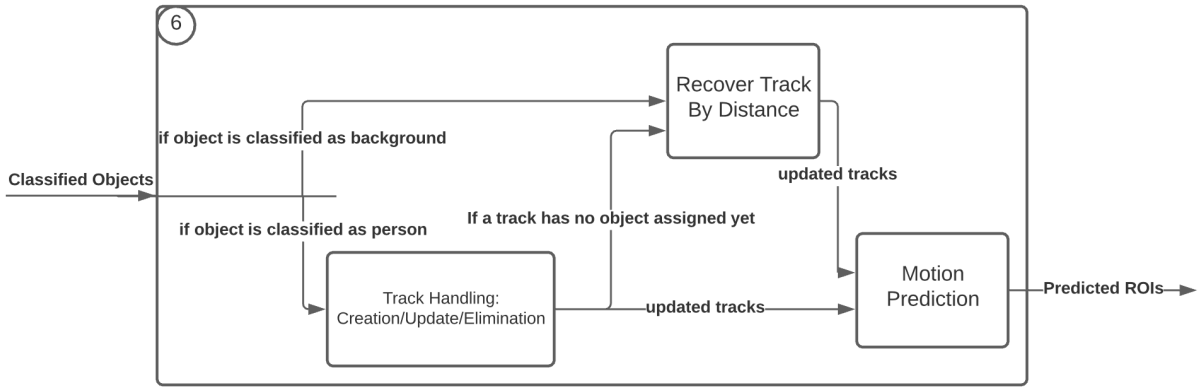

Figure 4.10: Pipeline of the tracking module. The objects classified as persons are assigned to the corresponding tracks. If a person from a previous track is miss classified, the recover by distance method should correct it. The output of this module are predicted ROIs that are also going to be input to the voxelization module in the following time step.

false negatives, but with a proper tracking solution like GNN [17] we can compensate for these errors. The pipeline for this module is shown in Figure 4.10.
This section is subdivided to explain each one of its different steps, in the following order: Track handling, the recover by distance method and lastly, and the motion prediction stage to extract ROIs.

4.5.1 Track handler

This subsection is composed of three fundamental parts of any track handling: Track creation, track update (data association) and finally, the notion of tracking attempts and track elimination. A pipeline of this process can be seen in Figure 4.11.

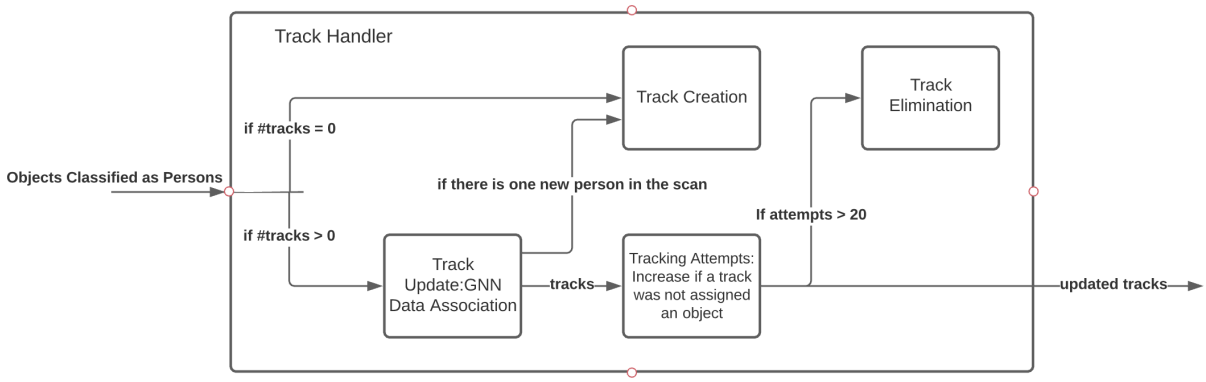

Figure 4.11: Pipeline of the track handler. The input is all the objects in the ROI classified as persons. The output is the tracks updated. The track handler has all the information about the current tracks at all times.

Track creation

After an object is classified as a person for the first time, the track creation process starts. But in order to introduce some redundancy and minimize the effects of possible mistakes in classification we do not confirm the track until the object is classified a specific number of times in consecutive scans. This number is called the Track Threshold and can be set according to the level of confidence on the classifier. For us the Track Threshold is of 3 consecutive scans. The track creation is straightforward if there are no tracks initially, as it would only have to create the tracks for detected persons and then confirm them in subsequent scans. If the track threshold is not met after a specific number of scans then the track is not confirmed and therefore eliminated from the track list. This number of scans can be set to be equal to the track threshold in which case it is a strict condition and also depends on the level of confidence on the classifier. For a more flexible approach the limit for number of scans could be greater than the track threshold. For example, the track could be confirmed if it is classified as a person 3 times in the next 5 scans. For our specific case we find that setting these two parameters to the same value helps to be sure of the track confirmations and also dispose of the ones that were created by a miss classification.

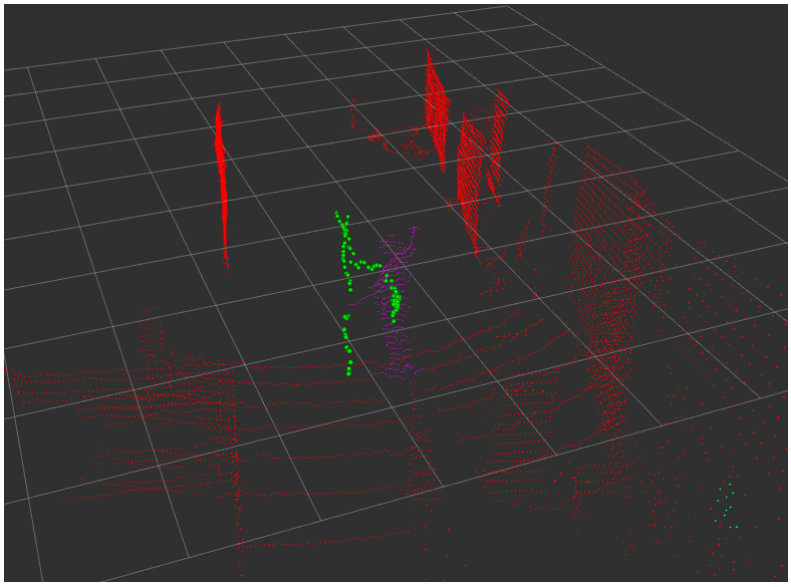

Figure 4.12: One example of a track trail. Points classified as persons in violet, background in red. In green, the last recorded centroid position of the person in the track trail.

Track update

When we already have confirmed tracks in the track list, in the subsequent scans we must update them. Updating a track means to find an object that corresponds to it at any given scan. In this section we are concerned only by the objects that are classified as persons in a particular scan. In the section recover by distance, we explain how to recover a track when a false positive classification happens.
The update method relies on the principle of the Global Nearest Neighbor method for tracking. Basically this method states that the object that is closer to the last known location of the

tracked object should be assigned to the same track. This relies on the principle that we are dealing with persons and scan after scan they must be near their last location, a person could not simply disappear.
When assigning objects to tracks by distance, it is necessary to introduce a notion of maximum inter-scan translation distance. This is done because an object can only be too far from the last recorded position in a certain amount of time. When running at 10Hz (full speed), the inter-scan time is 0.1s, and the maximum translation distance is set to 0.5m that corresponds to the average speed of a running person. A running speed is a limit case off course, as most persons in an environment such as an office would not be running at high speeds but mostly just walking.
When a person is assigned to a track, all of the important data is recorded on the track list. The track list contains up to 30 scans of information and a typical trail for a track can be seen in Figure 4.12. Among which we have object dimensions, the last shift of the object and the last tracked 3D centroid of the object.
The shift of a person at each scan is equal to the difference between their last recorded centroid and the new one. This parameter provides a notion of how fast an object is moving between scans, and it is helpful to predict the object's next position assuming the person has constant velocity.

Tracking attempts & track elimination

Every time a track is not assigned a suitable object at any given scan, the track is not eliminated right away. As it is standard for tracking methods, the loss of a track object in a scan (or in frame for traditional vision applications) is not rare and should not be considered of enough importance to eliminate the track, instead the number of attempts goes up by one until an object is yet again assigned to it. In fact, the only way a track is considered completely lost and therefore eliminated from the track list is when an object is not assigned to it after several consecutive attempts. In our prototype, a track is eliminated after a maximum number of attempts equal to 20 scans (running at full speed at 10Hz it would mean 2s).
In fact, to get the predicted ROIs when tracking an object, or when predicting the position of an object when trying to recover by distance the tracking attempts play a crucial role to ensure the robustness of the tracking method. Every time a prediction based on the last recorded shift is done, the prediction method has to take into account the number of attempts of each track. This is evident since if after one scan a person might have moved the same as the last recorded shift, after two attempts the person could have moved double that amount.
Furthermore, another addition to make a reliable tracking method is the condition to increase and reset the tracking attempts. Normally, the tracking attempts would be reset to zero if an object is assigned to the track. Nevertheless, in our approach there are two ways to assign an object to a track: either by finding a suitable person or if no person is found for a track, we implement a recover by distance method. This method finds a suitable object for the track, this object was not originally classified as a person. This is done to take into account extreme pose changes and classification errors. This method is explained in detail in the next section.
Therefore, we make sure to reset the number of attempts only when the object assigned to the track was classified as a person by our generalized classifier. We make sure to update the tracked center only in this case as well. This was made to ensure that a mistake in the recovery

by distance method would not be fatal for the track and the person could still be found in the following scans.

4.5.2 Recover by distance

As mentioned before, our solution relies for the most part on the classification module. This means that at each scan ideally we would want to segment and classify the person even if they are already being tracked. It is done in this way to make sure we get a proper segmentation of the person at each scan since the sparse points can change significantly from one scan to another. And the classification at each scan introduces redundancy to be sure that what we are tracking is indeed a person. But relying completely on the classification results to update the tracks would imply that our classifier is almost perfect, which of course it is not. Although we have a high confidence that the majority of positive classifications are true positives, the classifier is prone to give a high amount of false negatives because of the wide variety of pose changes a person can experience as it was discussed in chapter 2.
To overcome this issue we introduce a method to recover a tracked person by distance. The objective of this method is to assign an object that was not classified as a person to a track that was not assigned a person in the track update method. To do this, several techniques are applied to ensure that it can find the best suitable candidate for the track according to their predicted position (assuming constant velocity). The principle of the recover by distance method is shown in Algorithm 3.

Algorithm 3: Recover by distance

inputs : List of tracks TL that were not assigned a person in the track update method, list of segmented objects OL
output: Updated list of tracks

```
1 for each track in TL do
2     Get predicted position of the track based on the last recorded velocity and
        orientation (assuming constant velocity);
3     Create a gating area with a radius of 0.5m around the predicted position;
4     Look for an object in OL that is inside the gating area;
5     if object found then
6         if the object has the minimum conditions to be considered as a person then
7             Assign object to track;
8 end
```

Since the object list input contains every segmented object in the regions of interest, line 6 of Algorithm 3 is there to introduce more confidence in the recovery process by checking that the object has the minimum conditions to be considered a person.
These characteristics are less demanding than the ones used by our classifier and are aimed to be used only when the first classifier failed to take them into account. These are new thresholds for shape conditions, and they are trying to take into account extreme pose cases that our generalized classifier does not. Despite having an extreme pose change, the object still has to have a minimum set of conditions for it to be considered a person. Between these minimum conditions are: vector normal threshold (to discard the object being a wall) and a maximum height, length and width.

This method solves most of the extreme pose cases presented in chapter 2, along with the problem of a person appearing chopped or incomplete when standing close to the sensor because of the restrictive vertical field of view.
Even though our generalized classifier is not able to recognize the person as such in these cases, if the person is being tracked, the recover by distance method can still accurately detect the person. An example of a person being detected while crouching is shown in Figure 4.13. The only limitation of this approach is that the person should be in a track already in order to be recovered. More examples of the results of detection in difficult poses are presented in the next chapter.

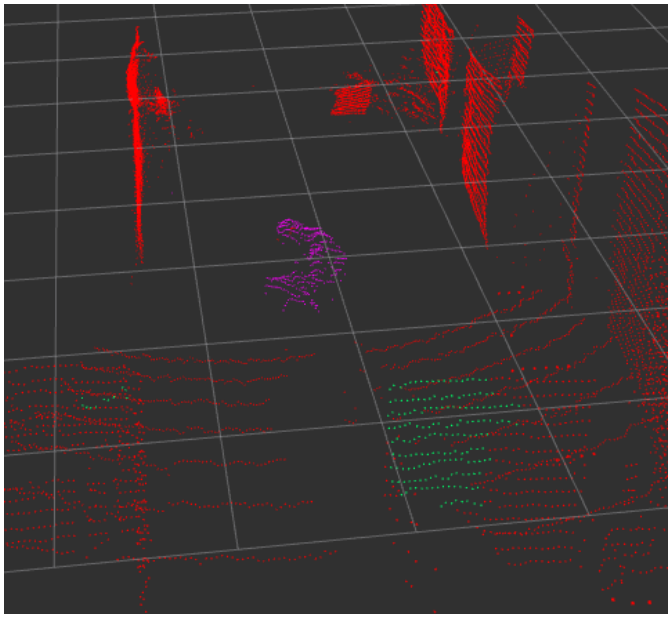

Figure 4.13: Classification of a person while being crouched. Red points correspond to the background, violet points to a person, and green points correspond to the result of the movement detection.

4.5.3 Motion prediction: predicting ROIs

Now that we have introduced the notion of object shift, we can explain the use of predicted ROIs. As of now, we stated that ROIs are recovered from the movement detection algorithm. But that is not the only method to extract them. Although movement detection is done perpetually to detect new moving objects, once an object is classified as a person and their track is created and confirmed, for the following scans we can predict an area where that person could be.
At each scan, we do movement detection to extract the first set of ROIs, and then we iterate through the track list to extract new predicted ROIs. For each track, we predict the new position of the object taking into account the last recorded centroid and the last shift assuming that the person has a constant velocity. Due to the fact that we are dealing with sparse point clouds, a ROI is created for each predicted centroid in a spherical area around it. In this way, we make sure to include all of the possible points that might belong to the person after the translation and

include them into the dynamic voxel list. In Figure 4.14 we can see an example of a predicted ROI, we can see that when given a big enough radius around the predicted centroid the ROI includes the person.
The reason why we do this is that our prototype solution relies heavily on classification, as we aim to segment and classify the objects in the ROIs even if they are being tracked. Usually other solutions would rely on data association and tracking techniques to find the tracked object after the first detection, but when dealing with sparse point cloud data segmentation is crucial for the entire solution. For this reason, we carefully select the ROIs (either by prediction or by movement detection) and then segment and classify every object in them scan by scan. These groups of ROIs form the point cloud that the algorithms for voxelization, segmentation and classification work with. At the end of the classification stage we have a list of objects classified as persons, which is the one used to update the tracks as explained in the Track Update section.

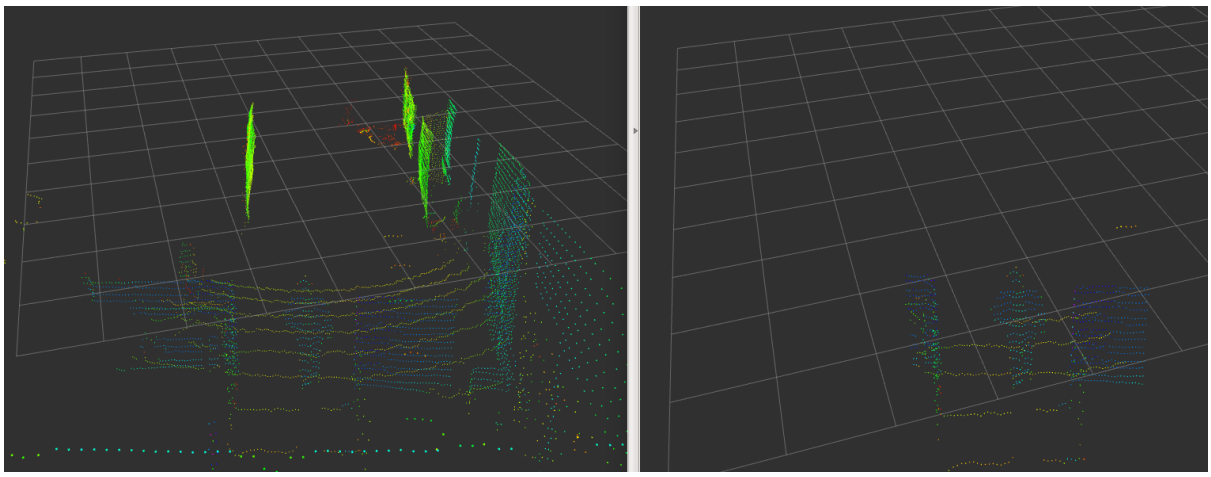

Figure 4.14: On the left the raw data, on the right an example of a predicted ROI

4.6 Summary

In conclusion, our implementation relies on a robust method for segmentation of the objects in the regions of interest. The regions of interest are extracted from movement detection and motion prediction. Furthermore, the objects are classified and assigned to their corresponding tracks. In case of a miss classification our recover by distance method can still assign the correct person to the correct track relying on motion prediction to calculate an estimated position. All of this allows for a robust implementation against pose changes, while also achieving real-time performance. In the following section the results of the evaluation are analyzed.

— 5 —
Results

This section is aimed to discuss the results of our implementation. Our prototype consists of two branches: a dedicated single person solution that serves as a baseline prototype and a dedicated multiple person solution that works as intended in principle but there are still many aspects that could be improved to enhance results and performance. To present the results, we start with the most simple cases up to the more complicated ones. Videos for the results of all of the test recordings presented here can by found by clicking here: Video Access or going to the url: `https://lig-membres.imag.fr/aycard/html//Projects/JuanGomez/JuanGomez.html`.
It is worth mentioning as a disclaimer, that due to time restrictions the results are presented and analyzed in a mostly qualitative way. In spite of this, quantitative results are presented using a simplified way to compare the ground truth to our prototype results in order to introduce more credibility to our evaluation. In this approach, the ground truth is not the result of manual annotations of the exact position of every person made scan by scan since this process would be very time consuming because the entire test set is taken by us in our laboratory trying to mimic an office environment situation. Instead, we present generalized results by checking how many times the person is not correctly tracked (false negatives) along with false positives and true positives, and how many times the track is lost for each of the situations.
This chapter aims to explain and analyze the results of the evaluation of our prototype with each of the test recordings collected. They aim to cover different possible situations such as extreme pose changes and occlusions.
The outline of this chapter is the following: First, we introduce the definition of the metrics we have used for the evaluation. Then, we describe and explain each one of the test recordings used for evaluating our prototype. Followed by the results for the single person tracking branch. Finally, we analyze the performance of the multiple person branch in both the single person recordings (for comparison purposes) and a final one that contains two persons on it.

5.1 Definition of the evaluation metrics

This section is used to remind the definition of the metrics that we use to evaluate the performance of our prototype. The metrics, in order of revision are: Accuracy, Precision, Recall, F1 Score and Frequency.

- Accuracy: This metric is defined as the correct number of predictions over the total number of predictions. This metric is useful for applications with a balanced class distribution. In our approach, it is useful to know the accuracy of the solution with regard to the class of person. In summary, we use it to calculate the ratio between the true positives and the total number of frames the person was visible in the scan. At the end, it gives a measure of how often the person was correctly classified when present in the frame. The formula is as follows:

$$Accuracy_{person} = \frac{TP}{GT_{person}} \tag{5.1}$$

Where TP is true positives and GT_{person} stands for the ground truth of the class person, which in our case refers to the total number of frames the person was in the scanning range.

- Precision: This metric is defined as the samples correctly predicted from a class over the total number of samples predicted as that class. It is helpful to see how confident the prototype is when it comes to classifying a specific class. In our case, the class of interest is the person. The formula for precision is the following:

$$Precision_{person} = \frac{TP}{TP+FP} \tag{5.2}$$

Where TP and FP are true positives and false positives respectively.

- Recall: It is defined as the ratio of samples from a class which are correctly predicted by the model. This metric is important in our application to see how our prototype is affected by false negatives. False negatives correspond to how often a person is not correctly detected as such. The formula:

$$Recall = \frac{TP}{TP+FN} \tag{5.3}$$

Where FN stands for false negatives.

- F1 Score: For our application both recall and precision are important, so the F1 score is a way to combine precision and recall. It is defined as the harmonic mean of the two. The formula:

$$F1_{score} = \frac{2*P*R}{P+R} \tag{5.4}$$

Where P and R stand for precision and recall respectively.

- Frequency: Finally, the frequency metric measures at what frequency the output of our system is running. The operation frequency of the sensor is $10hz$, which means that the raw input data runs at this frequency. It is interesting to see how fast our solution can run depending on the situation. It is defined as the total number of processed frames over the total amount of frames in the original test set. The formula is as follows:

$$Frequency_{output} = \frac{PF}{TF} \tag{5.5}$$

Where PF and TF are Processed Frames and Total Frames respectively.

5.2 Description of the test recordings

For the purposes of this evaluation we recorded 3 test sets that each dealt with a specific situation to evaluate our prototype. The test sets are the following:

- Baseline test recording: This set is aimed to be the simplest, this should be the easier case for our prototype to handle. This set involves a single person present from the beginning of the recording walking in an office. It does not involve any extreme pose changes or occlusions. The only complexity of the situation is that the person goes out of frame once for about 3 seconds and then comes back.

- Pose changes test recording: This would be considered a challenging data set for many other detection and tracking solutions, since it involves a lot of extreme pose changes. Extreme pose changes are not the regular pose changes that take place when a person is walking, but sudden changes in body parts position and orientation that would certainly confuse a lot of classifiers. In this test recording a single person walks around the same office environment, introducing sudden and unexpected pose changes such as crouching to tie their shoes, doing different types of warm-up exercises such as jumping jacks and stretching the arms in every direction. The person also picks a box from one side and places it on another.

- Occlusions test recording: Occlusions are another topic of discussion in the community of detection and tracking of moving objects. For evident reasons, they present a challenge in tracking and many works devote themselves explicitly to account for occlusions as we have seen in [19]. Unfortunately, our solution does not deal explicitly with occlusions but it is still interesting to see how it performs in this situation. For this test recording, a big obstacle was placed in the middle of the office.

- Two persons test recording: This test recording was limited to two persons at the same time and it serves as a proof of concept for our prototype. The recording was kept simple, with no occlusions caused by external objects and without any major pose changes. The only occasional occlusion was caused by each other. Furthermore, each of the targets went out of frame once and then came back after a few seconds (more than 2 seconds outside the frame).

5.2.1 Organization of the videos

All of the recordings are available in video format here: Video Access or going to the url: `https://lig-membres.imag.fr/aycard/html//Projects/JuanGomez/JuanGomez.html`. In this page two distinct sections are found: One for the single person branch and one for the multiple person branch. In every video we can find at the left the raw input to the prototype and at the right the results are shown in real-time. In the right, red points belong to the background, violet to the tracked person and green for the movement detection. The layout of the sections is the following:

- Single person branch: The first 3 test recordings were used to validate this branch. The videos in this section are: $Baseline_{single}$, $Poses_{single}$ and $Occlusions_{single}$.

- Multiple person branch: The 4 test recordings were used to validate this branch. The videos in this section are: $Baseline_{multiple}$, $Poses_{multiple}$, $Occlusions_{multiple}$ and $TwoPersons_{multiple}$.

For reference purposes, in the following section presenting and analysing the results we refer to the videos in the same notation as presented here.

5.3 Single person branch results and analysis

Firstly, we evaluate the branch of our prototype that focuses only on single person tracking. The main difference between this and multiple person tracking, is the fact that when we know that one person maximum is going to be on the scan at any given time, processing time can be decreased significantly. For example, in this case the movement detection module only has to be done when no person is detected by the classifier at the beginning of the recording. Once movement is detected, the person is segmented and classified as such, the tracking begins and the movement detection stops since we can be sure that we are tracking our only target. The biggest advantage of this branch is a reduced computational time, so it runs almost at a full 10hz (same frequency as the input data). If the track is lost completely, it is eliminated and movement detection starts once more until the person is yet again classified. At which point a new track is created. First we analyze the results for a simple case, and then for more complicated situations like pose changes and occlusion.

Video	Accuracy(%)	Precision(%)	Recall(%)	F1 Score(%)	Frequency(*hz*)
$Baseline_{single}$	81.42	92.21	87.44	89.76	9.27
$Poses_{single}$	91.15	97.15	93.66	95.37	9.34
$Occlusions_{single}$	45.94	87.88	49.05	62.96	9.06

Table 5.1: Metric results for the single person branch.

5.3.1 Baseline test recording

Overall the results are satisfactory, qualitatively the person tracking seems very consistent during the entire recording as it can be seen in the video $Baseline_{single}$. In spite of this, the length of this test recording is the shortest out of them all. For this reason some quantitative results might seem lower than they should be. In Table 5.1 we can see the results and the calculated metrics.
In the results we see an alarming amount of false negatives, these happened when the person went off frame for 7 seconds. Because our track handler tries to find the target for no more than 2 seconds, the track was eliminated. When the person comes back, movement detection correctly captures the movement but the tracking does not start until after some seconds of the movement detection. The amount of time between the movement detection and the beginning of the tracking depends on how fast the classifier can correctly identify the moving object as a person. In this case, it took the classifier a couple of seconds to be sure that the moving object

was indeed a person.
In this case the accuracy is not as high as expected, but if the recording was longer, the effects of the delay between movement detection and beginning of the track would be far less impacting in the accuracy results. On the other hand, the precision result is good as expected and the recall is also affected by the amount of false negatives. In Figure 5.1 we can see an example of detection for this situation.

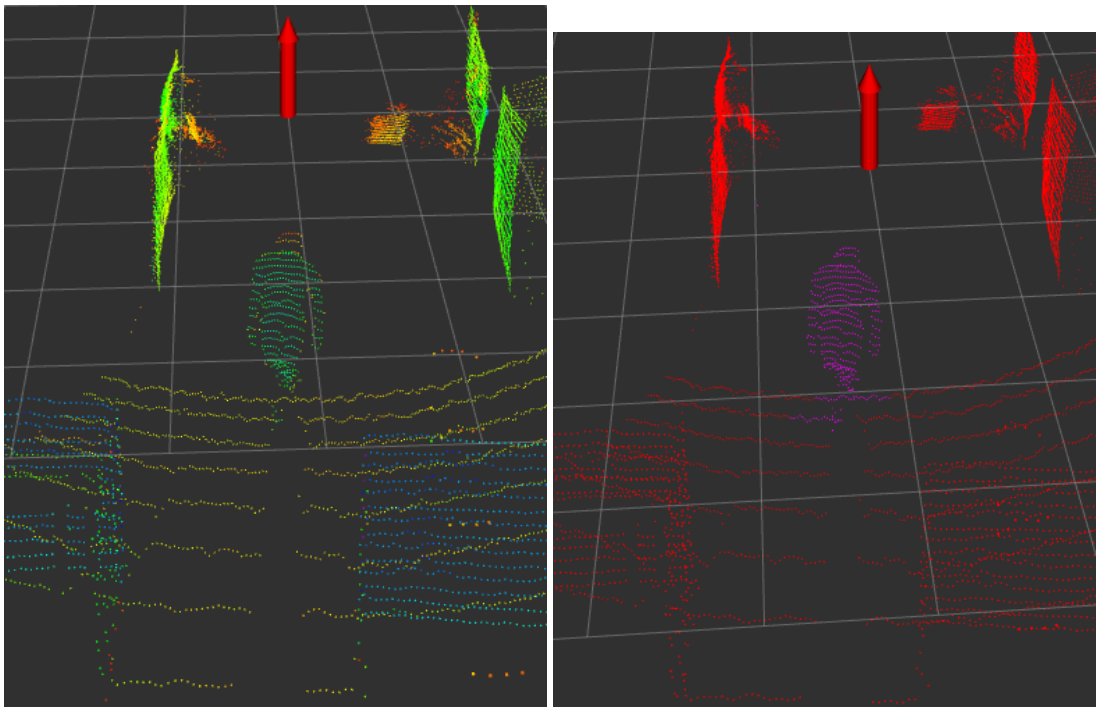

Figure 5.1: Detection results for the baseline case. On the left the raw input data and on the right the detection results.

5.3.2 Pose changes test recording

Overall, the prototype does a fantastic job with the tracking as we can see in the results in Table 5.1 and in the video $Poses_{single}$. Due to the fact that the recording is longer than the last one and that the person only went out of frame once, the overall impact of the false negatives is less, which means better results for recall. For the same reason, the number of true positives was higher and therefore the accuracy and precision are also higher. The false negatives amount is about the same as in the last recording as the person also went out of frame once, causing the track to be lost once and the delay between movement detection and track start is counted as false negatives.
As we can see from the results, our prototype does a great job when dealing with the extreme pose changes presented in chapter 2. Most of the time the person was correctly tracked even when crouching, jumping, stretching and doing a pick and place activity. During the pick and place, we can see that the track is maintained correctly when the person is crouching to pick up the object, and is also maintained when the person is holding the box. Some examples of the

results of detection in extreme poses can be seen in Figure 5.2.
A person holding accessories is also a point of discussion for classification modules in the literature, since most classifiers are trained with a supervised train set of normal people, when a person is holding an accessory like a bag or an umbrella the classifier might fail. In our solution, the combination of our generalized classifier and the recover by distance method result in maintaining the track even when the person is holding a big box into their chest.
In conclusion, this challenging test recording can show the potential of our prototype, as it has the best results out of all the situations even though is a difficult one.

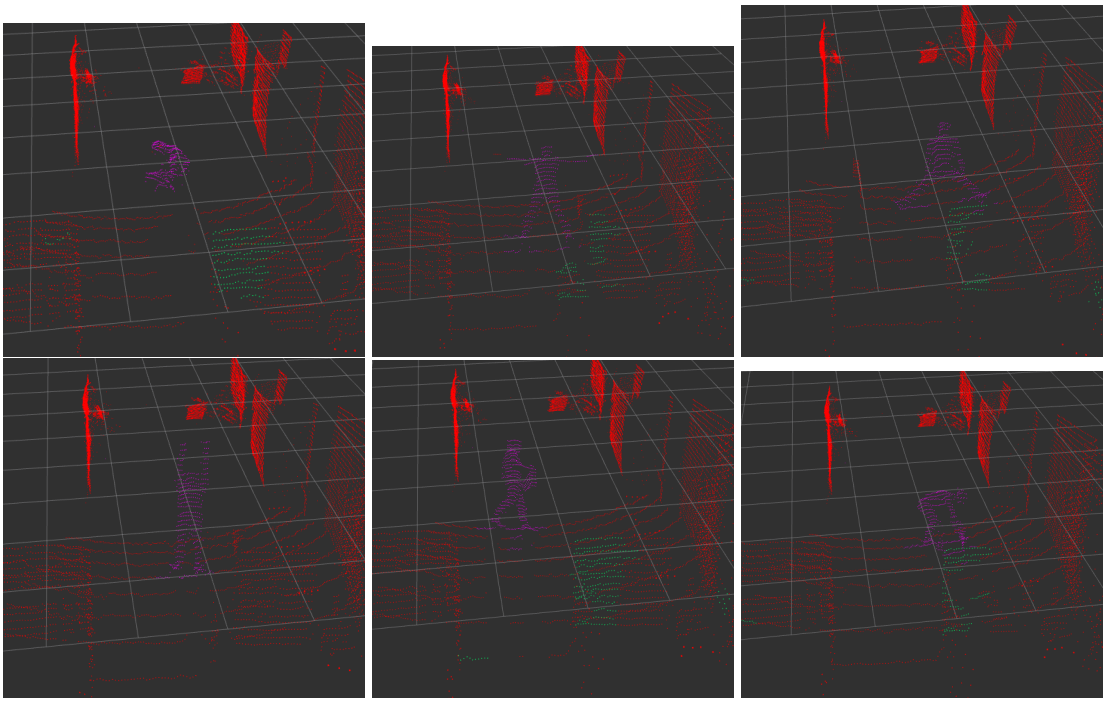

Figure 5.2: Detection results for extreme pose changes. Ranging from crouching, exercising, stretching, holding hands up to pick and place actions (in that order). Red points belong to the background and violet points to the detected person.

5.3.3 Occlusions test recording

As we can see in Table 5.1, the results are not very good, as it was expected. The only good metric was precision, because the number of true positives is still far superior than the number of false positives. In fact, we see that our prototype is very robust in terms of positive classifications. Even though we might have some false positives, they never confirm a track because of the redundancy that we integrated in the track handling. When we get a false positive, a track is created but never confirmed so is quickly discarded. Even with occlusions the confidence of our solution remains consistent for positive classifications, as we can see by the precision metric. The prototype is still able to sometimes correctly detect a partially occluded person thanks to the recover by distance method. In Figure 5.3 we can see some results for these cases.
On the contrary, the other metrics are very inferior to the other situations, as the amount of false

negatives is overwhelming due to the fact that the big obstacle occluded the majority of the area behind it and the person was lost at those moments as we can see in the video $Occlusions_{single}$. Overall, we can see that an explicit solution for occlusions necessary for future works, an approach similar to the one presented in [19] with reverse laser beam tracing to spot occlusions could be successfully implemented.

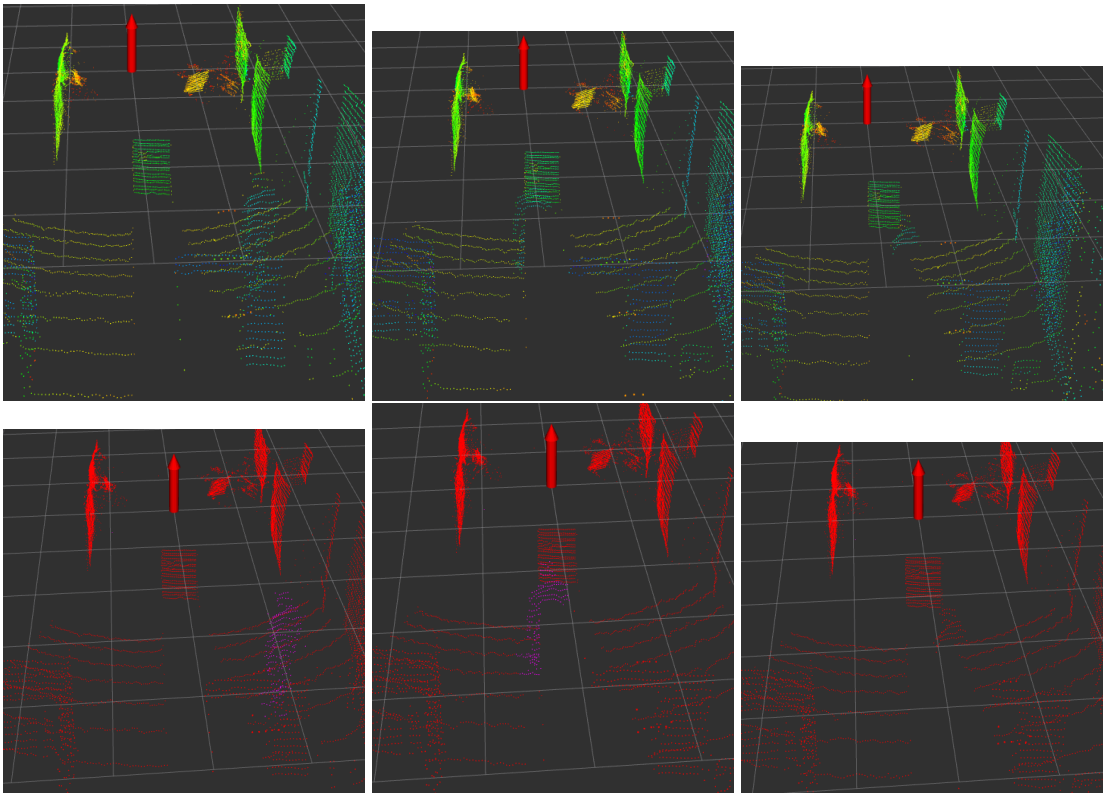

Figure 5.3: Detection results for occlusion cases. On the top we can see the raw input data and on the bottom the corresponding detection results.

5.4 Multiple person branch results and analysis

Contrary to the single person tracking branch, this branch implements track handling and data association techniques to find which object belongs to which track as explained in the Methodology section. Furthermore, the movement detection module is always active as even if a person is already being tracked the solution has to take into account possible new moving objects coming into the scan range. Besides, for every moving object in the scan a region of interest has to be predicted, extracted and processed. Therefore computational time increases proportional to the amount of moving objects in the stream of scans. First we analyzed the results of this branch for single person tracking situations, as it is interesting to see how a more complex generalized implementation performs with simple test recordings. After that, we present results for multiple people tracking up to two persons at the same time to show a proof of concept of

our solution.

5.4.1 Single person test recording

Overall, as we can see in Table 5.2 the metrics for all of the single person tracking situations were either very similar or slightly superior than the ones from the single person branch. Particularly in the Occlusions Test Recording we see a significant improvement of the metrics. It is good to see that the generalized solution in this multiple person tracking branch works very well and even better according to the metrics since it shows the potential of our prototype. The results can be seen in the videos $Baseline_{multiple}$, $Poses_{multiple}$ and $Occlusions_{multiple}$.
Even though the metrics are better, this branch presents some problems that were not a concern with the single person branch. For example, a speed performance decrease is evident as it was to be expected since the movement detection module is active at all times in order to detect new moving objects coming to the frame. One particular situation that causes computational time to increase, is that when the person gets too close to the sensor they occlude a large amount of the laser beams. Therefore, the background changes a lot with respect to the background model resulting in a lot of discrepancies between the two that lead to more points and voxels to process.
On top of this, we also see that in contrast to their single person branch results counterpart, the first two test sets present one and the third set presents two additional track losses. This track loss should not happen as there is only one time when the person goes out of frame long enough to lose a track. This could be corrected by improving the data association techniques used for assigning moving objects to each track. Despite this, the person is still correctly classified at the same rate as in the single person branch, but imperfections in the track handling also deteriorate the speed performance.

Video	Accuracy	Precision	Recall	F1 Score	Frequency
$Baseline_{multiple}$	83.2	94.86	85.13	89.73	8.6
$Poses_{multiple}$	94.04	94.88	96.49	95.68	8.61
$Occlusions_{multiple}$	69.50	90.18	72.24	80.22	7.84
$TwoPersons_{multiple}$: Target1	89.39	93.65	95.16	94.40	6.80
$TwoPersons_{multiple}$: Target2	74.90	93.91	78.72	85.65	6.80

Table 5.2: Metrics results for different situations with the multiple tracks branch.

5.4.2 Two persons test recording

The results in Table 5.2 are presented for each one of the targets individually. Overall the results are satisfactory as it can be seen in the video $TwoPersons_{multiple}$. For the first target we find that the amount of false negatives is extremely low, so basically it was correctly tracked almost at all times. Therefore, the metrics for this target are good and consistent. This particular target only went out of frame once, so the correct amount of track lost should be one. The second

track loss was caused at the beginning of the recording, where the target moves forward for a few scans and then suddenly changes their direction and starts going backwards. In this case the predicted center is too far from the real position as the prediction expects the subject to continue to go forward. Even though a new track is created, the target is consistently classified and tracked throughout the process.
In the case of the second target the amount of false negatives is considerably larger. This happens since at the beginning of the recording, at the stage of background modeling the classifier did not classify the target as a person, therefore leaving them in the background model at the beginning. For this reason, the target had to begin movement to be detected, eventually the object was detected and the tracking began.
For both of the targets the precision remains consistent as it does for all of the situations, meaning that our prototype has high confidence when it comes to positive classifications and tracking results.
In terms of speed performance, we can clearly see that with more than one person in the stream of scans the frequency decreases more. As we have previously stated, the more persons there are in a stream of scans the slower the processing time is. Still, we still manage to obtain a frequency of 6.8 Hz which is still considered good for 3D point clouds applications. An example of a detection result of this situation can be seen in Figure 5.4

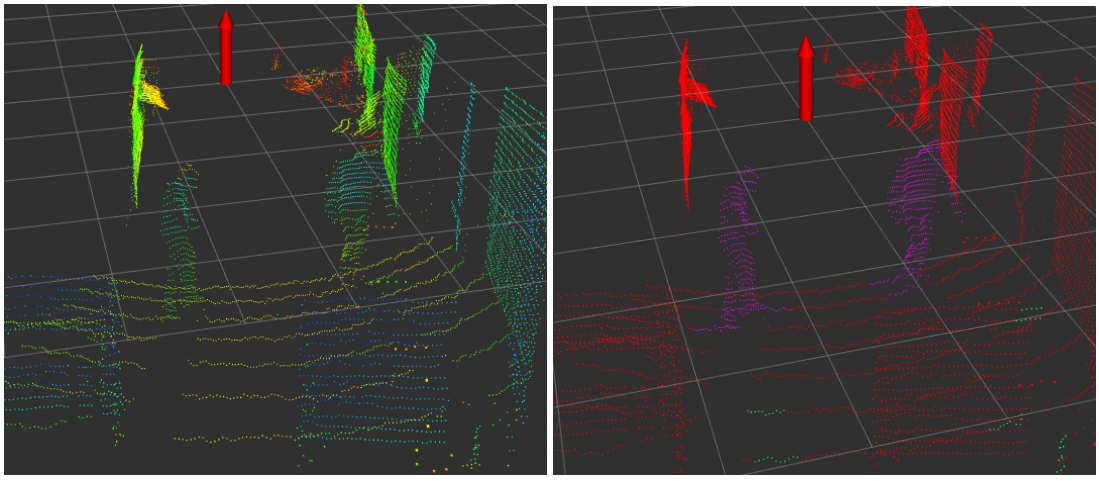

Figure 5.4: Detection result for the two person case. On the left the raw input data and on the right the detection results.

— 6 —
Discussion & Conclusion

In this work, we have presented a prototype for solving person detection and tracking in 3D point clouds in real time in a low performance machine. After discussing the different techniques for people detection and tracking with range sensors we have presented a method with robust object segmentation based on super-voxels and a chaining method. The classification of the objects plays an important role instead of being just a verification step like in other works, and it works in a simple reliable way to classify objects with minimum required characteristics to be considered persons. This classification is done based on basic geometric features.
Although 3D point clouds are normally challenging because of the large amount of data to process, the achievement of a real-time implementation is a result of the techniques of movement detection and prediction, that limit the amount of points to be processed by finding and predicting regions of interest.
Even though the classifier is not meant to deal with the variety of poses a person can assume, the integration of the classification module with the tracking result in an excellent handling of different extreme pose changes a person can undergo.
Overall, in the results section it is shown that our prototype has a high confidence on positive classifications and detection as it can be seen by the consistent results in the precision metric, even in harder situations like when handling occlusions.
For the short amount of time we had to conceptualize, implement and evaluate our solution, the presented prototype is intended as a proof of concept, where the potential is shown for a solution of this type as the results are overall very good.
Given that our prototype achieved good results in validation and shows a lot of potential for improvements in the field of person detection with 3D LiDar sensors, we intend to submit this work to the 2021 robotics conference IEEE International Conference on Robotics and Automation (ICRA) (submitting deadline in September the 15th of 2021).
With more time to expand upon this project, the current solution could be expanded with either short term or long term solutions thanks to the modular nature of it. Here, we take a look at the future prospects of our work within these two categories.

6.1 Short term rospects

This section covers the ideas we have in order to enhance the results of our current solution. As already mentioned, the current prototype serves as a proof of concept, and these ideas represent

possible immediate enhancements to polish our prototype.

- As presented in the results chapter, the track handling module could be enhanced to obtain more robust results. One possible solution would be the introduction of an Interacting Multiple Model filter (IMM) would be an improvement over the current simple model for motion prediction that relies on assuming a constant velocity and orientation of the target at each scan. A more immediate improvement would be to predict the next target position not only taking into account the last target velocity and orientation, but to do it according to all of the previous positions and velocities of the object that are saved at all time (the last 30) in the track. This initially was the idea for prediction, but again due to time constraints a simpler version was implemented.
- A more robust method for data association would be determinant in enhancing the multiple person tracking. For these types of situations, methods for data association like Join Probabilistic Data Association would handle multiple target tracking much better than the current Global Nearest Neighbor technique.
- Another important aspect to deal with would be occlusions. As it is stated in the results section, our prototype does not explicitly handle occlusions. An interesting approach for handling occlusions in 3D point cloud data from a LiDar sensor was studied in the bibliography [19], this approach traces rays from position near the person to the sensor and determines if the ray hits an obstacle and adapts the observation model of a particle filter. The inclusion of an explicit occlusion handling method would be necessary for a more robust implementation of our prototype.

6.2 Long term prospects

This section covers the ideas we have in order to expand upon our work in the event of a doctoral thesis. They involve the continuation of our work on a bigger scale, with applications such as social robotics for following robots and applications in the industrial field with human-robot collaboration. As a part of a doctoral thesis, the current proof of concept for a perception module could be expanded to work in conjunction with a robotic decision architecture that takes into account the way that the humans would react to the robot's movements. Some of the objectives would be:

- Expanding upon the geometric methods to detect and segment a person in multiple levels: The person like an unique object for applications like robot following and the person as multiple objects (like arms, head, etc) linked together for applications like human-robot collaboration in industries.
- Implementation of robust tracking methods to track a person in situations with complex dynamics and for tracking the different parts of a person with each their own dynamic.
- Introduce a second 3D LiDar sensor with less limitations to do sensor fusion to capture the person from different positions.
- Movement and behavior prediction, these predictions can be integrated into the robot's movement planning taking into account the person's movement. Therefore, the robot would choose how to move according to how the person would react [25].

- Analysis of the implication of each one of these obstacles and of the social-relational values that humans will attribute to the robot's decision, and integration of these human factors into decision-making.
- Experimentation, testing and validation of the work on perception and its integration on a mobile robot and a robotic arm.

In conclusion, our approach shows a lot of potential for future works and applications. And it shows the convenience and some of the advantages of working with a 3D LiDar sensor.

Bibliography

[1] O. Aycard A. Azim. Detection, classification and tracking of moving objects in a 3d environment. *2012 IEEE Intelligent Vehicles Symposium*, 2012, pp. 802-807, doi: 10.1109/IVS.2012.6232303.

[2] Laurent Trassoudaine. Ahmad Kamal Aijazi, Paul Checchin. Segmentation et classification de points 3d obtenus à partir de relevés laser terrestres: une approche par supervoxels. *RFIA 2012 (Re- connaissance des Formes et Intelligence Artificielle)*, Jan 2012, Lyon, France. pp.978-2-9539515-2-3.hal-00656538.

[3] Beñat Irastorza Renzo Verastegui Sebastian Süss et al. Bangalore Ravi Kiran, Luis Roldão. Real-time dynamic object detection for autonomous driving using prior 3d-maps. *First International Workshop On Autonomous Navigation in Unconstrained Environments - In Conjunction with ECCV 2018*, Sep 2018, Munich, Germany. hal-01890980.

[4] Takayuki Ikeda Tetsushi Miyashita Takahiro Brscic, Drazen Kanda. Person tracking in large public spaces using 3-d range sensors. *Human-Machine Systems, IEEE Transactions on*, 2013. 43. 522-534. 10.1109/THMS.2013.2283945.

[5] Amaury Nègre Denis Pellerin Serge Olympieff. Bruce Canovas, Michèle Rombaut. Speed and memory efficient dense rgb-d slam in dynamic scenes. *IROS 2020 - IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2020, Las Vegas, United States. pp.4996-5001,10.1109/IROS45743.2020.9341542 . hal-03143986.

[6] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, 1981.

[7] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[8] Armando Pesenti Gritti, Oscar Tarabini, JÃ©rÃ´me Guzzi, Gianni A. Di Caro, Vincenzo Caglioti, Luca M. Gambardella, and Alessandro Giusti. Kinect-based people detection and tracking from small-footprint ground robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4096–4103, 2014.

[9] Jwu-Sheng Hu, Jyun-Ji Wang, and Daniel Minare Ho. Design of sensing system and anticipative behavior for human following of mobile robots. *IEEE Transactions on Industrial Electronics*, 61(4):1916–1927, 2014.

[10] Omid Hosseini Jafari, Dennis Mitzel, and Bastian Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5636–5643, 2014.

[11] Zhizhong Kang, Juntao Yang, Ruofei Zhong, Yongxing Wu, Zhenwei Shi, and Roderik Lindenbergh. Voxel-based extraction and classification of 3-d pole-like objects from mobile lidar point cloud data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(11):4287–4298, 2018.

[12] Xiaozhi Shen Shaojie. Li, Peiliang Chen. Stereo r-cnn based 3d object detection for autonomous driving. 2019.

[13] Hengli Liu, Jun Luo, Peng Wu, Shaorong Xie, and Hengyu li. People detection and tracking using rgb-d cameras for mobile robots. *International Journal of Advanced Robotic Systems*, 13, 09 2016.

[14] Jun Liu, Ye Liu, Guyue Zhang, Peiru Zhu, and Yan Qiu Chen. Detecting and tracking people in real time with rgb-d camera. *Pattern Recognition Letters*, 53:16–23, 2015.

[15] Martial Sanfourche Guy Le Besnerais. Maxime Derome, Aurelien Plyer. Moving object detection in real-time using stereo from a mobile platform. *Unmanned systems, Word Scientific*, 2016, 3 (4), p. 253-266. 10.1142/S2301385015400026 . hal-01393423.

[16] Trung Dung Vu Qadeer Baig, Olivier Aycard and Thierry Fraichard. Fusion between laser and stereo vision data for moving objects tracking in intersection like scenario. *IVâ2011 - IEEE Intelligent Vehicles Symposium*, Jun 2011, Baden-Baden, Germany. pp.362-367, 10.1109/IVS.2011.5940576 . inria-00625511.

[17] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[18] R. Danescu F. Oniga S. Nedevschi, T. Marita and S. Bota. Onboard stereo sensor for intersection driving assistance. architecture and specification. *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, 2009, pp. 409-416, doi: 10.1109/ICCP.2009.5284726.

[19] Florian SchÃ¶ller, Jens Behley, Volker Steinhage, Dirk Schulz, and Armin B. Cremers. Person tracking in three-dimensional laser range data with explicit occlusion adaption. In *2011 IEEE International Conference on Robotics and Automation*, pages 1297–1303, 2011.

[20] Kai Triebel Rudolph Siegwart Roland. Spinello, Luciano Arras. A layered approach to people detection in 3d range data. 2010.

[21] Luciano Spinello, Matthias Luber, and Kai O. Arras. Tracking people in 3d using a bottom-up top-down detector. In *2011 IEEE International Conference on Robotics and Automation*, pages 1304–1310, 2011.

[22] Nils Appenrodt Trung-Dung Vu, Olivier Aycard. Online localization and mapping with moving object tracking in dynamic outdoor environments. *2007 IEEE Intelligent Vehicles Symposium*, Jun 2007, Istanbul, Turkey. inria-00194152.

[23] Olivier Aycard Trung-Dung Vu. Laser-based detection and tracking moving objects using data-driven markov chain monte carlo. *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3800-3806, doi: 10.1109/ROBOT.2009.5152805.

[24] Olivier Aycard Trung-Dung Vu, Julien Burlet. Grid-based localization and local mapping with moving object detection and tracking. *Information Fusion, Elsevier*, 2011, 12 (1), pp.58-69.10.1016/j.inffus.2010.01.004 . hal-01023076.

[25] Aubergé V. The so called âsocio-affective robotâ: a tool to understand human links. *ACM Conf, keynote AVEC 2019 â State of Mind, Detecting Depression with AI, and Cross-Cultural Affect Recognition, Nice, France*, 2019.

[26] F. Moreno-Noguer J. Solà A. Sanfeliu V. Vaquero, I. del Pino and J. Andrade-Cetto. Deconvolutional networks for point-cloud vehicle detection and tracking in driving scenarios. *2017 European Conference on Mobile Robots (ECMR)*, 2017, pp. 1-7, doi: 10.1109/ECMR.2017.8098657.

[27] Víctor Vaquero, Iván del Pino, Francesc Moreno-Noguer, Joan Solà, Alberto Sanfeliu, and Juan Andrade-Cetto. Dual-branch cnns for vehicle detection and tracking on lidar data. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2020.

[28] Bihao Wang, Sergio Alberto Rodríguez Florez, and Vincent Frémont. Multiple obstacle detection and tracking using stereo vision: Application and analysis. In *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1074–1079, 2014.

[29] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. What could move? finding cars, pedestrians and bicyclists in 3d laser data. In *2012 IEEE International Conference on Robotics and Automation*, pages 4038–4044, 2012.

[30] Zhongyang Zhao, Yinglei Cheng, Xiaosong Shi, and Xianxiang Qin. Classification method of lidar point cloud based on threedimensional convolutional neural network. *Journal of Physics: Conference Series*, 1168:062013, 02 2019.