Noname manuscript No.
(will be inserted by the editor)

Online Learning for 3D LiDAR-based Human Detection: Experimental Analysis of Point Cloud Clustering and Classification Methods

Zhi Yan · Tom Duckett · Nicola Bellotto

Received: date / Accepted: date

Abstract This paper presents a system for online learning of human classifiers by mobile service robots using 3D LiDAR sensors, and its experimental evaluation in a large indoor public space. The learning framework requires a minimal set of labelled samples (e.g. one or several samples) to initialise a classifier. The classifier is then retrained iteratively during operation of the robot. New training samples are generated automatically using multi-target tracking and a pair of "experts" to estimate false negatives and false positives. Both classification and tracking utilise an efficient real-time clustering algorithm for segmentation of 3D point cloud data. We also introduce a new feature to improve human classification in sparse, long-range point clouds. We provide an extensive evaluation of our the framework using a 3D LiDAR dataset of people moving in a large indoor public space, which is made available to the research community. The experiments demonstrate the influence of the system components and improved classification of humans compared to the state-of-the-art.

Keywords Online learning · Human classification · Point cloud clustering · 3D LiDAR-based tracking · Dataset

PACS 87.85.St · 42.68.Wt · 42.79.Qx

Mathematics Subject Classification (2000) 68T40 · 93C85

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645376 (FLOBOT).

Zhi Yan
Distributed Artificial Intelligence and Knowledge Laboratory (CIAD), University of Technology of Belfort-Montbéliard (UTBM), France
E-mail: zhi.yan@utbm.fr

Tom Duckett, Nicola Bellotto
Lincoln Centre for Autonomous Systems (L-CAS), University of Lincoln, UK
E-mail: tduckett, nbellotto@lincoln.ac.uk

1 Introduction

Human detection is a key feature of mobile service robots operating in public and domestic spaces. Besides obvious safety requirements, distinguishing between humans and inanimate objects provides extra information for the robot to plan and adapt its next movement in the environment. Typically, the robot observes the surrounding area through onboard sensors and estimates the location of all the relevant static and/or moving objects within range. In order to recognize whether these objects are humans or not, cues from tracking and trajectory analysis can be exploited for classification purposes.

Most human detection and tracking systems for service robots must deal with both hardware limitations and changing environments. RGB-D cameras can provide colour information and dense point clouds, but their sensing range is usually just a few meters and their field-of-view is usually less than 90° in both horizontal and vertical directions. A SICK-like 2D LiDAR is capable of sensing ranges of several tens of meters but it is difficult to extract useful information from the sparse distribution of points obtained. In response to changing environments, a feature-based classifier usually needs to be re-tuned and often tediously retrained to achieve acceptable performance in new scenarios. An approach to eliminate this dilemma is to train a generalized classifier. However, the latter requires a large amount of labelled data, usually associated with high labor costs.

The application of 3D LiDAR sensors in robotics and autonomous vehicles has grown dramatically in recent years, either used alone [32,38,23,40,43,44,10,25,42] or in combination with other sensors [15,33,46,16], including their use for human detection and tracking. An important specification of this type of sensor is the ability to provide long-range and wide-angle laser scans. In addition, it produces point clouds that become more sparse as the distance in-

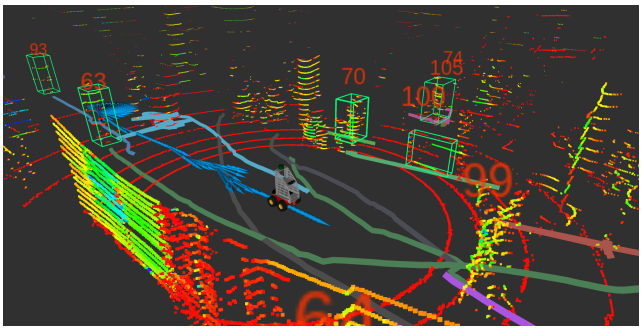


Fig. 1 A screenshot of the 3D LiDAR-based tracking system in action with an online learned human classifier. The detected people are enclosed in green bounding boxes. The colored lines are the people trajectories generated by the multi-target tracking system. The blue arrows indicate the odometry of the robot. Please note that the bounding box with tracking ID 100 is a false positive derived from the classifier.

creases, but which usually are very accurate and not affected by lighting conditions. However, human detection in 3D LiDAR scans is still very challenging, especially when the person is too close or too far away from the robot. Moreover, since increasing the sensing range increases also the area under consideration, and therefore the number of people potentially within it, 3D LiDAR-based human detection can be computationally very expensive. Finally, a large number of manually-annotated training data was usually required by previous methods to learn, offline, a human classifier [32, 23, 33, 46, 16, 25, 42]. Unfortunately, labelling 3D LiDAR point clouds is tedious work and prone to human error, in particular if many variations of human pose, shape and size need to be correctly classified. Offline manual annotation is also not very feasible for complex real-world scenarios and where the same system needs to be retrained for different operational environments.

This paper extends our recent work [47] on human classification, in which we developed a framework for online learning to classify humans in 3D LiDAR scans, taking advantage of a suitable multi-target tracking system [3] (see Fig. 1). Since online learning methods do not rely on explicit human supervision, they are affected by errors. In order to deal with the latter, and inspired by previous solutions for tracking-learning-detection [20], we proposed an online learning approach based on two types of "experts", one converting false negatives into new positive training samples (P-expert), and another one correcting false positives to generate new negative samples (N-expert). We then showed an improvement in the performance of the human classifier by iteratively correcting its errors.

Compared to [47], the current paper includes several new contributions.

- Firstly, we improve the 3D LiDAR-based human classification with a new low-cost feature, coupling human profiles in point clouds with distance changes, hence enhancing the sensitivity of the classifier to samples far away from the robot.
- Secondly, we provide a detailed comparison of our recent 3D LiDAR dataset to other popular ones for pedestrian detection as a guideline for researchers in this area.
- Thirdly, we perform a thorough evaluation of our clustering algorithm for 3D LiDAR point clouds, including a comparison with other state-of-the-art solutions, covering both precision and runtime performance.
- Finally, we extend significantly the performance evaluation of our approach for human classification, including a detailed stability analysis of the online learning process.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work, in particular on 3D point-cloud-based segmentation and human classification. Then, we introduce our online framework in Section 3 and the relation between its modules, including those for human tracking and classification. The former is presented in Section 4, including a detailed description of the clustering algorithm. The actual learning process for human classification is explained in Section 5, which clarifies the link between the P-N experts and tracking. Section 6 presents our publicly available 3D LiDAR dataset and a comparison to the existing ones, while Section 7 provides a comprehensive experimental evaluation of the system performance, including clustering and human classification. Finally, conclusions and future research are discussed in Section 8.

2 Related Work

Human detection and tracking for mobile service robots have been widely studied in recent years, including solutions for RGB [33, 16] and RGB-D cameras [31, 27, 18, 39], as well as 2D [1, 2, 28, 24, 27, 4] and 3D LiDARs [32, 38, 40, 41]. Although the latter provides accurate distance information, one of the main challenges working with LiDARs is the difficulty of recognizing humans from a relatively sparse set of points and without additional color information.

The traditional perception pipeline for object tracking consists of several stages, typically including segmentation (e.g. clustering), feature extraction, classification, data association, and position/velocity estimation. However, emerging methods like end-to-end learning (typically deep learning) provide alternative frameworks for tracking applications. For example, Dequaire *et al.* [9] employed a recurrent neural network (RNN) to capture the environment state evolution with 3D LiDAR (the 3D point clouds were reduced to a 2D scan), training a model in an entirely unsupervised manner, and then using it to track cars, buses, pedestrians and cyclists from an autonomous car. Zhou and Tuzel [50] developed another end-to-end network that divides the 3D point

cloud into a number of voxels. After random sampling and normalization of the points, several Voxel Feature Encoding (VFE) layers are used for each non-empty voxel to extract voxel-wise features. This network is trained to learn an effective discriminative representation of objects with various geometries. Despite encouraging results in this field, we have not integrated deep learning-based approaches into our online framework because such methods typically require considerable fine-tuning with manual intervention, longer training times, and special hardware requirements (e.g. GPU and power supply), making it difficult to meet the requirements of autonomous mobile robots in real-world applications.

The framework described in this paper involves a multi-stage perception pipeline, where the most relevant methods for 3D point cloud segmentation utilise the scan line run [49], depth information [6], and Euclidean distance [36]. The run-based segmentation includes two steps which first extracts the ground surface in an iterative fashion using deterministically assigned seed points, and then clusters the remaining non-ground points using a two-run connected component labeling technique from binary images. Depth clustering is a fast method with low computational demands, which first transforms 3D LiDAR scans into 2D range images, then performs the segmentation on the latter. The Euclidean method, instead, clusters points by calculating the distance between any two of them directly in the 3D space. Our clustering method is compared against these three alternatives in Section 7.1.

Regarding point-cloud-based human classification, a very common approach is to train a classifier offline, under human supervision, and then apply it to sensor data during robot operations. For example, Navarro-Serment *et al.* [32] introduced seven features for human classification and trained an SVM classifier based on these features. Kidono *et al.* [23] proposed two additional features considering the 3D human shape and the clothing material (i.e. using the reflected laser beam intensities), showing significant classification improvements. Háselich *et al.* [17] implemented eight of the above-mentioned features for human detection in unstructured environments, discarding the intensity feature due to a lack of hardware support. Li *et al.* [25] implemented instead a re-sampling algorithm in order to improve the quality of the geometric features proposed by the former authors. Wang and Posner [46] applied a sliding window approach to 3D point data for object detection, including humans. They divided the space enclosed by a 3D bounding box into sparse feature grids, then trained an SVM classifier based on six features related to the occupancy of the cells, the distribution of points within them, and the reflectance of these points. Spinello *et al.* [40] combined a top-down classifier, based on volumetric features, and a bottom-up detector to reduce false positives for tracking distant persons in 3D LiDAR scans. An alternative approach by Deuge *et al.* [10] introduced an unsupervised feature learning approach for outdoor object classification by projecting 3D LiDAR scans into 2D depth images.

Although our paper focuses on a different sensor, the use of 2D LiDARs in human detection and tracking is worth mentioning. Often, this type of sensor is used for robot navigation, including localization and safe obstacle avoidance, and it is therefore installed on the lower front side of the mobile robot, close to the ground. Some researchers have exploited this configuration to perform human tracking by detecting human legs in 2D range data [1,2,28,24,27,4].

The inconvenience with offline methods is that the pre-trained classifier is not always effective when the robot moves to a different environment, and fine-tuning or retraining are typically required as a consequence. To help relieve the user from these tedious tasks, some authors proposed methods requiring less annotation. For example, Teichman *et al.* [44] presented a semi-supervised learning method for multi-object classification, which needs only a small set of hand-labelled seed object tracks for training the classifiers. However, it requires a large training set of background objects (i.e. with no pedestrians, cyclists, cars, etc.), provided by a human expert, in order to achieve good classification performance. Other authors proposed classifier-free methods. Shackleton *et al.* [38], for example, employed a surface matching technique for human detection and an Extended Kalman Filter to estimate the position of a human target and assist the detection in the next LiDAR scan. However, the method is suitable only for simple scenarios, and is further restricted to the case of moving humans. Dewan *et al.* [11] proposed a classifier-free approach to detect and track dynamic objects, which again relies on motion cues and is therefore not suitable for slow and static objects such as pedestrians.

Despite the need for low-annotation or classification-free approaches for human detection, good datasets are important for validation and comparison of new methods to previous ones. Unfortunately, unlike the abundance of datasets collected with RGB and RGB-D cameras[1], there are only a few 3D LiDAR datasets available to the scientific community [40,43,10,15], in particular for mobile service robots. Our dataset introduces a new combination of interesting properties, useful for 3D LiDAR-based human detection and tracking in large indoor environments, which are compared with the existing datasets in Section 6.

To summarize, by looking at the current state-of-the-art, it is clear that an effective low-annotation method would be highly beneficial to detect humans in 3D LiDAR scans. Our work contributes to this need by demonstrating that human detection can be improved by combining tracking and online learning with a mobile robot, even in highly dynamic envi-

[1] `http://www.cvpapers.com/datasets.html`

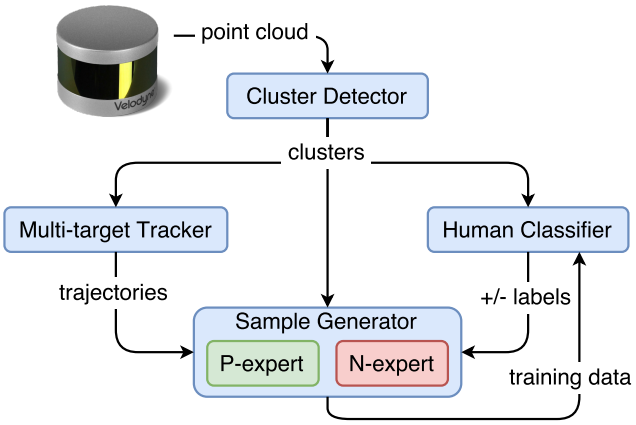


Fig. 2 Process details of the online learning framework.

ronments, and that such an approach provides comparable or superior results with respect to previous methods.

3 General Framework for Online Learning

Our system consists of four main components: a cluster detector for 3D LiDAR point clouds, a multi-target tracker, a human classifier and a training sample generator (see Fig. 2). The classifier is initialised by supervised learning with a small set of human clusters. The size of this set could be as small as one, since more samples will be added incrementally and used for retraining in future iterations.

The online process works as follows. At each new iteration, a 3D LiDAR scan (i.e. 3D point cloud) is segmented by the cluster detector. Positions and velocities of the clusters are estimated in real-time by a multi-target tracking system. These estimates are buffered in trajectory arrays together with their respective cluster observations. At the same time, the classifier labels each cluster as 'human' or 'non-human'.

The sample generator exploits all the information about clusters, trajectories and labels. The latter are typically affected by two types of errors: false positive and false negative. The sample generator tries to correct them by using two independent "experts", which cross-check the output of the classifier with that of the tracker. Based on the experts' decisions, the sample generator produces new training data. In particular, the P-expert in Fig. 2 converts false negatives into positive samples, while the N-expert converts false positives into negative samples. When enough new samples have been generated, they are used to retrain the classifier, so the system can learn and improve from previous errors. The process and the experts are explained in detail in Section 5.2.

Although conceptually similar to a previous tracking-learning-detection framework [20], the proposed systems differs in three key aspects, namely the independence of the tracker from the classifier, the frequency of the training process, and the implementation of the experts. In particular, while the performance of the human classifier depends on the reliability of the experts and the tracker, the latter is completely unaffected by the classification performance. This decoupling makes the system modular and potentially applicable to alternative classification methods. Also, instead of retraining incrementally from single samples (i.e. frame-by-frame training), our system performs a less frequent batch-incremental training [35] (i.e. gathering samples in batches to retrain the classifier after a certain period), collecting new data online as the robot moves in the environment. This feature can effectively prevent under-fitting in online learning. Finally, our implementation of the experts is specifically designed to deal with more than one target simultaneously, and therefore to generate new training samples from multiple detections, speeding up the learning process.

4 Point Cloud Cluster Detection and Tracking

Two key components of the proposed system shown in Fig. 2 are the cluster detector and the multi-target tracker. The former detects, in real-time, clusters of point clouds from 3D LiDAR data. Their positions and velocities are estimated by the tracker, also in real-time, independently of whether the clusters belong to humans or not. Details about the two modules are provided next.

4.1 Point Cloud Cluster Detector

The input of the cluster detector is a 3D LiDAR point cloud $P = \{p_i \mid p_i = (x_i, y_i, z_i) \in \mathbb{R}^3,\ i = 1, \ldots, I\}$, where I is the total number of points in a single scan. In order to discard points belonging to the floor, the detector removes every p_i with $z_i < z_{min}$, leaving a subset $P^* \subset P$. This obviously works well only for relatively flat floors and z-axis roughly perpendicular to ground, which is one of our assumptions, and accepting the fact that small lower parts of an object could be discarded. It is however worth noting that the core idea of our proposed clustering algorithm is not subject to ground removal, and other methods could be used for this in more complex situations [13,30].

As in [36], non-overlapping clusters of adjacent points are extracted based on their 3D Euclidean distance. In particular, if $C_i, C_j \subset P^*$ are two such clusters, the non-overlapping condition can be written as follows:

$$C_i \cap C_j = \emptyset, \text{ for } i \neq j \ \Rightarrow \ min\|p_i - p_j\|_2 \geq d^* \qquad (1)$$

where $p_i \in C_i$, $p_j \in C_j$, and d^* is a distance threshold.

Although relatively fast and simple, distance-based clustering can be inaccurate, especially if the density of points in the cloud varies significantly with the distance from the sensor. If the threshold d^* is too small, single objects could

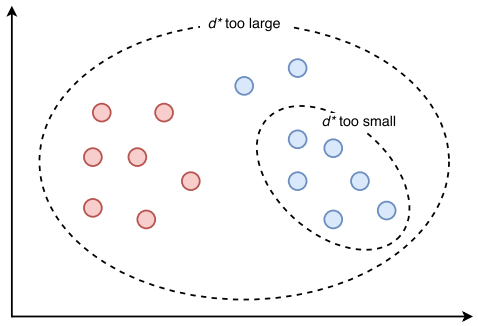


Fig. 3 Point cloud of two distinct objects (red and blue). Different values of d^* lead to different clustering results.

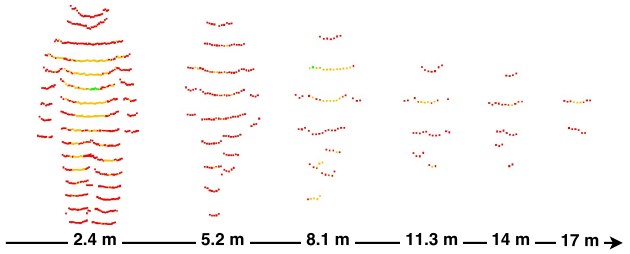


Fig. 4 Example of 3D LiDAR human point clouds at different distances. The further the person is from the sensor, the sparser is the relative point cloud.

be split into multiple clusters. If too large, multiple objects could be merged into a single cluster (see Fig. 3).

In our case, the shape of the point clouds formed by 3D LiDAR scans on the human body can vary significantly with the distance of the person from the sensor. In particular, due to its vertical angular resolution ($\Theta = 2^\circ$), the vertical distance between two consecutive points can be very large, as shown in Fig. 4. We therefore adapt the threshold d^* to the actual scan range r as follows:

$$d^* = 2 \cdot r \cdot \tan \frac{\Theta}{2} \tag{2}$$

This equation simply computes the expected vertical distance between two adjacent scan channels for some straight vertical object.

Unfortunately, clustering of 3D points can be computationally intensive. The longer the maximum scan range, the higher the value of d^*, and therefore the number of points falling within the same cluster. In addition, considering a large area increases the number of potential clusters. To reduce the complexity, we divide the horizontal space into nested circular regions around the sensor, and fix a constant distance threshold within each of them (see Fig. 5).

In practice, we consider a set of values d_i^* at fixed intervals Δd, where $d_{i+1}^* = d_i^* + \Delta d$. For each of them, we compute the maximum cluster detection range r_i using the inverse of Equation (2), and determine the corresponding radius $R_i = \lfloor r_i \rfloor$, where R_0 is centre of the sensor. The width of a region with constant threshold d_i^* is $l_i = R_i - R_{i-1}$. In

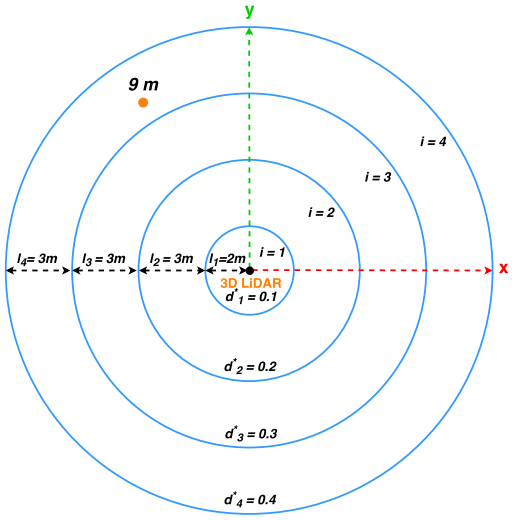


Fig. 5 Different values of d^* correspond to different nested regions. In this example, a cluster (red dot) at 9m from the sensor lies on the 4th circular region, where the clustering threshold is $d_4^* = 0.4$m.

our application, we define circular regions 2-3m wide using $\Delta d = 0.1$m, which is a good resolution to detect potential human clusters.

Finally, clusters that are too large or too small to correspond to humans are filtered out, so that only the following set of cluster detections is actually used by the tracker:

$$\overline{C} = \{C_j \mid 0.2 \leq w_j \leq 1,\ 0.2 \leq d_j \leq 1,\ 0.2 \leq h_j \leq 2\} \tag{3}$$

where w_j, d_j and h_j are the width, depth and height (in meters), respectively, of the volume containing C_j. The time complexity of our proposed clustering algorithm is $\mathcal{O}(\log n)$ with an implementation based on a k-d tree.

4.2 Multi-target Tracker

The multi-target tracker is based on an efficient Unscented Kalman Filter (UKF) implementation, using Global Nearest Neighbour (GNN) data association to deal with multiple clusters simultaneously [2, 3]. Human clusters are tracked on a 2D plane, corresponding to a flat floor, estimating horizontal coordinates and velocities with respect to a fixed world frame of reference. In the current implementation, the 3D cluster size is not taken into account, although it could prove beneficial in more challenging tracking scenarios.

The prediction step of the estimation is based on the following constant velocity model [26]:

$$\begin{cases} x_k = x_{k-1} + \Delta t\ \dot{x}_{k-1} \\ \dot{x}_k = \dot{x}_{k-1} \\ y_k = y_{k-1} + \Delta t\ \dot{y}_{k-1} \\ \dot{y}_k = \dot{y}_{k-1} \end{cases} \tag{4}$$

where x_k and y_k are the Cartesian coordinates of the target at time t_k, $\dot{x}_k$ and $\dot{y}_k$ are the respective velocities, and $\Delta t = t_k - t_{k-1}$.

The position of a cluster is computed by projecting onto the (x,y) plane its centroid c_j, computed as follows:

$$c_j = \frac{1}{|C_j|} \sum_{p_i \in C_j} p_i \tag{5}$$

The update step of the estimation then uses a 2D polar observation model to represent the position of the cluster:

$$\begin{cases} \theta_k & = tan^{-1}(y_k/x_k) \\ \gamma_k & = \sqrt{x_k^2 + y_k^2} \end{cases} \tag{6}$$

where θ_k and γ_k are the bearing and the distance, respectively, of the cluster from the sensor. Note that noises and coordinate transformations, including those relative to the robot motion, are omitted in the above equations for the sake of simplicity.

The choice of the above models and estimation technique is motivated by the type of sensor used. In particular, the polar observation model better represents the actual functioning of our LiDAR sensor (i.e. the raw measures are range values at regular angular intervals) and its noise properties. The non-linearity of this model leads therefore to the adoption of the UKF, which is known to perform better than a standard EKF [19,3]. Previous studies [3,27] showed that our estimation framework is an effective and efficient solution to track multiple people with mobile robots. More details about our track management solution (i.e. initialisation, maintenance, deletion) and possible application can be found in [2,3,12].

Finally, the covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ of the noises for the prediction and observation models, respectively, are the following [26]:

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^4}{4}\sigma_x^2 & \frac{\Delta t^3}{2}\sigma_x^2 & 0 & 0 \\ \frac{\Delta t^3}{2}\sigma_x^2 & \Delta t^2\sigma_x^2 & 0 & 0 \\ 0 & 0 & \frac{\Delta t^4}{4}\sigma_y^2 & \frac{\Delta t^3}{2}\sigma_y^2 \\ 0 & 0 & \frac{\Delta t^3}{2}\sigma_y^2 & \Delta t^2\sigma_y^2 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix} \tag{7}$$

where the noise standard deviations σ_x, σ_y, σ_θ and σ_γ were empirically determined to optimize the human tracking performance of our robot platform.

5 Online Learning for Human Classification

The point cloud clusters detected in Section 4.1 are analysed, in real-time, by a classifier distinguishing between humans and non-humans. Our online learning framework is an iterative process in which the classifier is periodically retrained, online, using old and new cluster detections, which are provided by the sample generator. The latter selects, based on tracking information, a pre-defined number of positive and negative clusters, and uses them to retrain the human classifier. The classification and sample generation processes are explained in detail in the following sub-sections.

5.1 Human Classifier

A standard Support Vector Machine (SVM) [8] is used for human classification. The SVM method has a solid theoretical foundation in mathematics, which is good at dealing with small data samples and therefore very suitable for our proposed online learning framework. Moreoever, this algorithm is known to work well in non-linear classification problems, and has already been applied successfully for 3D LiDAR-based human detection [32,23].

In order to train the SVM, we extract seven different features from each point cloud cluster, which are shown in Table 1. Features $(f_1,\ldots,f_7)$ were proposed by [32]. However, we discarded the last three because of their heavy computational cost and relatively low classification performance [23], which make them unsuitable for real-time people tracking in large populated environments, and replaced them instead with three different features: f_8, f_9 and f_{10}. The former two, f_8 and f_9 were originally proposed by [23]. The combination of these two features with $(f_1,\ldots,f_4)$ was shown to successfully characterise both standing and sitting people [47].

Feature f_{10}, instead, is a new addition to our system, which aims at coupling the detection distance of the sensor with the 3D shape of humans, especially for long distance classification with a sparse point cloud. This new feature, called "slice distance", is based on the 10 slices of [23] and is computed as follows:

$$f_{10} = \{\|c_i\|_2 \mid c_i = (x_i, y_i, z_i) \in \mathbb{R}^3, i = 1,\ldots,10\} \tag{8}$$

where c_i is the centroid of each slice, which can be calculated as in Equation (5). The purpose of the 10 slices was to extract the partial features of humans observed from a long distance, where the spatial resolution of the LiDAR's point clouds decreases. The 3D points in the cluster are divided into 10 blocks, and the length and width of each block are computed as features (see Fig. 6). The respective feature vector is the following:

$$f_8 = \{L_j, W_j \mid j = 1,\ldots,10\} \tag{9}$$

In Section 7.3 we will show that, by using our feature set, the classifier improves on the state-of-the-art.

The full set of features from cluster C_j forms a vector $(f_1,\ldots,f_4,f_8,\ldots,f_{10})$, with 71 dimensions in total. At each iteration of the learning process, a binary SVM is trained to label human and non-human clusters based on these features. The software tool we use is LIBSVM [7], setting the

Table 1 Features for human classification*

Feature	Description	Dimension
f_1	Number of points included in the cluster	1
f_2	Minimum cluster distance from the sensor	1
f_3	3D covariance matrix of the cluster	6
f_4	Normalized moment of inertia tensor	6
f_5	2D covariance matrix in 3 zones including the upper half, the left and right lower halves	9
f_6	The normalized 2D histogram for the main PC plane	98
f_7	The normalized 2D histogram for the secondary PC plane	45
f_8	Slice feature for the cluster	20
f_9	Reflection intensity's distribution (mean, standard dev. and normalized 1D histogram)	27
f_{10}	Distance from the centroid of each slice to the sensor	10

*Rows with gray background indicate the features used in our approach.

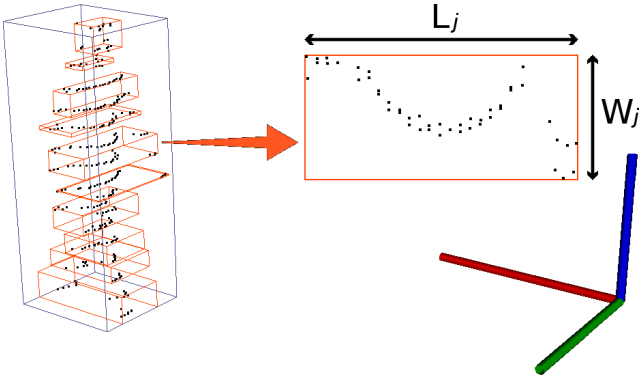


Fig. 6 The slice feature proposed by [23].

ratio of positive and negative training samples to $1:1$, and scaling all the feature values within the interval $[-1,1]$. The SVM uses a Gaussian Radial Basis Function kernel [22] and outputs probabilities associated to the labels. Currently our software does not support incremental learning, so it stores all the training samples since the beginning and uses all of them to retrain the SVM at each new iteration. The training/retraining time is proportional to the number of samples. For our experimental configuration in Section 7, it takes from less than 1 millisecond to a few minutes. However, based on our released source code, one can easily decouple the training process from the online learning (e.g. by using independent threads) as needed, or fine-tune the k-fold cross validation (for finding optimal training parameters) to speed up the training process. Moreover, our framework for online learning allows for the implementation of different classifiers and training algorithms.

5.2 Sample Generator

The training samples needed to retrain the human classifier are selected from the current cluster detections by the sample generator in Fig. 2. This is based on two independent modules: a positive (P) expert and a negative (N) expert. At each time step, the P-expert analyses the current clusters classified as non-humans to identify the potentially incorrect ones (i.e. false negatives). The latter are added to the training set as new positive samples. Conversely, the N-expert examines the current clusters classified as humans, identifies the wrong ones (i.e. false positives), and adds them to training set as new negative samples. This process is repeated several times, until a sufficient number of new positive and negative samples have been collected. Then the human classifier is retrained with the augmented training set. In practice, the P-expert increases the generality of the classifier, while the N-expert increases the discriminability. The implementation details are described in Algorithm 1.

Algorithm 1: Sample Generation and Retraining

Data: H: human classifier, C: clusters,
C_h: human clusters, C_n: non-human clusters,
S_p: positive samples, S_n: negative samples,
T_p: positive training set, T_n: negative training set,
p: number of iterative positive training samples,
n: number of iterative negative training samples,
i: number of iterations,
m: maximum number of iterations
Result: H: human classifier
$T_p \longleftarrow S_p$;
$T_n \longleftarrow S_n$;
$train(H, T_p, T_n)$;
$i \longleftarrow 1$;
repeat
  repeat
    $[C_h, C_n] \longleftarrow Classify(H, C)$;
    $S_p \longleftarrow P_{expert}(C_h, C_n)$;
    $S_n \longleftarrow N_{expert}(C_h, C_n)$;
    $T_p \longleftarrow T_p + S_p$;
    $T_n \longleftarrow T_n + S_n$;
  until $|T_p| >= p * i$ **and** $|T_n| >= n * i$;
  $train(H, T_p, T_n)$;
  $i \longleftarrow 1 + 1$;
until $i > m$;

The P-expert selects new positive samples based on the tracked trajectories of the detected clusters. In particular, clusters classified as non-human, but belonging to a trajec-

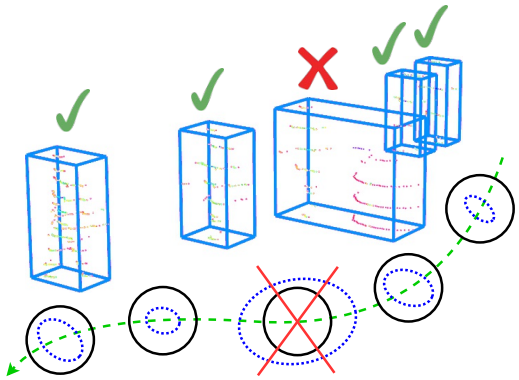


Fig. 7 Example of human-like trajectory samples, including one (red-crossed) filtered out because too uncertain. The green dashed line is the target's trajectory, while the blue dashed circles are the position's uncertainties.

tory where at least one cluster was classified as human, will be added to the training set as positive samples. The conditions to be satisfied by such new positive samples are as follows:

1. the cluster is continuously tracked for the time interval $K\ \Delta t$, during which it covers a minimum distance r^p_{min}:

$$r_k = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \ \text{ and } \ \sum_{k=1}^{K} r_k \geq r^p_{min} \quad (10)$$

2. its velocity is not null but also not faster than a person's preferred walking speed $v^p_{max} = 1.4$ m/s [29]:

$$v_k = \sqrt{\dot{x}_k^2 + \dot{y}_k^2} \ \text{ and } \ v^p_{min} \leq v_k \leq v^p_{max} \quad (11)$$

3. the variances (σ_x^2, σ_y^2) of its estimated position (x_k, y_k) satisfy the following condition:

$$\sigma_x^2 + \sigma_y^2 \leq (\sigma^p_{max})^2 \quad (12)$$

The values of K, r^p_{min}, v^p_{min} and σ^p_{max} used in our system were empirically tuned before the experiments. The last condition, in particular, is particularly useful to filter out clusters that, even if associated with "human-like" trajectories, have a high level of uncertainty because of sudden movements or the proximity of other clusters (see Fig. 7).

The N-expert analyses clusters classified as humans and selects those which are potential false positives, transforming them into new negative samples for future retraining. This selection is based on the assumption that humans are not *completely* static, as there are always some changes in a cluster's shape and its centroid position, even if the person is simply standing or sitting. Taking advantage of the 3D LiDAR's high accuracy, these static clusters (considered as negative samples) can be identified if they satisfy the following conditions:

$$r_k \leq r^n_{max} \ \text{ and } \ v_k \leq v^n_{max} \ \text{ and } \ \sigma_x^2 + \sigma_y^2 \leq (\sigma^n_{max})^2 \quad (13)$$

The parameters r^n_{max}, v^n_{max}, and σ^n_{max} were determined empirically for the experiments. The performance of the P-N experts with respect to the stability of the online learning process is also discussed and evaluated in Section 7.2.

6 System Setup and Dataset

Our framework has been fully implemented into ROS (Robot Operating System) [34] with high modularity. All components are ready for download[2] and use by other researchers. We evaluated the framework on a dataset[3] collected with a mobile robot and a Velodyne VLP-16 3D LiDAR in a large university building.

6.1 Recording Platform

The 3D LiDAR provides 16 scan channels with 360° horizontal and 30° vertical field-of-view. The sensor was mounted on the top of a Pioneer 3-AT robot, as shown in Fig. 8. Its supporting frame and placement resemble the design of the industrial cleaning robot developed in the EU project FLOBOT[4]. The inner mechanism rotates counter-clockwise on its vertical axis, at rates ranging between 300 and 1,200 revolutions per minute (RPM), equivalent to 5 and 20 Hz, respectively. The scanning resolution is inversely proportional to the rotation speed. However, changing the spin rate does not change the data rate, because the horizontal angular resolution is automatically adjusted by the sensor. This means the unit sends out always the same number of packets at a rate of 0.3 million points per second, regardless of the selected spin rate.

In our case, the 3D LiDAR was set to rotate at 10 Hz (i.e. 600 RPM) with a maximum scan range of 100 m. The robot maximum speed was set to 2.52 km/h, and the maximum turning speed to 140°/s. Both robot and sensor were connected to an embedded PC, mounted on the same platform, with an Intel i7-4765T processor and 8GB memory. In addition, a fish-eye camera was mounted on the top of the 3D LiDAR and connected to the PC via WiFi, so panoramic images of the environment around the robot were recorded and used for data annotation[5].

All the software for data recording and robot motion was implemented in ROS, running on Linux Ubuntu 14.04 LTS (64-bit). The 3D point clouds were provided by the Velodyne's official ROS driver and calibration file. All data were recorded in *rosbag* files for easy sharing with the robotics

[2] `https://github.com/LCAS/online_learning`

[3] `http://lcas.lincoln.ac.uk/wp/research/data-sets-software/l-cas-3d-point-cloud-people-dataset/`

[4] `http://www.flobot.eu`

[5] Due to privacy issues, currently panoramic images are not included in the dataset.

Fig. 8 Our hardware platform, in which the (1) Velodyne VLP-16 3D LiDAR is mounted at 0.8m from the floor, on the top of the Pioneer 3-AT robot.

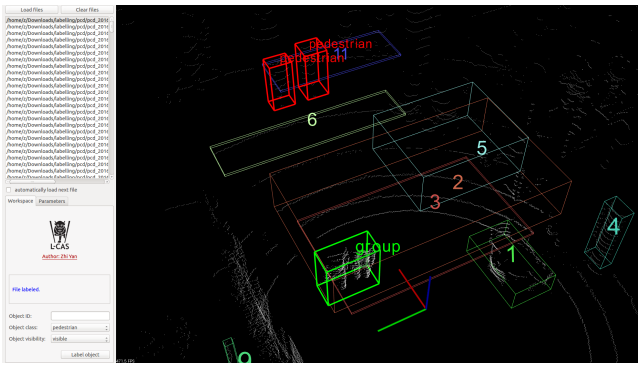


Fig. 9 Screenshot of our annotation tool in operation.

Table 2 Overview of our dataset

Type	Start time	Length
Stationary	12:00	19 minutes
Moving	12:27	12 minutes
Stationary	12:39	18 minutes

community. Moreover, together with the point cloud data, the robot odometry and the coordinate transformations (sensor positions) were also recorded into the *rosbag* files. All the data is timestamped, providing essential information for ground-truth where any object (robot, sensor or human) can be accurately localized.

6.2 Data Annotation

We developed an open-source Qt-based tool for data annotation (see Fig. 9), also available with our dataset. The tool provides a semi-automatic labelling function commonly used by similar 2D image and video annotation software [21, 5]. In our case, the 3D point cloud (from a PCD file[6]) is first segmented into candidate clusters to be labelled by a user, who can indicate information such as the candidate's ID, category (e.g. pedestrian, group[7], etc.) and visibility (e.g. visible or partially visible). The tool allows to modify and add new annotations to previous datasets. It can be also easily extended to label other objects, such as bicycles or cars [42], or to segment and annotate tracklets.

6.3 L-CAS 3D Point Cloud People Dataset

The dataset contains 28,002 scans, recorded with our mobile robot, both while moving and stationary in a large and crowded academic building (see Fig. 10), as detailed in table 2. Each 3D scan contains around 30,000 points. In addition to the X-Y-Z coordinates, each point data includes the scan channel's number and the intensity of the returned light pulse. The latter is very useful for human classification (see feature f_9 in Table 1), since each material has a unique reflection characteristic in the near-infrared region of the laser beam, which can help distinguish between human bodies (i.e. clothing) and other objects [23].

A total of 5,492 scans were manually annotated on the 19 minutes of stationary data, which contains 6,140 human clusters. The minimum and maximum number of 3D points generated by the scan of a person are 3 and 3,925, respectively, while the minimum and maximum distance from the sensor are 0.5 m and 27 m, respectively. As shown in Fig. 11, the dataset captures many challenging situations, such as human groups, children, people with trolleys, etc., the combination of which is seldom included in similar datasets.

A comparison of our 3D LiDAR dataset with other similar ones can be seen in Table 3. To our knowledge, this is the only such dataset collected indoors with a moving and stationary robot. Although the scan channels and the resolution of the Velodyne VLP-16 are lower than other models, the quality of the data provided by our sensor is perfectly suitable for most indoor applications, allowing us to detect and track people up to 25 m from the robot. Our dataset provides also the second largest number of laser scans, stored in two file formats, i.e. *bag* for ROS and *pcd* for Point Cloud Library (PCL) [37], which are both very popular in the robotics community.

7 Experimental Results

The experiments reported in this paper were carried out on a laptop with an Intel i7-7560U (2.40GHz) processor and

[6] `http://pointclouds.org/documentation/tutorials/pcd_file_format.php`

[7] Although the "group" samples are not used in this paper, these annotations are included in our dataset to be used by other researchers for group tracking or other applications.

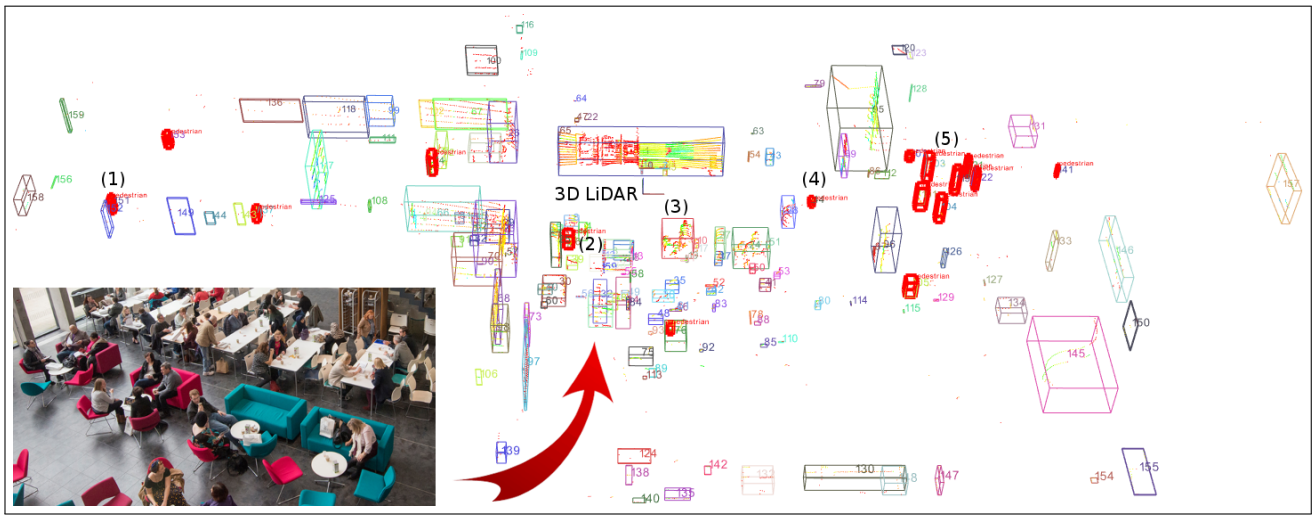


Fig. 10 Example of annotated laser scan of the academic building. Ground and ceiling points have been removed. Each bounding box represents a cluster, and red boxes are humans. The image contains some typical annotation challenges: (1) human sample at 25 m from the sensor consisting of 5 points only; (2) cluster of a sitting person, included the chair; (3) cluster of two people sitting at a circular table, who cannot be labelled as a humans; (4) human head only, the rest of the body is occluded; (5) clusters of people with many occlusions.

Table 3 Comparison of existing 3D LiDAR datasets.

Dataset	Recording setting	Sensor model	Sensor rate/range/altitude	#Scans
Spinello *et al.* [40]	outdoor (stationary)	Velodyne HDL-64E S2	5 Hz, 20 m, unknown	1,402 (.ezd)
Geiger *et al.* [15]	outdoor (moving)	Velodyne HDL-64E	10 Hz, 100 m, 1.73 m	14,999 (.bin)
Teichman *et al.* [43]	outdoor (stationary & moving)	Velodyne HDL-64E S2	10 Hz, unknown	1,300,000 (.tm)
Deuge *et al.* [10]	outdoor (stationary)	Velodyne HDL-64E	unknown	41 (.bin & .csv)
L-CAS dataset [47]	indoor (stationary & moving)	Velodyne VLP-16	10 Hz, 100 m, 0.8 m	28,002 (.bag & .pcd)

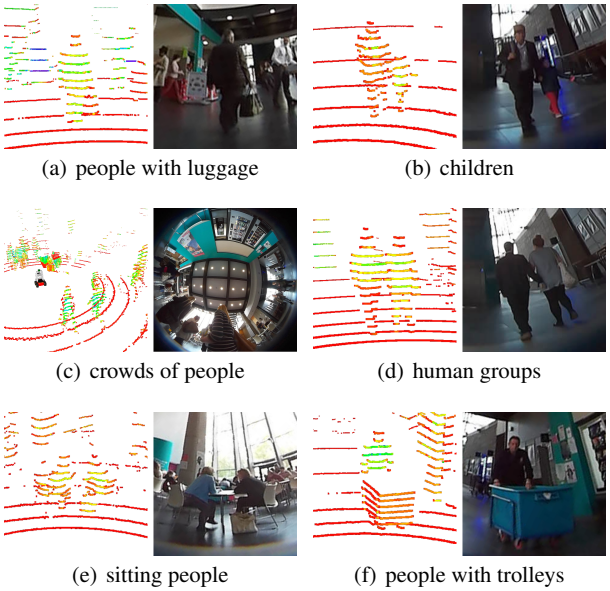
#Points per scan	#Annotations	Occlusion label	Track label	Ground-truth	Annotation tool
120,000	5,277 pedestrian	visible, partially visible	✓		
100,000	4,487 pedestrian (orientation)	visible and 3 occlusion types	✓	✓	✓
120,000	54,484 pedestrian	unavailable	✓		
unknown	152 pedestrian	unavailable			
30,000	4,811 pedestrian, 3,054 group	visible, partially visible			✓

16GB memory (Ubuntu 16.04 64-bit + ROS Kinetic) for the comparison of the clustering precision and runtime as well as the online classification under uncertainty, and a laptop with an Intel i7-6560U (2.20GHz) processor and 8GB memory (Ubuntu 14.04 64-bit + ROS Indigo) for the rest. Experiments for evaluating the clustering performance were run using only one core of the CPU.

7.1 Clustering Performance

The first experiment evaluated the runtime performance of our point cloud clustering algorithm. We used the 19 minutes of data for the stationary case, and reported the operating frequency at different detection distances. The results in Fig. 12 shows that the frequency decreases as the distance increases, in particular during the first few meters, where there is a large drop from ~50 Hz to ~20 Hz. This is due to the fact that the greater the distance from the sensor, the more laser points are considered. In particular, there are fewer obstacles (e.g. pedestrians) within the first few meters, and therefore fewer laser beams reflected by them that are taken into account by the clustering algorithm. Due to some code optimization, the results are also much improved in comparison to our previous work [47] and show that our clustering frequency is well above the minimum 5 Hz typically required for human tracking [3].

We then compared our approach against the current state-of-the-art in terms of precision and runtime. For the former, we considered a large open space, where a person walked towards the 3D LiDAR starting at 25 m and finishing at 1 m from the sensor, while a second person walked from one side to another at 19 m and 11 m, respectively, from the sensor. Fig. 13 depicts a moment of the experimental scenario. The whole process took 17 seconds, with the 3D LiDAR running at 10 Hz. In total, 172 scans were acquired and subsequently



(a) people with luggage (b) children

(c) crowds of people (d) human groups

(e) sitting people (f) people with trolleys

Fig. 11 Different challenging scenarios captured by our dataset.

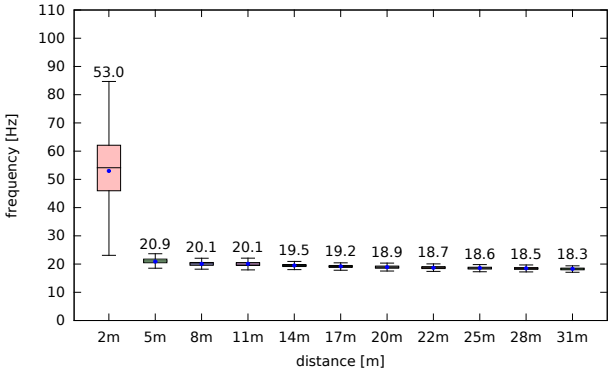


Fig. 12 Runtime performance of the clustering algorithm varying the maximum LiDAR detection range. For example, $2m$ refers only to the measurements (points) within this particular distance. The respective mean is shown on the top of each box. As one can see, the amount of clutter (and therefore detected clusters) is inversely proportional to the considered distance from the sensor. This is consistent with the complexity of the algorithm, i.e. $\mathcal{O}(\log n)$.

fully labelled[8] to analyse the precision of the cluster detector.

We compared our method to the run-based clustering in [49], depth clustering in [6], and the Euclidean clustering of PCL [37], evaluating the segmentation precision on the above labelled scans. In particular, we counted how many human clusters were correctly segmented, noting "missed" if no clusters were present, "over-segmented" if a person's point cloud was split into multiple clusters, and "under-segmented" if the cluster size exceeded 30% of the manually labelled one. The parameters of four clustering algorithms and results are shown in Table 4. For the run-based method, we used the parameter values reported in [49]. For the depth and the Euclidean methods, we used the thresholds reported to give the best performance [6]. Note that the main reason for the low performance of the depth clustering is that the latter is based on a 2D range image, where points are clustered based on the angle between laser beams. This makes it difficult to distinguish between objects that are very close in depth, or relatively small objects against larger ones in the background, affecting in particular the detection of person 2 in the experiment. As shown by the table, instead, our method can handle almost all of these situations, giving a precision which is generally much higher than that of the other approaches.

[8] The data is available at: `https://github.com/LCAS/cloud_annotation_tool`

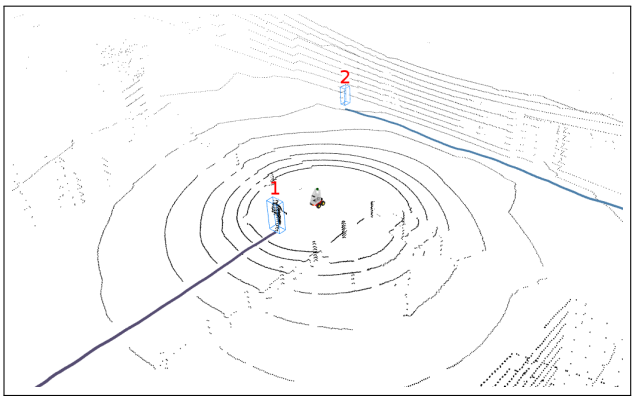


Fig. 13 Two people walking in proximity of the robot in a large open space, one towards the robot and another one from side to side.

Regarding runtime performance, we tested the same four methods on the first 2,500 scans of the 19 minutes of data for the stationary robot. The results in Fig. 14 show that the run clustering, depth clustering and our method are an order of magnitude faster than the Euclidean approach. The run method is slightly faster than the depth one given the fact that relatively more points were removed along with the ground in its first step, winning the time required for the second step of clustering. The run method is three times and the depth method is twice as fast as ours, but at the expense of precision, as previously demonstrated.

From the above experiments we can draw the following conclusion. The run and the depth methods consider the clustering from the perspective of 2D images, while we directly process the 3D point cloud. This results in the first two methods being faster but not accurate enough, while ours is relatively slower but obtains higher precision.

7.2 Stability Analysis

A stability analysis of the learning process is possible by considering the variations of false positives α and false neg-

Table 4 Parameter settings and segmentation precision of four different methods.

Approach	Ground removal	Min/Max points	Clustering threshold	Precision
Run clustering [49]	$Params_{GPF}$	2/30,000	$Params_{SLR}$	51.7% (133 missed, 33 over- 0 under-segmented)
Depth clustering [6]	7°	5/30,000	10°	39.2% (162 missed, 11 over- 36 under-segmented)
Euclidean clustering [37]	-0.8 m	5/30,000	0.5 m	64.5% (88 missed, 34 over- 0 under-segmented)
Current paper	-0.8 m	5/30,000	Adaptive	89.8% (1 missed, 16 over- 18 under-segmented)

$Params_{GPF} = \{N_{segs} = 3, N_{iter} = 3, N_{LPR} = 20, Th_{seeds} = 0.4m, Th_{dist} = 0.2m\}$
$Params_{SLR} = \{Th_{run} = 0.5m, Th_{merge} = 1m\}$

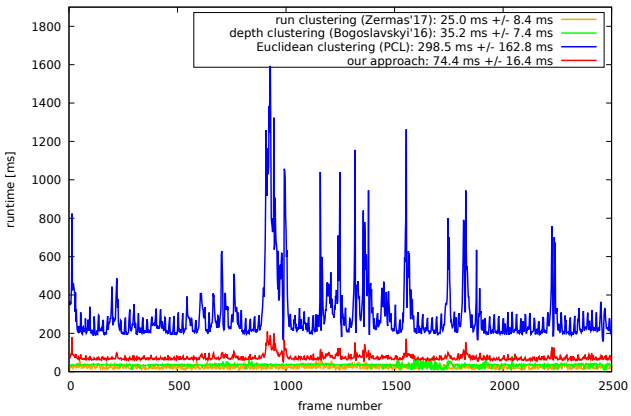


Fig. 14 Runtime performance of the four clustering methods.

Table 5 Performance analysis of the P-N experts.

Retrain the classifier every 5 positives and 5 negatives					
P-expert		N-expert		Eigenvalues	
P^+	R^+	P^-	R^-	λ_1	λ_2
1.0	0.714	1.0	0.0215	**0.9785**	**0.286**
Retrain the classifier every 10 positives and 10 negatives					
P-expert		N-expert		Eigenvalues	
P^+	R^+	P^-	R^-	λ_1	λ_2
1.0	0.8857	1.0	0.0405	**0.9595**	**0.1143**
Retrain the classifier every 20 positives and 20 negatives					
P-expert		N-expert		Eigenvalues	
P^+	R^+	P^-	R^-	λ_1	λ_2
1.0	1.0	1.0	0.0271	**0.9729**	**0**

atives β generated by the human classifier [20]:

$$\begin{bmatrix}\alpha_{k+1}\\ \beta_{k+1}\end{bmatrix} = \begin{bmatrix}1-R^- & \frac{1-P^+}{P^+}R^+\\ \frac{1-P^-}{P^-}R^- & 1-R^+\end{bmatrix} \times \begin{bmatrix}\alpha_k\\ \beta_k\end{bmatrix} \tag{14}$$

where k indicates the training iteration, while P^+, R^+, P^- and R^- are the P-precision, P-recall, N-precision and N-recall of the experts, respectively. The recursive equation in (14) is a discrete dynamical system that can be written as $\mathbf{x}_{k+1} = \mathbf{M}\,\mathbf{x}_k$. It can be shown that, if the eigenvalues λ_1 and λ_2 of $\mathbf{M}$ are less than one, then $\mathbf{x}$ converges to zero.

An interesting observation by looking at (14) is that, if $P^+ = P^- = 1$, then $\mathbf{M}$ becomes a diagonal matrix, and therefore its eigenvalues are simply $\lambda_1 = 1-R^-$ and $\lambda_2 = 1-R^+$. In this case, even a small recall for both experts is sufficient to guarantee $\lambda_1 < 1$ and $\lambda_2 < 1$, and therefore the stability of the learning process. According these criteria, we use the following performance metrics for the P-N experts:

$$\begin{aligned}&\text{- Precision of P-expert} && P^+ = n_t^+/(n_t^+ + n_f^+)\\ &\text{- Recall of P-expert} && R^+ = n_t^+/\beta\\ &\text{- Precision of N-expert} && P^- = n_t^-/(n_t^- + n_f^-)\\ &\text{- Recall of N-expert} && R^- = n_t^-/\alpha\end{aligned} \tag{15}$$

where n_t^+ and n_f^+ are the number of true and false positives, respectively, by the P-expert, while n_t^- and n_f^- are those by the N-expert. The α false positives by the human classifier are the total number of instances that should be corrected by the N-expert. Equally, the β false negatives by the human classifier are the total number of instances that should be corrected by the P-expert.

Table 5 shows the results for 10 training iterations performed on the same 172 scans of the previous experiment. Three classifiers were manually initialised using just one sample, and then retrained every 5, 10 or 20 new positives and an equal number of negative samples, respectively. In this way, the stability of our online learning was verified also for different training intervals. As we can see, the performance of both experts is generally good. In particular, since $P^+ = P^- = 1$ and also $R^+ > 0 \wedge R^- > 0$, the stability of the online learning is always guaranteed by the eigenvalues $\lambda_1 < 1 \wedge \lambda_2 < 1$, as discussed above.

7.3 Classification Performance

In this section we evaluate the performance of the 3D LiDAR-based human classifier, first by analysing the classification results of the SVM trained online and offline, then by assessing the online trained SVM under uncertainty, and finally by comparing the results with our new feature set against other state-of-the-art feature combinations.

7.3.1 Online Versus Offline Classification

We first compared the SVM classifier, trained online by our system, versus the same classifier trained offline with manually labelled data. In order to illustrate the improvement

gained over time by the former, the performance at the beginning and end of the online training is considered. The offline classifier was trained using 1,000 single-person samples from annotated data, with an equal amount of randomly selected background samples. The online classifier was initially trained with 5 positive and 5 negative samples, chosen manually, and then retrained automatically by the system every 100 positive and 100 negative samples, until the final version had been trained by 1,005 positives and 1,005 negatives.

The test set included 50 scans randomly selected from the 18 minutes of our stationary robot dataset (excluding the scans used to manually train the offline classifier). Both standing and sitting people were added to this set. In particular, it contains 567 single persons with respective clusters containing between 5 and 2,122 points, at a varying distance of 0.5 m to 25 m from the sensor.

The classification performance was evaluated using standard metrics [14] including Precision, Recall, Average Precision (AP) and F-measure. A human cluster was considered a true positive if the area of overlap between the predicted and the ground-truth bounding boxes was more than 50%. The results are shown in Fig. 15 (the left column), where the improvement of the final online classifier compared to the initial one is evident. Most importantly, the graph shows that the final online classifier outperforms the one trained manually offline. This is largely due to the fact that our tracking system is robust enough to facilitate the detection of clusters far away from the robot, even when these are very difficult for a human annotator to spot, to the benefit of our online learning framework.

7.3.2 Online Classification Under Uncertainty

We then trained a classifier online with the same settings as in Section 7.3.1 but without the P-N experts. Only human trajectories that satisfy Equations 10, 11 and 12 are learned as positives samples. The negatives samples are randomly selected background samples. The idea is to report on experiments when the P-expert and N-expert fail, and as a consequence the SVM is retrained with a given number of false positives and false negatives. The results are shown in Fig. 15 (the middle column). As we can see, the classification performance is improved as the number of training rounds increases. However, without the help of the experts to correct the errors online, the improvement is slow and limited.

Another experiment was conducted on the 12 minutes of moving data. We ran the full system (i.e. with the P-N experts) and fine-tuned the parameters described in Section 5.2 to cope with the increased noise generated with the robot moving. A classifier was trained online with 8 rounds and we report its performance on even training rounds, as shown in Fig. 15 (the right column). As we can see, the classification performance is improved within our online learning framework but affected by noise. However, it is worth pointing out that in this experiment the tracking system provides the 2D position of people relative to the robot's odometry, in which drift errors are accumulated. Using LiDAR odometry or map-based localization would improve the current results.

7.3.3 Feature Sets Comparison

Finally, four classifiers with different feature sets were evaluated in this experiment to demonstrate the improvement generated by our new slice distance feature f_{10}. The four sets are $(f_1,\ldots,f_7)$ as proposed by [32], $(f_1,\ldots,f_9)$ by [23], our previous set $(f_1,\ldots,f_4,f_8,f_9)$ in [47], and finally the current $(f_1,\ldots,f_4,f_8,\ldots,f_{10})$.

For training, both (positive) human and (negative) background samples were taken from the 19 minutes of stationary robot data, starting with 5 positives and 5 negatives for the initial supervised stage. The classifiers were retrained every 100 positive and 100 negative samples by our online learning framework, until reaching 1,005 positives and an equal amount of negatives. The final classifiers were then evaluated on the same test set used in the previous experiment, with 567 human clusters extracted from the 18 minutes of stationary robot data, including both standing and sitting people at a varying distance from the sensor.

The results are illustrated in Fig. 16, showing that the AP of our new classifier, enhanced by the proposed slice distance feature, is significantly better than our previous solution and, in general, is similar to the other two classifiers. However, instead of using features $(f_5,\ldots,f_7)$ based on projected planes with 152 dimensions, our L^2 distance-based feature f_{10}, with only 10 dimensions, can significantly improve the system performance, and so it is more suitable for real-time human detection. Also, the Precision-Recall and the F-Measure of our new classifier are generally better than that of the others.

8 Conclusions

This paper presented an improved online learning framework for human detection in 3D LiDAR scans, including an extensive evaluation of runtime performance, stability, and classification results. The framework relies on a multi-target tracking system with a real-time clustering algorithm and an efficient sample generator. It enables a mobile robot to learn autonomously from the environment what humans look like, which greatly reduces the need for tedious and time-consuming data annotation.

We showed that our adaptive clustering method is more precise than other state-of-the-art algorithms, while still maintaining a low computational cost suitable for most human

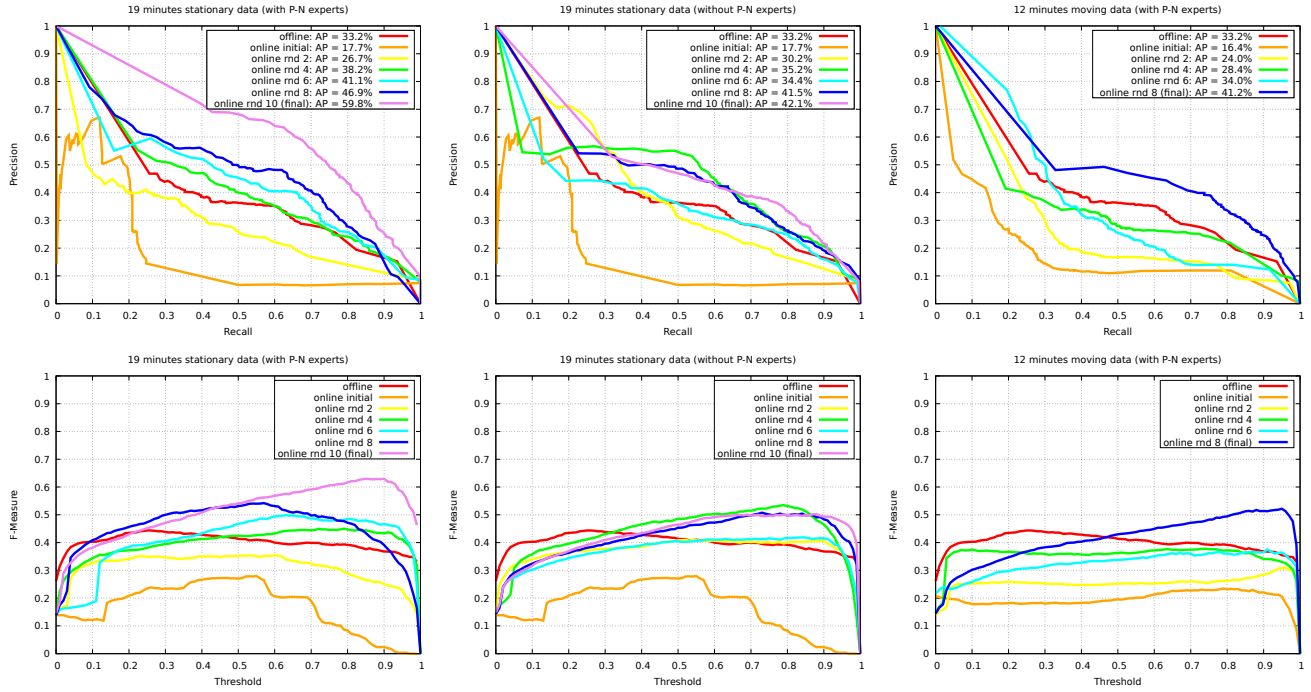


Fig. 15 Performance evaluation of the 3D LiDAR-based human classifier. Left column: online versus offline classification. Middle column: online classification without P-N experts. Right column: online classification with robot moving. The initial online classifiers are trained with manually chosen 5 positive and 5 negative samples.

tracking applications. The stability of the online learning process has been analysed in detail, and has been guaranteed in practice by the high-precision of our P-N expert modules in providing good training samples. Our experiments showed also that, thanks to an augmented set of efficient features, our human classifier performs better than other state-of-the-art solutions in terms of F-measure (accuracy), but with comparable precision and recall.

The whole system is implemented in ROS following a modular design. Both the software and the dataset used in our experiments are publicly available for research purposes. Although currently used for human detection and tracking, our software could be extended to deal with other moving objects such as cars, bicycles or animals [42]. The 3D LiDAR-based cluster detection module could also be replaced by other detectors based on different sensors, such as RGB-D cameras and 2D LiDARs [48].

Future extensions should include coupling the human classifier to the multi-target tracker, so that improving the classification of human clusters also improves people tracking. Moreover, although our solution enables human detection with mobile robots in dynamic environments, in the current paper it has been tested only on datasets recorded for relatively short period of times. Daily routines in public environments could be exploited by a service robot, for example, to collect negative background samples at night, when there are no moving objects, and positive human samples during the day[9]. Long-term operation and open-ended learning are therefore two promising directions for future research in this area [45]. Future work should look at other classification methods such as deep neural networks, exploiting online learning to overcome the difficulty of collecting extensive training samples.

References

1. Arras, K.O., Mozos, O.M., Burgard, W.: Using boosted features for the detection of people in 2d range data. In: Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA), pp. 3402–3407 (2007)
2. Bellotto, N., Hu, H.: Multisensor-based human detection and tracking for mobile service robots. IEEE Transactions on Systems, Man, and Cybernetics – Part B **39**(1), 167–181 (2009)
3. Bellotto, N., Hu, H.: Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters. Autonomous Robots **28**, 425–438 (2010)
4. Beyer, L., Hermans, A., Linder, T., Arras, K.O., Leibe, B.: Deep person detection in two-dimensional range data. IEEE Robotics and Automation Letters **3**(3), 2726–2733 (2018)
5. Bianco, S., Ciocca, G., Napoletano, P., Schettini, R.: An interactive tool for manual, semi-automatic and automatic video annotation. Computer Vision and Image Understanding **131**, 88–99 (2015)
6. Bogoslavskyi, I., Stachniss, C.: Fast range image-based segmentation of sparse 3d laser scans for online operation. In: Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 163–169 (2016)

[9] Dataset collection is also in progress at University of Lincoln and at University of Technology of Belfort-Montbéliard.

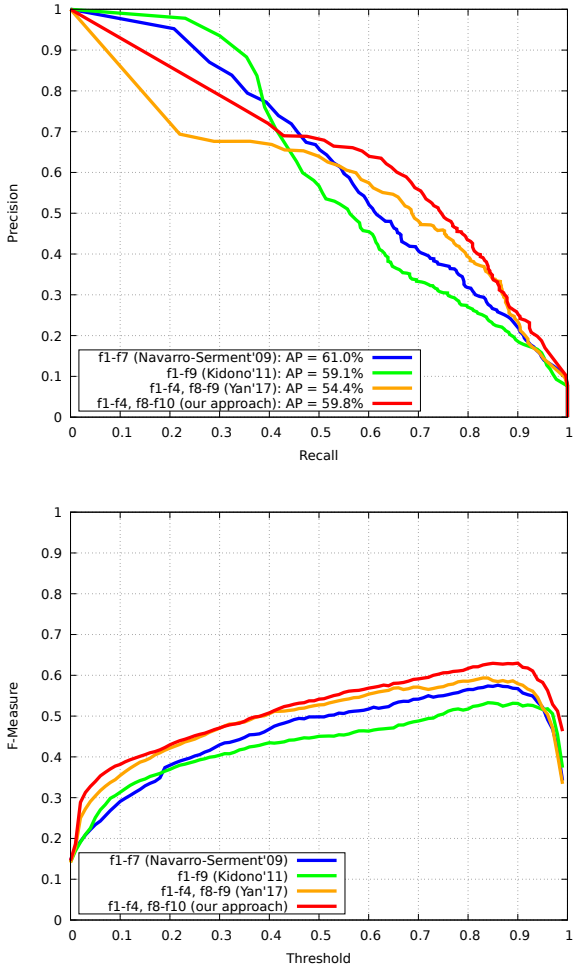


Fig. 16 Performance evaluation of different human classifiers.

7. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 1–27 (2011)
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (1995)
9. Dequaire, J., Ondruska, P., Rao, D., Wang, D., Posner, I.: Deep tracking in the wild: End-to-end tracking using recurrent neural networks. International Journal of Robotics Research (IJRR) pp. 1–21 (2017)
10. Deuge, M.D., Quadros, A., Hung, C., Douillard, B.: Unsupervised feature learning for classification of outdoor 3d scans. In: Proceedings of the Australasian Conference on Robotics and Automation (ACRA) (2013)
11. 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), pp. 4508–4513 (2016)
12. Dondrup, C., Bellotto, N., Jovan, F., Hanheide, M.: Real-time multisensor people tracking for human-robot spatial interaction. In: Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Workshop on Machine Learning for Social Robotics (2015)
13. Douillard, B., Underwood, J.P., Kuntz, N., Vlaskine, V., Quadros, A.J., Morton, P., Frenkel, A.: On the segmentation of 3d LIDAR point clouds. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 2798–2805 (2011)
14. Everingham, M., Gool, L.J.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (VOC) challenge. International Journal of Computer Vision **88**, 303–338 (2010)
15. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3354–3361 (2012)
16. González, A., Villalonga, G., Xu, J., Vázquez, D., Amores, J., López, A.M.: Multiview random forest of local experts combining RGB and LIDAR data for pedestrian detection. In: Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV), pp. 356–361 (2015)
17. Häselich, M., Jöbgen, B., Wojke, N., Hedrich, J., Paulus, D.: Confidence-based pedestrian tracking in unstructured environments using 3d laser distance measurements. In: Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4118–4123 (2014)
18. Jafari, O.H., Mitzel, D., Leibe, B.: Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 5636–5643 (2014)
19. Julier, S.J., Uhlmann, J.K.: Unscented filtering and nonlinear estimation. Proc. of the IEEE **92**(3), 401–422 (2004)
20. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**, 1409–1422 (2012)
21. Kavasidis, I., Palazzo, S., Salvo, R.D., Giordano, D., Spampinato, C.: A semi-automatic tool for detection and tracking ground truth generation in videos. In: Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications (2012)
22. Keerthi, S.S., Lin, C.J.: Asymptotic behaviors of support vector machines with gaussian kernel. Neural Computation **15**(7), 1667–1689 (2003)
23. Kidono, K., Miyasaka, T., Watanabe, A., Naito, T., Miura, J.: Pedestrian recognition using high-definition LIDAR. In: Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 405–410 (2011)
24. Leigh, A., Pineau, J., Olmedo, N.A., Zhang, H.: Person tracking and following with 2d laser scanners. In: Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 726–733 (2015)
25. Li, K., Wang, X., Xu, Y., Wang, J.: Density enhancement-based long-range pedestrian detection using 3-d range data. IEEE Transactions on Intelligent Transportation Systems **17**, 1368–1380 (2016)
26. Li, X.R., Jilkov, V.P.: Survey of maneuvering target tracking. part i: Dynamic models. IEEE Trans. on Aerospace and Electronic Systems **39**(4), 1333–1364 (2003)
27. Linder, T., Breuers, S., Leibe, B., Arras, K.O.: On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In: Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 5512–5519 (2016)
28. Luber, M., Arras, K.O.: Multi-hypothesis social grouping and tracking for mobile robots. In: Proceedings of Robotics: Science and Systems (2013)
29. Mohler, B.J., Thompson, W.B., Creem-Regehr, S.H., Pick, H.L., Warren, W.H.: Visual flow influences gait transition speed and preferred walking speed. Experimental brain research **181**(2), 221–228 (2007)
30. Moosmann, F., Pink, O., Stiller, C.: Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In: Proceedings of the 2009 IEEE Intelligent Vehicles Symposium (IV), pp. 1931–0587 (2009)

31. Munaro, M., Menegatti, E.: Fast RGB-D people tracking for service robots. Autonomous Robots **37**, 227–242 (2014)
32. Navarro-Serment, L.E., Mertz, C., Hebert, M.: Pedestrian detection and tracking using three-dimensional ladar data. In: Proceedings of the 7th Conference on Field and Service Robotics (FSR), pp. 103–112 (2009)
33. Premebida, C., Carreira, J., Batista, J., Nunes, U.: Pedestrian detection combining RGB and dense LIDAR data. In: Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4112–4117 (2014)
34. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
35. Read, J., Bifet, A., Pfahringer, B., Holmes, G.: Batch-incremental versus instance-incremental learning in dynamic and evolving data. In: Proceedings of the Eleventh International Symposium on Intelligent Data Analysis (IDA 2012), pp. 313–323 (2012)
36. Rusu, R.B.: Semantic 3D object maps for everyday manipulation in human living environments. Ph.D. thesis, Computer Science department, Technische Universitaet Muenchen, Germany (2009)
37. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA) (2011)
38. Shackleton, J., Voorst, B.V., Hesch, J.A.: Tracking people with a 360-degree lidar. In: Proceedings of the Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 420–426 (2010)
39. Spinello, L., Arras, K.O.: People detection in RGB-D data. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3838–3843 (2011)
40. Spinello, L., Luber, M., Arras, K.O.: Tracking people in 3d using a bottom-up top-down detector. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 1304–1310 (2011)
41. Sun, L., Yan, Z., Mellado, S.M., Hanheide, M., Duckett, T.: 3DOF pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data. In: In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA). Brisbane, Australia (2018)
42. 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 Robotics and Automation Letters **3**(4), 3749–3756 (2018)
43. Teichman, A., Levinson, J., Thrun, S.: Towards 3d object recognition via classification of arbitrary object tracks. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 4034–4041 (2011)
44. Teichman, A., Thrun, S.: Tracking-based semi-supervised learning. International Journal of Robotics Research (IJRR) **31**(7), 804–818 (2012)
45. Vintr, T., Yan, Z., Duckett, T., Krajnik, T.: Spatio-temporal representation for long-term anticipation of human presence in service robotics. In: Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA). Montreal, Canada (2019)
46. Wang, D.Z., Posner, I.: Voting for voting in online point cloud object detection. In: Proceedings of Robotics: Science and Systems (2015)
47. Yan, Z., Duckett, T., Bellotto, N.: Online learning for human classification in 3D LiDAR-based tracking. In: In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Vancouver, Canada (2017)
48. Yan, Z., Sun, L., Duckett, T., Bellotto, N.: Multisensor online transfer learning for 3d lidar-based human detection with a mobile robot. In: Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid, Spain (2018)
49. Zermas, D., Izzat, I., Papanikolopoulos, N.: Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In: Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA). Singapore (2017)
50. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4490–4499 (2018)