applied sciences

MDPI

Article

Semantic Mapping with Low-Density Point-Clouds for Service Robots in Indoor Environments

Carlos Medina Sánchez [1,*,†], **Matteo Zella** [1], **Jesús Capitán** [2] **and Pedro J. Marrón** [1]

[1] Networked Embedded Systems Group, University of Duisburg Essen, 45127 Essen, Germany; matteo.zella@uni-due.de (M.Z.); pjmarron@uni-due.de (P.J.M.)

[2] Robotics, Vision and Control Group, University of Seville, 41092 Seville, Spain; jcapitan@us.es

* Correspondence: carlos.medina-sanchez@uni-due.de; Tel.: +49-201-183-6362

† Current address: Schützenbahn 70, 45127 Essen, Germany.

Received: 30 July 2020; Accepted: 8 October 2020; Published: 14 October 2020

check for updates

Featured Application: This work can be applied for autonomous mobile robots working in indoor scenarios as service robots.

Abstract: The advancements in the robotic field have made it possible for service robots to increasingly become part of everyday indoor scenarios. Their ability to operate and reach defined goals depends on the perception and understanding of their surrounding environment. Detecting and positioning objects as well as people in an accurate semantic map are, therefore, essential tasks that a robot needs to carry out. In this work, we walk an alternative path to build semantic maps of indoor scenarios. Instead of relying on high-density sensory input, like the one provided by an RGB-D camera, and resource-intensive processing algorithms, like the ones based on deep learning, we investigate the use of low-density point-clouds provided by 3D LiDARs together with a set of practical segmentation methods for the detection of objects. By focusing on the physical structure of the objects of interest, it is possible to remove complex training phases and exploit sensors with lower resolution but wider Field of View (FoV). Our evaluation shows that our approach can achieve comparable (if not better) performance in object labeling and positioning with a significant decrease in processing time than established approaches based on deep learning methods. As a side-effect of using low-density point-clouds, we also better support people privacy as the lower resolution inherently prevents the use of techniques like face recognition.

Keywords: semantic mapping; pointcloud-based sensors; service robots

1. Introduction

Service robots are becoming common in our everyday life in an increasing number of scenarios. They can perform tasks autonomously and cooperate with humans to guide people in airports [1], assist the elderly [2], help in the education of children [3], aid staff and patients in hospitals [4], and be of service in work environments in general [5]. An essential skill in these environments is the ability to perceive the surroundings and the aspects of relevance for operations in social contexts. To achieve this, it is not only necessary to accurately detect people but also to identify those objects essential to enable the coexistence and the interaction between robots and humans.

In indoor environments, recognizing objects like doors, chairs, and stairs (among others), can allow robots to understand better their environment and operate more efficiently with humans. For instance, detecting whether a room is closed or not, or whether a person is sitting in a room, is useful information that robots could use to react and adjust their ongoing plans accordingly. Moreover, if robots are to navigate in dynamic environments, object detection needs to be performed online in an efficient manner, so that the robots can react promptly to the presence of people or objects of interest. Ideally, we also want to avoid the need for extensive pre-training tasks.

Nowadays, object recognition is performed mostly through the use of high-resolution visual cameras. This brings up issues related to human privacy, especially in public spaces, as service robots could perform face recognition or store/share recorded videos. However, alternative sensors like 3D LiDARs are becoming more and more common, extending the range of options for perception. Even though they are not still so widely used in indoor environments, their characteristics like Field of View (FoV) and detection distance, make them a valuable option to perceive objects efficiently and reduce privacy concerns, as confirmed by recent market trends [6,7]. Table 1 compares the main characteristics of some typical sensors for perception in service robots, which we use in our experimentation.

Table 1. Comparison of exemplary sensors used for perception by service robots.

Sensor	Scope	Vertical FoV	Horizontal FoV	Samples × Second	Points × Sample	Max. Detection Distance
			Camera			
Astra camera $640px \times 480px$ ($160px \times 120px$) [8]	3D	49.5°	60°	33	307,200 (19,200)	6–8 m
			LiDAR			
UST-10LX [9]	2D	N/A	270°	40	1440	30 m
Velodyne VLP-16 [10]	3D	30°	360°(60°)	10	30,000 (5000)	100 m
Cube 1 [11]	3D	30°	70°	10	13,125	75 m
CE30 [12]	3D	9°	132°	20	7680	4 m

As we will discuss in Section 2, there exist multiple methods for adding, on top of the geometric map, a semantic layer including detected objects of relevance for robot navigation. We take navigation of service robots as the reference task, as any other task is likely to be based on navigation. In particular, we consider navigation in a 2D plane, according to the nature of service robots operating indoors. In this context, there are 2D and 3D approaches using different types of sensors to build geometric maps. From these maps, classification algorithms can recognize specific objects and add a new layer with semantic information on top of the physical one, the so-called semantic maps. Semantic maps can then be used to refine robot behaviors, computing more efficient plans depending on their surrounding information. At the best of our knowledge, however, existing solutions for building semantic maps either rely on rich information, as provided by visual cameras and/or high-end processing resources, to apply complex classification algorithms.

In our previous work [13], we identified a set of effective techniques to process point-cloud data for the detection of people in motion. In particular, our goal was to exploit a set of specialized filters to process both high- as well as low-density depth information with constrained computing resources. This work extends our previous approach (Section 3) to address also static people as well as different objects of relevance (like doors and chairs) for navigation tasks in social contexts. In particular, we introduce (a) height, (b) angle, and (c) depth segmentation as a way to differentiate among objects depending on their physical properties. By analyzing the information along these directions, our approach offers a set of tools that can be combined together in different ways to specify the unique features of the objects to be detected. In this way, generic pre-training tasks can be replaced by specific rules able to perform detection more efficiently in the operational environment. The extensive evaluation of our approach (Section 4), in comparison with established solutions, shows that we can offer a similar or better detection accuracy and positioning precision with a reduced processing time.

As a result, the main contributions of this work are the following:

- We propose different classes of segmentation methods to compute semantic maps with low-density point-clouds;

- Our object detection methods are effective with a performance similar to those of more complex approaches, but using less processing time;
- We present experimental results comparing the performance of our approach with different semantic mapping solutions for different sensor types as well as point-cloud densities.

In conclusion (Section 5), we show that semantic mapping for indoor robot navigation can be performed effectively even without high-end processing resources or high-density point-clouds. This poses the basis for privacy preserving, low-end robotic platforms able to coexist and cooperate with humans in everyday indoor scenarios.

2. Related Work

As the advent of mobile robots, there has been interest in equipping them with better sensors and perception functionalities to understand their surrounding environment and make more intelligent decisions accordingly. In this section, we discuss the perception technologies of interest for our work and the state-of-the-art in object detection and semantic mapping.

2.1. Perception Technologies

Lasers or 2D LiDARs have been mainly the sensors used for mapping indoor environments with robots for a long time. However, this type of sensors can only perceive features on a single plane, therefore providing limited information for object detection. For this reason, and due to new advances in sensor technology and computation resources, the use of cameras to process 3D information spread lately too. Stereo cameras or RGB-D cameras can provide 3D perception by means of processing algorithms that output high-resolution point-clouds, with a restricted FoV though (Table 1). Moreover, the availability of both depth as well as color information significantly facilitates the task of classifying different objects.

The main limitation of RGB-D cameras resides in the short distance at which objects can be detected. Such restriction paved the way for the extensive use of 3D LiDARs in outdoor environments. A reference application scenario for this technology is, indeed, autonomous driving [14]. In Figure 1, we report a representative comparison of the output of an RGB-D camera against that of a 3D LiDAR. The color and depth information provided by the camera has a 10-time higher resolution within a smaller FoV in comparison to what a 3D LiDAR can provide for a 360° view.

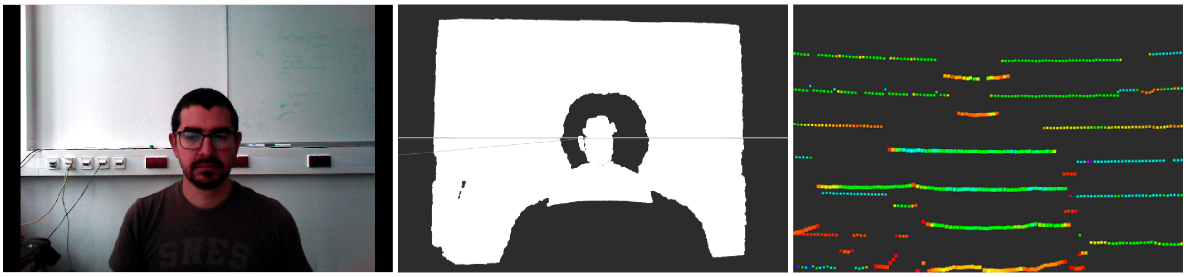

(**a**) (**b**) (**c**)

Figure 1. Illustrative example of sensed data by an RGB-D camera and a 3D LiDAR: (**a**) RGB-D camera color image; (**b**) point-cloud provided by an RGB-D camera; (**c**) point-cloud provided by a LiDAR.

The resulting low-density of 3D LiDARs and their high price has promoted the belief that this technology is inefficient for indoor scenarios. Thus, the advantage of offering an extended detection range seems to be uninteresting in small spaces like the ones typical of buildings. Interestingly enough, however, a sparse point-cloud has the side effect of offering an inherent higher privacy standard, e.g., given the inability of performing face recognition. Nonetheless, the initial high price for this technology is seeing a clear decrease (the well-known Velodyne VLP-16 made its appearance with a cost of around 8000 Euros and it is now available for 4000 Euros). This is due to both the larger adoption, currently mostly in outdoor scenarios, as well as the move to a solid-state technology, e.g., the Velarray from Velodyne [7], the Cube 1 from PPEP [11] or the CE30 from Benewake [12], which costs around 685 Euros [15]. The last Ipad Pro from 2020 [6] also integrated a solid-state LiDAR without an increase in price in comparison to previous versions.

In this work, we embrace the current trends promoting the use of 3D LiDARs and investigate the potential that this technology offers for indoor scenarios. We analyze how low-density point-clouds can be processed to detect typical indoor objects without sacrificing accuracy in comparison to established methods. In particular, we highlight the potentials and show how the properties of low-density point-clouds provided by 3D LiDARs can be exploited to simplify the processing, making the technology practical also for low-end robotic platforms.

2.2. *Object Detection and Semantic Mapping*

With a given technology to perceive their surroundings, service robots can build maps and then exploit them for navigation, e.g., in indoor environments. Mapping techniques go hand in hand with the perception technology. For these reasons, 2D LiDARs can easily build floor plans [16], which provide incomplete information but can be created with low-end hardware platforms. As can be easily imagined, however, the height of the robot cannot be taken into account, and obstacles interfering with the robot might not be detected if they are not observable at the same height of the sensor. In our previous work [13], we evaluated 2D approaches to detect people. In particular, we experimented with the Edge Leg Detector (ELD) package [17] and observed that the processing time as well as the number of false-positives were considerably higher than approaches exploiting 3D information.

With 3D cameras and LiDARs, it becomes possible to build more complex maps and improve, as a consequence, the navigation. Thus, there are recent examples of methods for geometric and semantic 3D mapping with LiDARs [18–20]. Of course, 3D sensor information can also be exploited to build 2D traversable maps, which are easier to compute and store, still considering important aspects like the height of the robot. For example, MONO SLAM [21] uses RGB-D cameras for this type of 2.5D maps, and our prior work, PFF [22], exploits low-end point-clouds obtained by RGB-D cameras and 3D LiDARs for the same purpose. In general, a large variety of approaches based on SLAM have been proposed to build geometric maps in 2D and 3D environments. Although they are helpful for robot navigation, in this work we are interested in gathering semantic information that the robot could exploit while performing its service tasks, in particular in the case of dynamic objects.

Many of the aforementioned approaches consider a stationary environment, where all the objects remain at their position and no moving element is recognized. In a realistic scenario, objects like chairs are moved, doors are opened or closed, and people move freely around. As a result, the computed maps become quickly obsolete and the computational effort must be continuously repeated. To address this challenge, it becomes essential to detect and classify efficiently the objects of relevance for navigation, producing then a richer, semantic map. For the detection, it is first necessary to specify the objects of interest. For example, the Astra Body Tracker (ABT) package [23] exploits 3D information from RGB-D cameras and models of the human body to detect relevant points of the human skeleton such as the head, knees, shoulders. However, deep learning techniques based on *Convolutional Neural Networks* (CNN) are becoming commonplace, as they can be used to learn how to recognize different classes of objects from high-resolution data like images. In this case, it is possible to avoid using models by exploiting specific training data sets for each of the objects that need to be detected. For instance, the method in [24] matches frames provided by a camera against a 3D map of the environment, enabling the labeling of objects in the environment and their correct placement. In outdoor environments, CNNs have also been used to detect driving lanes [25]. Similarly, other deep learning approaches have been presented for object-oriented semantic mapping [26] or geometric-based segmentation [27], and self-supervised training [28] can also be exploited. The resulting maps can reach an accuracy higher than 90% in certain cases.

Approaches explicitly designed to handle observations from 3D LiDARs are of particular relevance for our work. Still, within the context of solutions exploiting CNNs, Bayesian Filters [29] have been used to distinguish between static, mobile, and dynamic objects. Siamese neural networks [30] and also methods without explicit use of CNNs [31] have been proposed to detect various types of features from the environment. Related to our approach, the authors in [32] perform cluster detection based on a cost function with different features, to learn online how to detect humans. These works seek to reduce the error in detection by learning from information sensed previously. The observations are compared against the stored models at a cost of an increase in the computation time. Additionally, these contributions focus only on the detection of single elements, e.g., people in indoor environments, without building a complete semantic map.

However, methods based on deep learning require high-end, powerful hardware, often with a dedicated GPU, to perform the computation in real-time. To increase the computation effort, alternative approaches can be used paying attention to the physical characteristics of the objects to be detected. Thus, methods identifying legs [33], faces [34], or human poses [35] have been proposed. Their reduced computational effort makes these approaches suitable for semantic mapping on more common robotic platforms at the cost of a decrease in accuracy. In this work, we show that lightweight methods can still provide good accuracy even with low-density information.

3. Semantic Mapping via Segmentation

We now introduce our approach to solve the problem of creating semantic maps of relevance for the navigation of service ground robots in indoor environments. While RGB-D cameras allow one to exploit color images together with depth information to classify objects, we limit ourselves to depth information only and challenge ourselves further by looking at point-clouds with low-density. Specifically, we are interested in studying how sensors with low resolution, e.g., 3D LiDARs, can be used to identify different objects in an indoor scenario. As reported in Table 1, state-of-the-art 3D LiDARs have a significantly lower density in comparison with RGB-D cameras within a similar FoV. However, by looking at how the point-cloud is perceived, we offer a way to recognize the physical structure of different objects. Therefore, we introduce several segmentation methods along which point-clouds can be processed and unique, distinctive features can be extracted. Through this approach, we avoid complex processing tasks performed in classical deep learning approaches, which still should be trained for the specific objects to be detected. In addition, by applying these techniques to low-density point-clouds, our solution can offer higher privacy as personal and private details are not visible to the sensor. Finally, by operating on point-clouds, our methods can be applied to 3D LiDARs as well as RGB-D cameras, or any other sensor providing the same type of measurement.

As depicted in Figure 2, we propose different types of segmentation methods to detect different objects, focusing on their physical characteristics. Then, several of these methods could be executed in parallel with different configurations, enlarge the classes of objects to detect. For clarity of presentation, we introduce each segmentation method with its application to the detection of one specific object of interest in our indoor scenario. In particular, we are interested in detecting humans as well as objects like chairs and doors that could affect the ability of the robot to perform its service tasks, e.g., by navigating socially respecting proxemics rules or using the semantic information to search for humans or interact with them. In general, however, the same object can be detected by using different segmentation methods (alone or also in combination) if properly configured.

In the remaining discussion, we assume that the information about the robot position is available together with a traversable map of the environment, like the one we computed in our previous work [22]. As mentioned before, the segmentation is performed on low-density point-clouds provided, e.g., by a 3D LiDAR. The output of our method is then a semantic map composed of the positions within the geometric map of all the objects detected, together with their labeling.

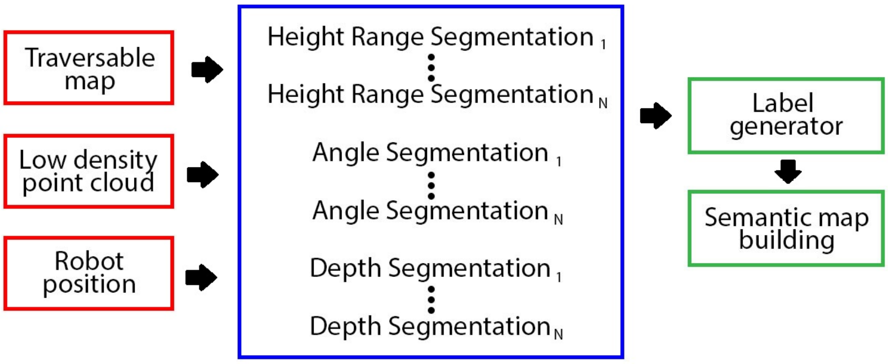


Figure 2. The architecture of our proposed approach for semantic mapping, based on multiple segmentation methods applied to detect different types of objects.

3.1. Height Segmentation

The first segmentation approach that we investigate focuses on the detection of objects that have peculiar characteristics at different heights. Low-density point-clouds typically provide information in layers, as depicted in Figure 3. For mechanical 3D LiDARs, the layers correspond to the array of rotating lasers, each tilted with a different angle. For this reason, the obstacles that are at a further distance from the sensor have bigger gaps between layers. Figure 3 shows how the number and position of useful points to detect a person drastically change with distance.

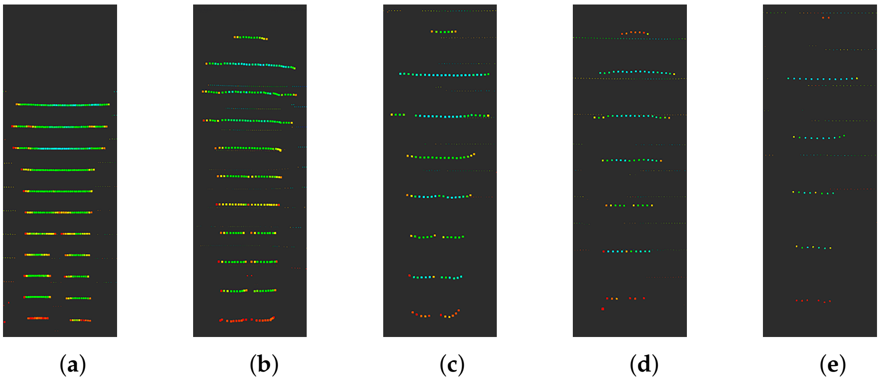


Figure 3. Sensed human body at different distances from a 3D LiDAR (Velodyne VLP-16) located at a height of 40 cm above the floor. The person is positioned (**a**) 2 m, (**b**) 4 m, (**c**) 6 m, (**d**) 8 m, and (**e**) 12 m from the sensor.

To implement such detection, it is necessary to define the various height ranges where the object of interest has distinctive features. If one or more perceived layers fall in such ranges and the measurements correspond to the physical structure of the target object, a match is found. This type of detection matches well with the perception of the human body, for instance. As represented in Figure 4, it is possible to identify two sectors of relevance: a height range with legs and one with the trunk of the human body. More specifically, denoting the height as h, we distinguish between a *Lower Section* (LS), defined as $S_1 < h < S_1 + LS_h$, and a *Higher Section* (HS), as $S_1 + LS_h + S_2 < h < S_1 + LS_h + S_2 + HS_h$. Nonetheless, our method is more general, and we could define more sectors with different height ranges to detect other objects with different physical characteristics.

This type of segmentation is a variation of the one proposed in our previous work [13] for the detection of moving people. The complete procedure is exemplified in Figure 5. Once the height ranges have been defined, all the points belonging to the different ranges are extracted and stored in different *Height Range Clouds* (HRC), and the remaining points are discarded. To obtain these clouds it is also necessary to define how many points should each have, according to the horizontal angle resolution (H_r) of the sensor. Taking into account that multiple layers could fall in the same range, among vertically aligned points, the nearest one is selected. This is done for the complete FoV at the given H_r resolution. At the end of this step, one layer is constructed for each HRC with FoV/H_r points, as shown in Figure 5c.

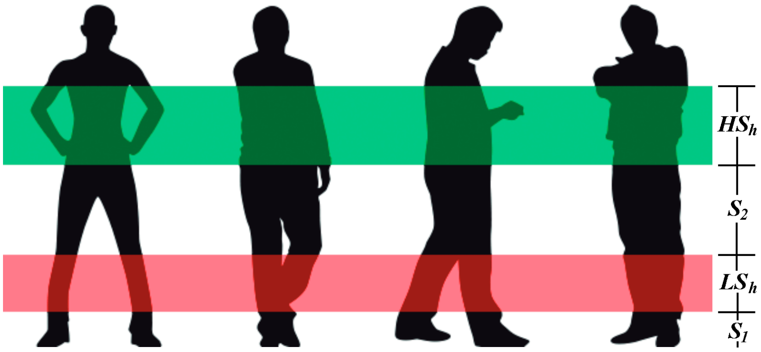


Figure 4. Example of height range segmentation for people detection. In red, the Lower Section; in green, the Higher Section.

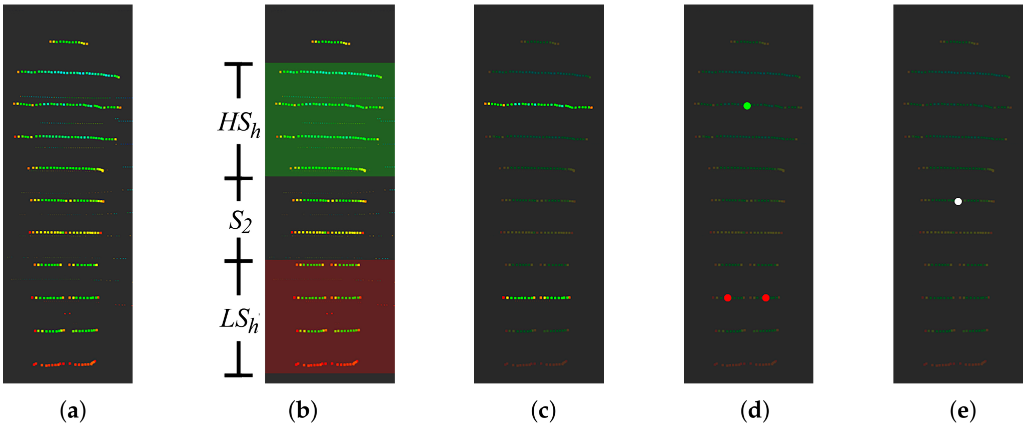


Figure 5. Height range segmentation for human detection: (**a**) sensed point-cloud; (**b**) definition of height ranges, points in green and red belong to two different HRC; (**c**) extraction of one layer per HRC; (**d**) extraction of cluster centroids for each layer; (**e**) matching between the points in the two sections to define the center of mass of the human body.

Within the layer of each HRC, clusters of points with a maximal Euclidean distance D_{max} between them are identified. Points in the point-cloud are visited in increasing angle, and the Euclidean distance (on the XY plane) between two consecutive points is measured. If greater than D_{max}, a new cluster is created and the previous one is completed. Once this process is finished, the width of each cluster is compared to a set of reference widths for the parts of the object to be detected according to the different sections. In this particular case, a lower section minimum width (LS_{min}), a lower section maximum width (LS_{max}), a higher section minimum width (HS_{min}), and a higher section maximum width (HS_{max}) are defined. If the width of a cluster is outside these ranges, all its points are deleted from the HRC. For the remaining clusters, only the information about its centroid is preserved, as shown in Figure 5d.

Finally, the centroids from clusters in the two different HRC are compared with each other, looking for a match. In the case of people detection, only the single points (i.e., a trunk) in HS with a corresponding pair of points (i.e., legs) in LS are preserved. These points are then labeled as the possible position of a detected human person. However, service robots in indoor environments are exposed to many different objects some of which could be falsely classified according to the presented scheme. To increase accuracy, we make use of traversable maps of the scenario to filter out objects placed in unlikely positions.

3.2. Angle Segmentation

We now introduce a second and alternative approach for object detection that is based not anymore on the absolute height of the measurements but rather on their relative position with respect to the robot. In particular, we look at the vertical angle of the observed points, searching for layers at certain reference angles. An example of the application of the method is shown in Figure 6. In this example, 3 reference angles are used for the detection of a chair: a low angle (θ_l) to detect the legs of the chair; a medium angle (θ_m) where the back of a chair can be found; a high angle (θ_h) to determine the height of the object. This last angle marks the biggest difference with respect to the previous segmentation method. In fact, by estimating the height of the objects, it is possible to discard those objects that may have similar structures to the desired one and produce false positives. The definition of these angles includes a vertical tolerance (V_t), which allows us to consider points within a range of angles. In the case of the VLP-16, which has a vertical resolution of 2°, the V_t value might permit only one layer. However, for an Astra camera, more layers might be accepted since its vertical resolution is approximately 0.1°. If multiple layers are selected, it is then necessary to extract the nearest point for each horizontal angle to build individual layers as presented in Section 3.1. The point layers at the two lower reference angles are then processed in terms of clustering and matching, as explained in Section 3.1. To make this approach possible, a reference distance (D) needs to be defined between the sensor and the object to be detected. This distance will determine the adequate values of the reference angles that are representative of the object of interest. For this angle segmentation, we only process objects observed at this reference distance (within a tolerance interval) and discard others, as measurements closer or further may present different physical proportions to those in our configuration. Nonetheless, our method could consider a dynamic reference distance D and adapt the reference angles according to that.

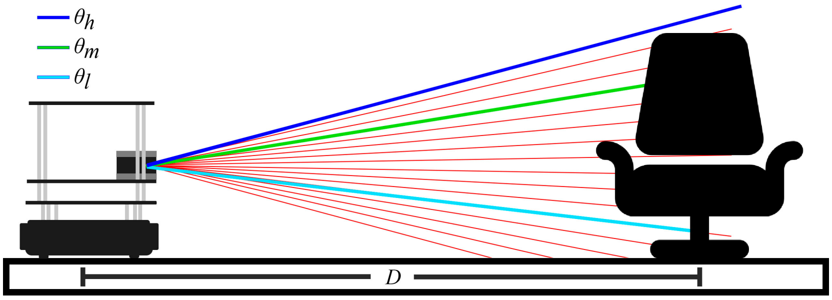
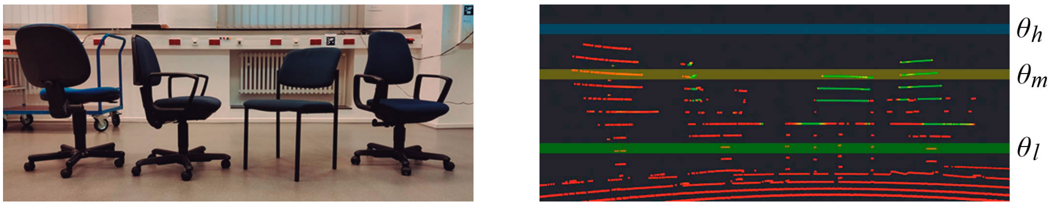


Figure 6. An illustrative example of the segmentation based on the angle for a chair sensed by a 3D LiDAR. Different colors highlight the three reference angles used for detection.

Differently from the procedure discussed in Section 3.1, there is no need to extract the nearest points for each reference angle, considering that only one layer is observed (in the height segmentation multiple layers had to be merged together at each section). However, it is still necessary to define a lower angle minimum width ($MW_{\theta l}$), a lower angle maximum width ($HW_{\theta l}$), a medium angle minimum width ($MW_{\theta m}$), and a medium angle maximum width ($HW_{\theta m}$). They represent the minimum and maximum allowed width for the object clusters to be detected within the layers at θ_l and θ_m, respectively. Thus, the clusters extracted from the two lowest reference angles are filtered out according to these parameters, and their centroids are computed.

In this angle segmentation, the highest reference angle (θ_h) is used to check the height of the object. Depending on the type of object to detect and the value of the reference distance D, θ_h is selected so that it falls within the FoV of the sensor and above the typical height of the objects of that class. In conclusion, the final object detection is confirmed by matching centroids from clusters in the two lowest reference angles (θ_l and θ_m), and checking that the object height (i.e., the top layer detected by the sensor) is below θ_h. Figure 7 depicts an example of sensing different models of chairs and the different segmentation angles.

In the case of chairs, applying angle segmentation for detection can also be exploited for detecting sitting people. In particular, after having detected a chair and estimated its position, it is possible to modify the algorithm to detect changes in the height of an already detected chair by looking at the highest layer detected by the sensor. An increase in height could be associated with the presence of a person sitting on the chair.



(**a**) (**b**)

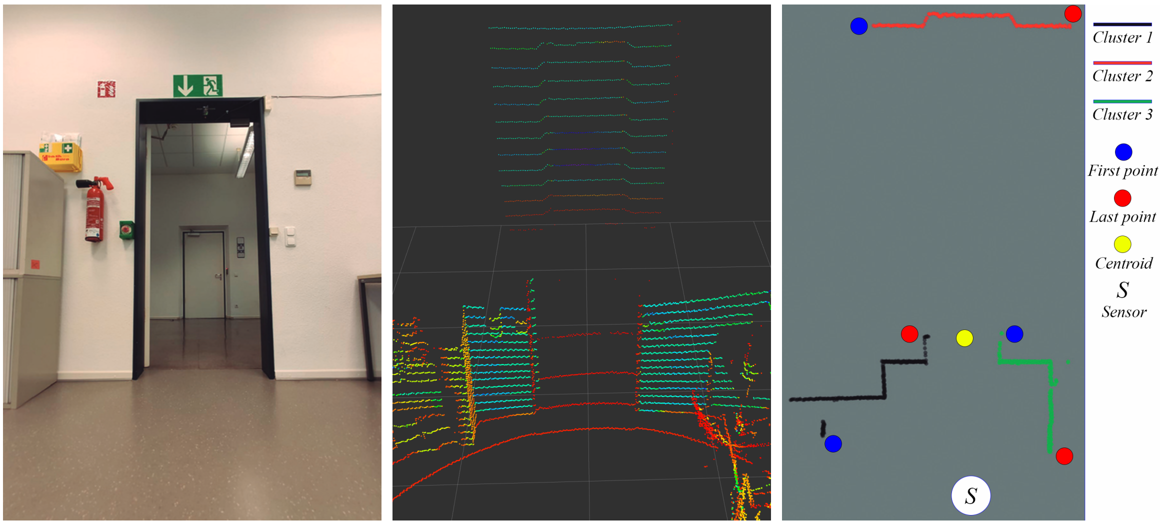
Figure 7. Angle segmentation: (**a**) real scenario with different models of chairs; (**b**) point-cloud with θ_l, θ_m and θ_h highlighted.

3.3. Depth Segmentation

The methods discussed until now focus on features that change along the vertical dimension of the object. For this reason, the key was to distinguish among different vertical layers, either in specific ranges (for height segmentation) or individually (for angle segmentation). We now look into objects that maintain the same structure throughout their vertical extension but that manifests a characteristic depth profile. In particular, we are interested in identifying objects like doors or cabinets.

For the sake of simplicity, we focus here on the detection of doors, but the method may apply to other objects with similar characteristics. In this case, it is possible to identify the reference top and bottom heights of a door and points above it, or below (i.e., reflected on the floor), can be discarded. All the layers in between can then be merged, throwing away the height information and preserving only the nearest points for each horizontal angle step, as done in the height segmentation. Again, clusters of consecutive points with a distance smaller than D_{max} are computed. We visit all points ordered by a horizontal angle and calculate the Euclidean distance (on the XY plane) with the next point. If this distance is greater than D_{max}, the cluster is closed and a new one is opened. In the case of a door, for instance, the value of D_{max} needs to be chosen so that it is smaller than the width of the door frame. Otherwise, all significant points would be fused into the same cluster, making the definition of the object more complex.

An example of this segmentation method for the detection of doors is presented in Figure 8. First, the point-cloud is sectioned in clusters and the distance between each pair of clusters is computed. This distance is the Euclidean distance on an XY plane, from the last point of one cluster (red circles in Figure 8c) to the first point of the next cluster (blue circles in Figure 8c). If those distances are within the tolerance interval defined by a minimum object width (OW_{min}) and maximum object width (OW_{max}), there is a match, and the middle point between the corresponding clusters is computed (yellow circle in Figure 8c) and labeled as the position of the detected object.



(**a**) (**b**) (**c**)

Figure 8. Depth segmentation: (**a**) real scenario; (**b**) sensed point-cloud; (**c**) top view of the point-cloud after merging heights, the three resulting clusters of the segmentation are shown.

4. Results and Discussion

We now evaluate the performance of our approach for semantic mapping. First, we analyze the processing time from the delivery of the sensor readings to the estimation of the object's positions. Furthermore, we assess the detection accuracy in terms of correct and false identifications. Finally, we quantify the precision of the position estimation in comparison to the real placement of the object in the environment.

We implemented the segmentation methods for semantic mapping introduced in Section 3 in C++ and integrated them into ROS Kinetic Kame. We tested the algorithm on a machine with an i7@2.2 GHz processor and 16 GB of RAM to which we connected a Velodyne VLP-16 3D LiDAR. We focus the experimentation on the VLP-16, but our method would work with any sensor providing point-clouds, like a solid-state LiDAR (e.g., the CE30 with the SDK for ROS [36]) or an RGB-D camera (e.g., the Astra camera through the package ros_astra_camera [37]). Actually, by using the VLP-16 we put ourselves in the condition of having a significantly lower point density than what should be expected from other technologies (see Table 1). Moreover, we avoid privacy issues related to visual cameras. The sensor was located 40 cm above the floor on a Turtlebot2. On top of the LiDAR, we also placed an RGB-D Astra camera that we used to compare against alternative solutions. Figure 9 shows the hardware platform used for the experiments.

The experiments were performed in the indoor environment depicted in Figure 10 with an approximate area of 450 square meters. The scenario is divided into multiple rooms and corridors where heterogeneous objects typical of office environments were placed. The movements of the robot were controlled remotely via teleoperation, as we were mainly interested in the mapping part. In each test, the robot was driven through all the rooms, which took between 20 and 25 minutes to complete. The segmentation methods previously discussed were configured to detect people (height segmentation), chairs (angle segmentation), and doors (depth segmentation). The parameters used in the experimentation are presented in Table 2. For these experiments, we used a traversable map of the environment that accounts for the height of the robot, as built by existing libraries [22].

We compare our technique against other alternative approaches. For people detection, we employed the ELD algorithm [17], which performs the detection of legs through readings obtained by a 2D LiDAR. We also tested the ABT package [23], which can detect human skeletons with an RGB-D camera, i.e., the Astra camera we installed on our robot. Last, we compare also against our previous approach, PFF_PeD [13]. Considering that the Velodyne driver for ROS can provide both 2D and 3D measurements, we used the same sensor to test the segmentation methods presented in this work as well as the ELD algorithm. As one of our goals is to preserve the privacy of people moving in the environment, we refrained from comparing our solution against approaches like face recognition.

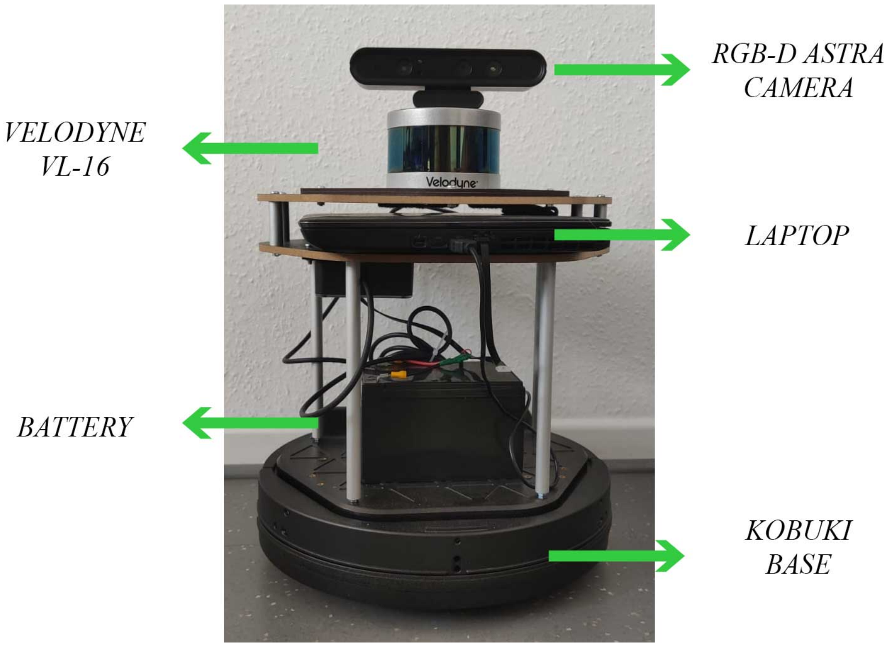


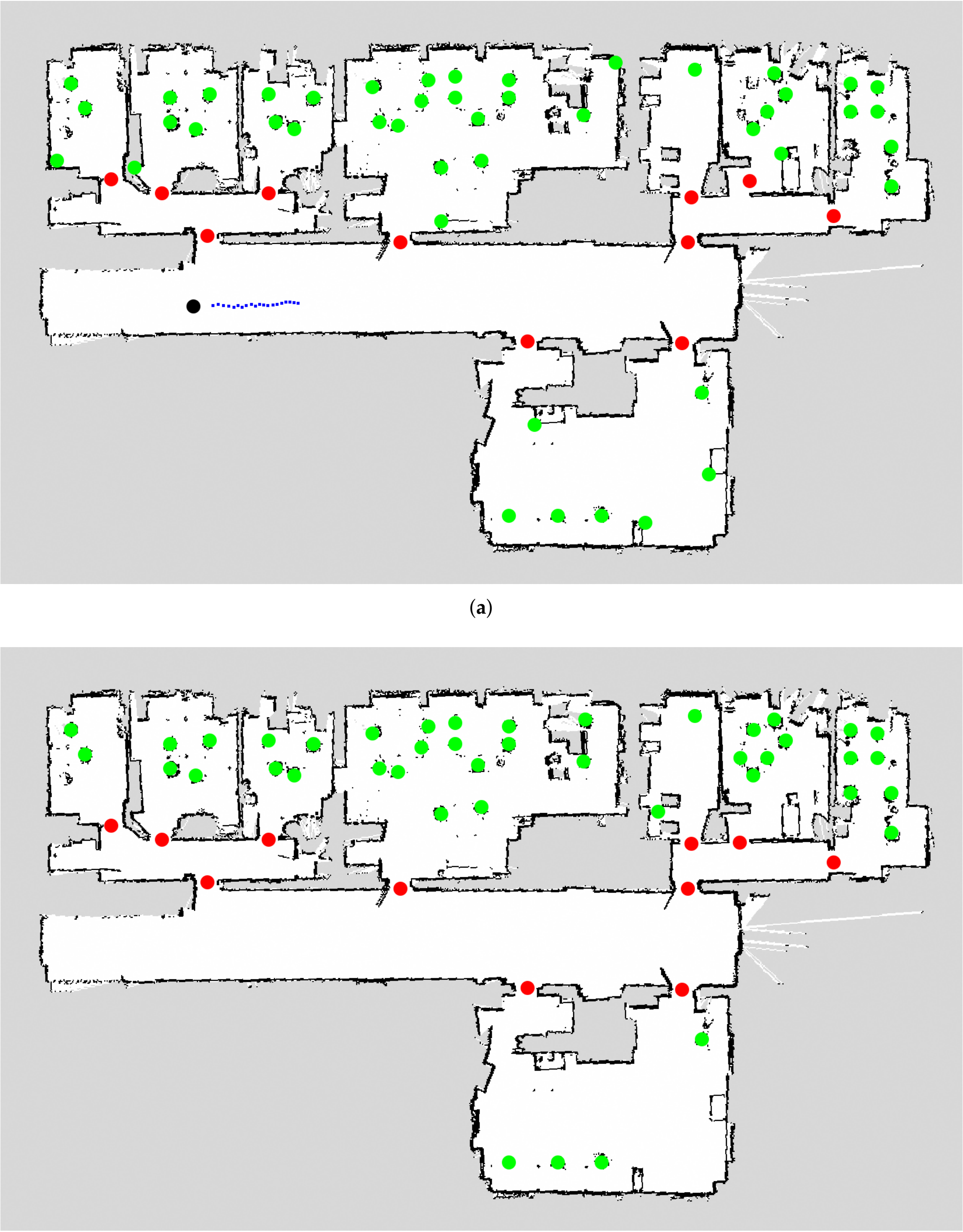
Figure 9. Reference hardware platform based on Turtlebot2 and used for the evaluation of the proposed methods.

For object detection, instead, we compare our approach against the Find Object (FO) package [38], which exploits information from existing datasets as well as images from the scenario. Furthermore, we tested the semantic_slam package (SS) [39], which uses CNN and an RGB-D camera to perform the detection based on color and depth information. This approach, together with the ORB-SLAM2 and Octomap packages, can build a full 3D semantic map. In our evaluation, we discarded solutions like Semantic Fusion [24] that explicitly require a powerful GPU to perform the computation.

At first, we trained the FO package with the dataset provided by Mathieu Aubry [40], which contains a large number of models of chairs, but without success in the detection. Therefore, we trained this approach locally, adding images of the scene for each object from different points of view. In total, we provided 50 images for each design of chair present in the environment, 50 images for the detection of people, and 20 for the detection of doors. Instead, the SS solution was trained with the ADE20K dataset [41].

Table 2. Parameters used in the experimentation.

Height Segmentation									
H_r	S_1	LS_h	S_2	HS_h	LS_{min}	LS_{max}	HS_{min}	HS_{max}	D_{max}
0.1°	25 cm	50 cm	10 cm	65 cm	5 cm	25 cm	15 cm	65 cm	15 cm
Angle Segmentation									
	D	$MW_{\theta l}$	$HW_{\theta l}$	$MW_{\theta m}$	$HW_{\theta m}$	θ_l	θ_m	θ_h	V_t
	2.3 m	0.5 cm	15 cm	5 cm	55 cm	−5°	9°	15°	±0.5°
Depth Segmentation									
			H_r	D_{max}	OW_{min}	OW_{max}			
			0.1°	50 cm	92 cm	98 cm			



(**a**)

(**b**)

Figure 10. Semantic map of an indoor office environment. (**a**) Estimated position of the objects with our method; (**b**) real position of the objects in the scenario. The semantic information included are doors (red points), chairs (green points), and people (blue squares). The black dot was the actual starting position of the moving person.

4.1. Processing Time

To quantify the performance of our solution, we analyze first the processing speed of the different segmentation methods. We define processing time as the time that the algorithm takes since it receives a new point-cloud from the sensor until it is processed resulting in the detector output. The results presented in this section are the average time required by the different algorithms to process 1000 samples (point-clouds) provided by the sensor in use. The results for the processing time are presented in Table 3.

Table 3. Average time employed by the different tested algorithms for processing the sensor readings and detecting objects.

Object	Method	Sensor	Samples	Horizontal FoV	Processing Time	Standard Deviation
Any	Ours	VLP-16	1000	360°	21.22 ms	3.28 ms
People	PFF_PeD [13]	VLP-16	1000	360°	27.97 ms	3.53 ms
People	ELD [17]	VLP-16	1000	360°	89.82 ms	0.5 ms
Any	FO [38]	Astra Camera	1000	60°	29.65 ms	3.72 ms
Any	SS [39]	Astra Camera	1000	60°	325.75 ms	40.5 ms

Considering that the algorithms based on CNN can distinguish between multiple classes of objects simultaneously, we also ran our solution according to the scheme presented in Figure 2, i.e., we executed the different segmentation methods in parallel to detect multiple object types simultaneously. In this case, our algorithm takes an average of 21.22 ms to process a point-cloud for object detection. This approach is simpler than our previous PFF_PeD work, which required two further steps to detect moving persons. As a result, the processing time decreases by approximately 6 ms. Apart from this increase in detection speed, we are now able to identify both moving and stationary persons. In contrast, the pure 2D solutions detecting legs (ELD) is 4 times slower than our approach.

The Find Object (FO) solution takes an average of 29.65 ms to detect any of the objects we are interested in identifying. In this case, the processing time depends on the number of images that the algorithm uses for feature extraction (FO uses a database with multiple images for comparison to process each sensor sample). In our experiments, we used 170 images for processing each sensor sample; if the number of images increases, this will have a negative impact on the processing time. For SS, instead, the dataset employed provides a large number of labels to distinguish a larger set of objects. However, this type of solution based on CNN has a high computation cost and are usually executed on machines with high-performance GPUs. For our scenario and our reference hardware platform, this resulted in 325.75 ms.

In conclusion, the results show that our proposed segmentation methods can significantly decrease the processing time, without imposing any strict hardware requirement. Moreover, the comparable approaches, which can distinguish among different objects, are based on the use of cameras, which offer a limited FoV (60°for the Astra camera in comparison to 360°for the Velodyne 3D LiDAR). They also require high-density point-clouds, which our reference 3D LiDAR cannot provide. Nonetheless, the lower density of the point-clouds exploited in our approach does not affect significantly the detection accuracy, as we discuss next.

4.2. Detection Accuracy

We also evaluated the detection accuracy of our method with the same experimental setup. For that, we propose as metric the *True Positive Rate*, which is calculated dividing the true positive detections (objects correctly detected) by the number of samples. Table 4 shows the results obtained with several approaches, using a number of 1000 samples (frames or point-clouds) for each case. The results are separated depending on the type of object to detect. It is also important to highlight that, in the case of approaches based on cameras, unlike it happens with LiDARs, illumination conditions may affect significantly the results. Therefore, we selected favorable illumination conditions to run the experiments and compare them fairly.

Table 4. Accuracy in object detection. Depending on the type of object we compare different approaches.

Object	Method	Sensor	True Positive Rate
Chairs	Angle Segmentation (ours)	VLP-16	67.3%
Chairs	FO [38]	Astra Camera	63.7%
Chairs	SS [39]	Astra Camera	73.2%
Doors	Depth Segmentation (ours)	VLP-16	85.4%
Doors	FO [38]	Astra Camera	<20%
Doors	SS [39]	Astra Camera	37.3%
People	Height Segmentation (ours)	VLP-16	96.95%
People	SS [39]	Astra Camera	97.51%
People	PFF_PeD [13]	VLP-16	90.1%
People	PFF_PeD [13]	Astra Camera	95.8%
People	ABT [23]	Astra Camera	95%
People	ELD [17]	VLP-16	63.78%

First, we analyze the detection of chairs. These are objects hard to detect, as they can present multiple shapes, sizes, and colors, depending on the model. Our Angle Segmentation method achieves a fairly good detection accuracy, only 6% below the best method, which was SS. However, as already discussed, worse lighting conditions can jeopardize SS results. Indeed, repeating our experiments with sunlight through the windows, detection accuracy for the SS method was reduced up to a 50%. In the case of the FO package, we had to add 50 images of each chair model in the environment, taken from different angles and distances. We tried first using only 10 images, but the accuracy was below 40%.

In the case of doors, there are fewer works able to detect them, so we compared our Depth Segmentation method against the packages FO and SS. Our method outperformed clearly the alternatives, as the others found difficulties to detect closed doors, as in this case, relevant characteristics are harder to be distinguished, mistaking it with the wall, which had the same color too. Even with the doors open, another wall of the same color could be seen through the door, which confused the visual-based approaches.

Finally, regarding the detection of people, our Height Segmentation method achieved a very high detection accuracy (96.95%), quite close to the best method, which was SS using CNNs (97.5%). We improved results from our previous work (PFF_PeD) when using both the LiDAR and the Astra camera. The ABT based on skeleton detection presented results slightly worse than our method, while the leg detection (ELD) performed considerably worse, presenting also a high amount of false-positives.

4.3. Positioning Precision

In this section, we show results to assess the accuracy of our method in terms of position estimation of the detected objects in the map, in relation to their real positions in the scenario. Table 5 depicts the results of our different methods with FO and SS as alternatives. The precision error is computed comparing the estimated position with the actual geometrical center of each object, that we measured by hand as ground truth. In the case of the chairs. two different models were considered with a different number of legs. Chairs of type 1 are like the first, second, and last chairs from left to right in Figure 7a, whereas chairs of type 2 were like the remaining model with 4 legs. For the chairs of type 1, our Angle Segmentation method got an error of less than 5 cm, as the estimated position depends on the width of the leg of the chair. For the chairs of type 2, our solution had an error of almost 16 cm, as the estimation in this model depends on the number of legs that are detected to estimate the position of the geometric center. For instance, if only one leg is detected the error is greater than when detecting all 4.

As for the results obtained with the FO package, two aspects can be observed. For chairs of type 1 the error is larger than with our method. FO makes detections using the color image and once the detection is made, the 3D position is estimated from the depth information of the camera. Part of the error comes from using pixels of the back of the chair to estimate the depth, instead of the center. For chairs of type 2, the error is considerably higher. In both cases, it is important to note that FO makes use of CNNs that are trained with rectangular images to detect objects on the images, and then, a 3D position is extracted from the depth information of that noisy detection. Using 3D information from point-clouds directly to train the CNNs would improve results. Moreover, in the case of SS, this algorithm created a 3D map with the information sensed using OctoMap with a resolution of 5 cm for each voxel. In part due to this, errors were in that order, 6.35 cm for chairs of type 1 and 8.36 cm for chairs of type 2.

For the door detection, our Depth Segmentation method reports an error significantly lower than the others. In part, this is because, on many occasions, FO and SS are estimated as the position of the door a point in the left or right side of the door frame, instead of the center. Instead, using LiDAR information, the middle point can be estimated with higher precision. Regarding people pose estimation, our proposal for Height Segmentation achieved the best performance. It is important to note that this test was carried out with static people for comparison. Our solution presents a better estimation of the position than methods that use cameras. Besides, these methods based on cameras had a limited range of up to 5 m to detect people, while our method could detect people up to 12 m thanks to the VLP-16 LiDAR.

Table 5. Average error distance between the actual position of the detected objects and their estimated positions in the map. People were static and 20 repetitions were run for each case.

Object	Method	Error (cm)	Deviation (cm)
Chair (type 1)	Angle Segmentation (ours)	4.53	1.37
Chair (type 1)	FO [38]	11.3	5.43
Chair (type 1)	SS [39]	6.35	2.15
Chair (type 2)	Angle Segmentation (ours)	15.75	8.73
Chair (type 2)	FO [38]	81.76	70.85
Chair (type 2)	SS [39]	8.36	4.49
Door	Depth Segmentation (ours)	8.87	3.27
Door	FO [38]	43.4	5.18
Door	SS [39]	47.35	8.92
People	Height Segmentation (ours)	8.38	5.34
People	FO [38]	19	11.28
People	SS [39]	9.42	4.76

Figure 10 presents the final results for our semantic map method in an indoor office scenario. As already explained, we teleoperated our service robot as the semantic map was built. In particular, a map considering doors, chairs, and people detections was created, though our method could also process other types of objects of help for service robots, only tuning any of the segmentation methods accordingly. For instance, in this experiment, the information about chairs can help the robot to understand whether there is a person sitting in the room or not, for whom the robot may be looking to deliver a package. Information about doors is interesting when the robot was following someone and lost track of him/her. It could predict possible paths from his/her last known position considering the doors around the office.

Regarding the mapping results, 11 doors with a width of 95 cm were detected in our experiment, although one of them was detected in the wrong position. For the chairs, in the tested environment, there were 44 chairs (36 of type 1 and 8 of type 2). 47 chairs were detected, including 8 false-positive detections and 5 false-negatives. Moreover, a moving person was included in the scenario to check the detection of moving people. The detected trajectory when the robot passes nearby is shown in Figure 10.

5. Conclusions and Future Work

In this work, we presented a set of methods for the classification of objects in indoor scenarios. Our solution extracts relevant information to identify the distinctive features of objects based on their physical structure. We consider different types of objects of interest for service robot that moves in a 2D plane, also accounting for people around. Focusing on physical properties for segmentation, we can avoid the use of complex training phases as well as resource-hungry online deep learning algorithms and exploit low-density point-cloud information as provided by 3D LiDARs. In this way, we embrace a perception technology that is becoming increasingly popular, showing its benefits also for indoor scenarios. The high detection and positioning accuracy as well as the limited requirements in terms of processing resources show, indeed, that our approach is practical and can support semantic mapping efficiently also on low-end service robots.

We are currently planning to exploit the presented results to enable the effective navigation and operation of service robots in indoor environments populated by persons, supporting the natural co-existence of robots and humans while preserving their privacy. We would like to explore the possibilities to improve the accuracy of our detection methods by integrating them with solutions based on 3D SLAM.

Author Contributions: Conceptualization, C.M.S., M.Z., and J.C.; software and validation, C.M.; writing and editing, C.M.S., M.Z. and J.C.; supervision, M.Z., J.C., and P.J.M.; funding acquisition, P.J.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We acknowledge support by the Open Access Publication Fund of the University of Duisburg-Essen.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ABT	Astra Body Tracker
ELD	Edge Leg Detector
CNN	Convolutional Neural Networks
FO	Find Object package
FoV	Field of View
GPU	Graphic Processing Unit
HRC	Height Range Cloud
PFF	Point-cloud Fast Filter
PFF_PeD	Point-cloud Fast Filter for People Detection
SLAM	Simultaneous Localization And Mapping
SS	Semantic SLAM package

References

1. Triebel, R.; Arras, K.; Alami, R.; Beyer, L.; Breuers, S.; Chatila, R.; Chetouani, M.; Cremers, D.; Evers, V.; Fiore, M.; et al. Spencer: A socially aware service robot for passenger guidance and help in busy airports. In *Field and Service Robotics*; Springer: Cham, Switzerland, 2016; pp. 607–622.
2. Perez-Higueras, N.; Ramon-Vigo, R.; Perez-Hurtado, I.; Capitan, J.; Caballero, F.; Merino, L. A social navigation system in telepresence robots for elderly. In Proceedings of the Workshop on the International Conference on Social Robotics, Kansas City, MO, USA, 1–3 November 2016.
3. Belpaeme, T.; Kennedy, J.; Ramachandran, A.; Scassellati, B.; Tanaka, F. Social robots for education: A review. *Sci. Robot.* **2018**, *3*. [CrossRef]
4. Messias, J.; Ventura, R.; Lima, P.; Sequeira, J.; Alvito, P.; Marques, C.; Carriço, P. A robotic platform for edutainment activities in a pediatric hospital. In Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Espinho, Portugal, 14–15 May 2014; pp. 193–198. [CrossRef]
5. Biswas, J.; Veloso, M.M. Localization and navigation of the cobots over long-term deployments. *Int. J. Robot. Res.* **2013**, *32*, 1679–1694. [CrossRef]
6. Apple. iPad Pro 2020 Datasheet. Available online: https://www.apple.com/ipad-pro/ (accessed on 23 April 2020).
7. Velodyne Lidar. Solid State LiDAR Velarray Datasheet. Available online: https://velodynelidar.com/products/velarray/ (accessed on 23 April 2020).

8. ORBBEC. Astra Camera. Available online: https://orbbec3d.com/product-astra-pro/ (accessed on 23 April 2020).
9. HOKUYO. Hokuyo UST-10LX. Available online: https://www.hokuyo-usa.com/application/files/2414/7196/1936/UST-10LX_Specifications.pdf (accessed on 23 April 2020).
10. Velodyne Lidar. Velodyne VLP-16. Available online: https://velodynelidar.com/products/puck (accessed on 23 April 2020).
11. Blickfeld. Solid State LiDAR Cube 1 Datasheet. Available online: https://www.blickfeld.com/products/ (accessed on 23 April 2020).
12. Benewake. CE30. Available online: http://en.benewake.com/product/detail/5c34571eadd0b639f4340ce5.html (accessed on 23 September 2020).
13. Sánchez, C.M.; Capitan, J.; Zella, M.; Marron, P.J. Point-Cloud Fast Filter for People Detection with Indoor Service Robots. In Proceedings of the 2020 Fourth IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, 9–11 November 2020; To be published.
14. Arnold, E.; Al-Jarrah, O.Y.; Dianati, M.; Fallah, S.; Oxtoby, D.; Mouzakitis, A. A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3782–3795. [CrossRef]
15. ROS Components. CE30 Price Information. Available online: https://www.roscomponents.com/es/lidar-escaner-laser/232-ce30-a.html (accessed on 23 September 2020).
16. Fuentes-Pacheco, J.; Ruiz-Ascencio, J.; Rendón-Mancha, J.M. Visual simultaneous localization and mapping: A survey. *Artif. Intell. Rev.* **2015**, *43*, 55–81. [CrossRef]
17. Bellotto, N.; Hu, H. Multisensor-Based Human Detection and Tracking for Mobile Service Robots. *IEEE Trans. Syst. Man Cybern.* **2009**, *39*, 167–181. [CrossRef] [PubMed]
18. Sun, L.; Yan, Z.; Zaganidis, A.; Zhao, C.; Duckett, T. Recurrent-OctoMap: Learning state-based map refinement for long-term semantic mapping with 3-D-Lidar data. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3749–3756. [CrossRef]
19. Chen, X.; Milioto, A.; Palazzolo, E.; Giguere, P.; Behley, J.; Stachniss, C. SuMa++: Efficient LiDAR-based Semantic SLAM. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4–8 November 2019.
20. Droeschel, D.; Behnke, S. Efficient Continuous-Time SLAM for 3D Lidar-Based Online Mapping. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–26 May 2018; pp. 5000–5007. [CrossRef]
21. Pumarola, A.; Vakhitov, A.; Agudo, A.; Sanfeliu, A.; Moreno-Noguer, F. PL-SLAM: Real-time monocular visual SLAM with points and lines. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4503–4508.
22. Sánchez, C.M.; Zella, M.; Capitan, J.; Marron, P.J. Efficient Traversability Mapping for Service Robots Using a Point-cloud Fast Filter. In Proceedings of the 2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2–6 December 2019; pp. 584–589.
23. Astra Body Tracker. Available online: https://github.com/KrisPiters/astra_body_tracker (accessed on 23 September 2020).
24. McCormac, J.; Handa, A.; Davison, A.; Leutenegger, S. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4628–4635. [CrossRef]
25. Meyer, A.; Salscheider, N.O.; Orzechowski, P.F.; Stiller, C. Deep Semantic Lane Segmentation for Mapless Driving. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 869–875. [CrossRef]
26. Sünderhauf, N.; Pham, T.T.; Latif, Y.; Milford, M.; Reid, I. Meaningful maps with object-oriented semantic mapping. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 5079–5085. [CrossRef]
27. Nakajima, Y.; Tateno, K.; Tombari, F.; Saito, H. Fast and Accurate Semantic Mapping through Geometric-based Incremental Segmentation. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 385–392. [CrossRef]
28. Ma, L.; Stückler, J.; Kerl, C.; Cremers, D. Multi-view deep learning for consistent semantic mapping with RGB-D cameras. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 598–605. [CrossRef]

29. Dewan, A.; Oliveira, G.L.; Burgard, W. Deep semantic classification for 3D LiDAR data. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3544–3549. [CrossRef]
30. Yin, H.; Wang, Y.; Ding, X.; Tang, L.; Huang, S.; Xiong, R. 3D LiDAR-Based Global Localization Using Siamese Neural Network. *IEEE Trans. Intell. Transp. Syst.* **2019**, 1–13. [CrossRef]
31. Dewan, A.; Caselitz, T.; Tipaldi, G.D.; Burgard, W. Motion-based detection and tracking in 3D LiDAR scans. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweeden, 16–21 May 2016; pp. 4508–4513. [CrossRef]
32. Yan, Z.; Duckett, T.; Bellotto, N. Online learning for human classification in 3D LiDAR-based tracking. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 864–871. [CrossRef]
33. Fahn, C.; Lee, C.; Yeh, Y. A real-time pedestrian legs detection and tracking system used for autonomous mobile robots. In Proceedings of the 2017 International Conference on Applied System Innovation (ICASI), Sapporo, Japan, 13–17 May 2017; pp. 1122–1125.
34. Masi, I.; Wu, Y.; Hassner, T.; Natarajan, P. Deep Face Recognition: A Survey. In Proceedings of the 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Parana, Brazil, 29 October–1 November 2018; pp. 471–478.
35. Rafi, U. Efficient Deep Learning Methods for Human Pose Estimation. Ph.D. Thesis, RWTH Aachen University, Aachen, Germany, 2018. [CrossRef]
36. CE30 SDK for RoS. Available online: https://github.com/CE30C/SDK-for-ROS (accessed on 23 September 2020).
37. ORBBEC. Astra Camera SDK for ROS. Available online: https://github.com/orbbec/ros_astra_camera (accessed on 23 September 2020).
38. Labbé, M. Find-Object. 2011. Available online: http://introlab.github.io/find-object (accessed on 23 April 2020).
39. Xuan, Z.; David, F. Real-Time Voxel Based 3D Semantic Mapping with a Hand Held RGB-D Camera. 2018. Available online: https://github.com/floatlazer/semantic_slam (accessed on 23 April 2020).
40. Aubry, M.; Maturana, D.; Efros, A.A.; Russell, B.C.; Sivic, J. Seeing 3D Chairs: Exemplar Part-Based 2D-3D Alignment Using a Large Dataset of CAD Models. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 3762–3769.
41. Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; Torralba, A. Semantic understanding of scenes through the ade20k dataset. *arXiv* **2016**, arXiv:1608.05442.

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.