Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 53-2 (2020) 15440–15445

Extended target tracking for autonomous street crossing

Filippo Parravicini [*] Matteo Corno [*] Sergio Savaresi [*]

[*] *The authors are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy. Email:* {filippo.parravicini, matteo.corno, sergio.savaresi}@polimi.it

Abstract: Autonomous navigation on sidewalks and pedestrian areas is a complex problem, that requires the solution of different challenging tasks. One that is particularly hard to tackle is that of autonomous street crossing, which requires the robot to be aware of the position and speed of surrounding vehicles in order to decide whether is safe to cross. This work is dedicated to the development of an obstacle speed estimation algorithm to be applied to the context of autonomous navigation at crosswalks. In particular, a novel approach to the extended-target tracking problem is presented, which leverages a nested structure and a clustering algorithm that reduces the problem to a standard target tracking one. The effectiveness of the algorithm is demonstrated through testing on a prototype parcel-delivery robot operating in a real-world urban environment.

Keywords: target tracking, mobile robots

1. INTRODUCTION

Recently, the interest in mobile robotics applied to sidewalks has been growing. The main use-case prompting this type of research is that of transportation of goods. A high level of autonomy is in general desirable, with the intent of having fleets of urban-navigating robots operating with minimum supervision. In this context, one of the most challenging tasks to be tackled is that of autonomously navigating crosswalks and intersections. This is because in those places the robot interacts with non-pedestrian road users, like cars, trucks or motorcycles. Robotic vehicles are therefore required to be able of detecting incoming vehicles and estimating their velocity in order to decide when is the best moment to initiate crossing. Whereas the general problem of autonomous navigation at crosswalks has scarcely been investigated in the literature, the related speed estimation problem can be cast into a target tracking framework. Target tracking is an extensively explored problem in which the goal is to estimate the actual and future trajectory of an obstacle moving in the surrounding environment, given a collection of measurements taken from an exteroceptive sensor. Typically utilized sensors in this field are radars, Lidars and cameras, but any sensor measuring the distance of surrounding obstacles could also be used. Standard target tracking algorithms can be clustered based on the choice of motion model (see Li and Jilkov (2003)), data association and filtering algorithm (see Bar-Shalom and Osborne (2015)), and track management techniques (see Vo et al. (1999)). A further classification separates the problem of tracking a single object from that of simultaneously tracking multiple dynamics objects, which is referred to as Multiple Target Tracking (MTT). A common assumption in traditional MTT algorithms were developed under the so called "small-object" assumption, according to which each obstacle in the scene generates at most one measurement per each time instant. This hypothesis is reasonable when low resolution sensors are utilized and in case of tracking remote objects which are very far from the sensor. However, this is generally not true in mobile robotics, in which obstacles are typically detected using Lidars or stereocameras which produce many measurement points for each of them. As such, applying MTT to autonomous navigation requires a more complex model, in which each obstacle produces in general more than one measurement, so that not only the number of obstacles is unknown, but also which and how many measurements should be associated to each of them is not known a priori. This problem is known as Extended-Target Tracking (ETT); a detailed analysis of the various approaches to the solution of this problem is presented in Granstrom et al. (2016). Most of the approaches to ETT are based on including the shape of the object in the tracks model, which is done by means of more or less complex statistical models. The price to be paid is either given by the model complexity or by the hypothesis that need to be done on the class of shapes which are considered for the obstacles, and on the nature of the stochastic objects used to model the measurement process. To overcome these limitations we propose a cascade approach in which standard MTT techniques are combined with a clustering algorithm to reduce the ETT problem to a standard MTT. To the best of our knowledge, a cascade ETT algorithm was never developed before. In particular, our approach involves a first target tracking layer that aims at enhancing the position measurements with an estimate of their speed; subsequently, measures are clustered based both on their relative position and speed; finally, a second target tracking algorithm is applied to the clustered measures, which outputs an estimate of the position and speed of the obstacles in the scene. Our technique makes no assumption on the shape of the detected obstacles, and as

10.1016/j.ifacol.2020.12.2366

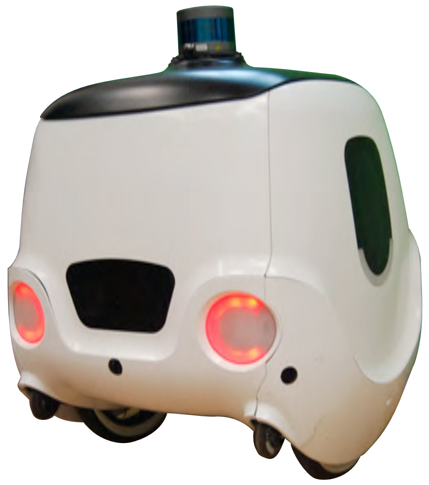

Fig. 1. YAPE: The prototype used in the experimentation

such it is capable of simultaneously tracking objects which are very different in size,such as pedestrians and cars. This technique has been implemented and experimentally tested on real-world urban scenarios using a prototype vehicle designed for autonomous last mile delivery, and provided promising results in the context of autonomous navigation on crosswalks. The outline of the rest of the paper is as follows: Section 2 describes the experimental setup, describing the vehicle mechanical structure and the sensors. Section 3 presents the algorithm structure, followed by Section 4 in which the experimental results are discussed.

2. EXPERIMENTAL SETUP

YAPE (Fig. 1) is a prototype vehicle designed for autonomous parcel delivery in urban environment (see Sabatini et al. (2018)). Structurally YAPE is a Two Wheeled Inverted Pendulum (TWIP) composed of a steel chassis and two parallel driving wheels; such structure allows turn-on-the-spot maneuvers, but it also implies a tilting motion of the chassis which is proportional to the vehicle longitudinal acceleration. In order to be able to operate autonomously, YAPE is equipped with a suite of sensors enabling it to perceive the external environment and its own motion. Our target tracking algorithm relies on a Quanergy-M8 3D Lidar mounted on the top lid. This sensor has 8 scanning layers providing 360° horizontal Field Of View (FOV) and 20° vertical FOV ($+3^\circ/-17^\circ$). Despite a declared measurement range of more than 150 [m] for each laser beam, we found 30 [m] to be the range at which the point cloud produced by the sensor was dense enough for our purposes. Particular attention has to be paid to the effects of the tilting behavior of the chassis on the sensor usable FOV. Tilting motion, which is intrinsically related to TWIP structure, can be measured and corrected but brings an inevitable reduction of the usable FOV in the longitudinal direction.

3. CASCADE TARGET TRACKING

This section describes the structure of our cascade algorithm for extended-target tracking, which develops as follows: first, a 2D grid representation of the environment is produced (Sec. 3.1) in which only dynamical obstacles are included (Sec. 3.2); extended-targets in the scene are then tracked using a three-stage algorithm composed of a low-level tracking stage (Sec. 3.3), a clustering algorithm (Sec. 3.4) and a high-level tracking stage (Sec. 3.5) providing the actual speed estimation for the obstacles in the scene.

3.1 Point cloud processing

YAPE's Lidar produces a 3D representation of the environment. However, since this application deals with vehicles navigating a two dimensional world we preferred to work with a 2D representation, which yields a reduction in the computational complexity. As such, the first step of our algorithm is to build a compact representation of the robot surroundings, in the form of a 2D binary-grid. This is done starting from the 3D point cloud produced by the Lidar sensor and running it through a set of steps.

Ground removal. Measures generated by laser beam pointing to the ground are removed from the point cloud, so that ground points are not mistaken as actual obstacles in the following steps. Since our algorithm is specifically designed to tackle crosswalks, which are generally characterized by a flat road surface, we found that a RANSAC plane fitting was enough to obtain satisfactory ground removal. Remaining ground points which are not excluded by the RANSAC fitting are treated as noise and handled by the following processing steps.

Scan registration. Since the Lidar reference frame is rigidly connected to the vehicle, objects in the point cloud are subject to an apparent motion which is opposite to that of the vehicle. In order to discriminate between static and dynamic obstacles is therefore necessary to compensate for such apparent motion, in a process that we referred to as scan registration. The rationale behind scan registration is that expressing the point clouds measurements with respect to a common, static reference frame eliminates the apparent motion, so that variations in the scene are imputable to dynamic objects only. This can be seen as a localization problem, as the sensor pose with respect to the static frame needs to be known in order to perform the change of coordinates on each point cloud. In our use case, we found a dead-reckoning estimate to be reliable enough to solve the localization process. In fact, even though odometric measures are known to be subject to slow drifting behavior due to error accumulation, in case of crosswalk navigation the estimation process is limited to a relatively small region of space (i.e. the width of the street that needs to be crossed), and to a relatively short trajectory. In other words, the robot does not travel far enough to accumulate a relevant error, and the resulting localization drift is slow enough to be considered negligible with respect to the motion of the dynamic obstacles in the scene. Our dead-reckoning localization algorithm exploits a Lidar-based odometry measure, which is mainly based on the concept of scan matching. A scan matching algorithm computes the rototranslation occurring between two Lidar acquisitions by optimizing the overlap obtained by projecting one point cloud over the other. This is done under the hypothesis that most of the scene observed by the sensor is static, so that variations between subsequent scans are mainly occurring due to apparent motion. Among all the scan matching algorithm available in the literature we chose the Coherent Point Drift (see Myronenko and Song (2010)).

Grid Quantization. The point cloud, which is now expressed in a world-fixed reference frame, can be defined as a set of k points with their 3D coordinates:

$$P = \{p_k \mid p_k \in \mathbb{R}^3 \; k = 1, ..., K\}. \tag{1}$$

The goal is that of building a grid-representation of a certain region of space. Supposing to consider squared cells of size γ, the matrix:

$$B = \{b_{n,m} \mid n = 1, ..., N \; m = 1, ..., M\}. \tag{2}$$

can be used to represent a rectangular space of $\gamma N \times \gamma M$ meters. In particular, we imposed N and M to be even numbers, and prescribed the fixed frame origin to be placed at the center of the grid (Fig. 2). Then, a possible 3D to 2D conversion corresponds to counting how many points in the point cloud are contained within a virtual column extending above each cell. This is done by assigning to $b_{n,m}$ a value corresponding to the number of points belonging to P which fulfill the following inequality:

$$n-1 \leq \frac{x_p}{\gamma} + \frac{N}{2} < n \;\wedge m-1 \leq \frac{y_p}{\gamma} + \frac{M}{2} < m. \tag{3}$$

where x_p and y_p are the x and y coordinates of the generic point expressed in the world-fixed reference frame.

Grid Binarization. Because of the fixed angular resolution, the cartesian resolution of the measurement decreases as the distance of the object from YAPE increases. It is thus expected that the point cloud becomes sparser as the distance from the sensor increases. In order to compensate for this fact, the values of the cells $b_{n,m}$ are normalized with respect to the maximum number of points $\bar{b}_{n,m}$ that each of them could potentially contain. The maximum number of points per cell was computed as:

$$\bar{b}_{n,m} = \alpha(n,m)\delta. \tag{4}$$

where δ is the angular point density of the sensor and $\alpha(n,m)$ is the angular coverage of cell $b_{n,m}$. The dependence of $\alpha(n,m)$ from the cell position and distance from the sensor is depicted in Fig.2. Notice that (4) only considers a single layer covering the grid cells; therefore the value $\bar{b}_{n,m}$ represents an underestimation of the actual maximum number of points per cell. However, since this approximation is more realistic as the distance from the sensor increases, and since the normalization step is particularly critical for peripheral cells, this assumption was considered acceptable, and proved to be valid in all experiments. Once the maximum value per each cell is known the corresponding normalized value is

$$\hat{b}_{n,m} = b_{n,m}/\bar{b}_{n,m}. \tag{5}$$

Finally, since the tracking algorithm only needs a representation in terms of occupied and free space, a binary value is computed per each cell by means of thresholding. The resulting normalized grid $\tilde{B}$ is defined by:

$$\tilde{b}_{n,m} = \begin{cases} 1 \text{ if } \hat{b}_{n,m} > \tau_{bin} \\ 0 \text{ otherwise} \end{cases}. \tag{6}$$

where τ_{bin} is a suitable threshold.

The results of the various point cloud processing steps are depicted in Fig. 3. Notice that the resulting grid is in fact a binary image in which black and white pixels correspond to free and occupied space respectively.

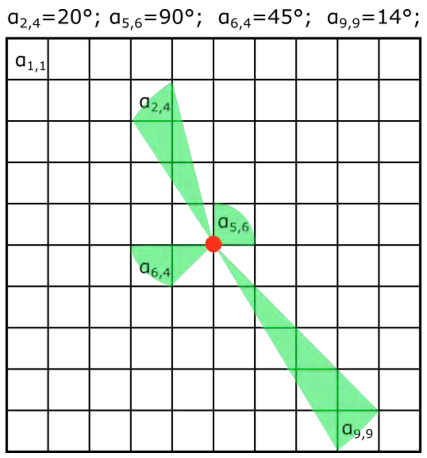

Fig. 2. Angular coverage per cell

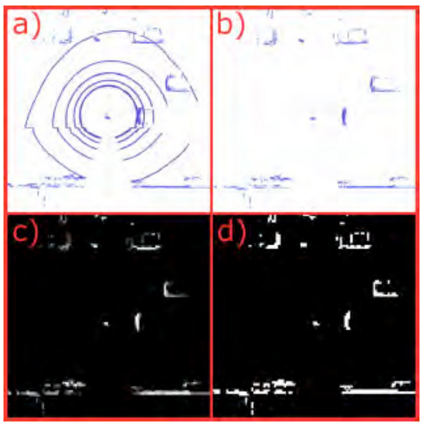

Fig. 3. Pointcloud preprocessing steps: a) original image b) background removal c) grid quantization d) grid binarization

3.2 Dynamical objects extraction

Once a binary image representing the surrounding environment is available, it is possible to segment it into background and moving obstacles. This is once again done in steps.

Background Removal. In order to separate moving obstacles from the background, the time evolution of the pixel value is considered. In particular, an estimate of the background image at each time is computed as a thresholded moving average over a window W of the previously computed frames. First, the moving average is trivially computed as

$$G(t) = \frac{\sum_{T=0}^{W} \tilde{B}(t-T)}{W}. \tag{7}$$

Then the background image $\tilde{G}$ is computed by requiring:

$$\tilde{g}_{n,m} = \begin{cases} 1 \text{ if } g_{n,m} > \tau_{bck} \\ 0 \text{ otherwise} \end{cases}. \tag{8}$$

where τ_{bck} is a suitable threshold. Notice that averaging the value and applying a threshold is equivalent to prescribe that a pixel has to be occupied for at least $\tau_{bck}W$ frames over the last W to be included in the background.

The final image F containing only dynamic pixels is obtained as:

$$f_{n,m} = \tilde{b}_{n,m} \wedge \neg \tilde{g}_{n,m}. \tag{9}$$

As shown in Fig. 4, this process removes most of the background obstacles from the scene; however, the resulting image presents some noise in the form of isolated pixels which are still marked as occupied. In order to remove them, a set of morphological operations are applied.

Image processing. The goal of the image processing step is dual: it aims at removing noisy pixels and at aggregating as much as possible those pixels which are related to the same obstacle. Both goals are achieved by applying a sequence of morphological operations (see Haralick et al. (1987)) to the BEV image. As much of the noise produced by background removal is located in a neighbor of the stationary obstacles, a dilation filter is applied to the background image before computing the subtraction. Then, a closure filter is slid on the resulting image to aggregate the sparse points belonging to a same obstacle. Finally, connected components with total area below a prescribed threshold are removed from the resulting image to further exclude faulty detections and noise. The resulting image is shown in Fig. 5

Shadow Management. As most radial range finders, Lidars are affected by a shadowing phenomenon for which each obstacle detected by the sensor creates a shadow cone hiding any other object which falls inside it. This has two main negative effects on our target tracking algorithm. First, any dynamical object can disappear and reappear in the scene as it enters and exits shadow cones: this problem and the adopted solution will be further discussed in the following section. Secondly, moving shadow cones generated by dynamical objects can produce relevant performance degradation in the background removal step. In order to understand how shadows affect background removal, let's consider a pixel that was correctly labeled as background in a certain sequence of frames. If such pixel is covered by a shadow cone for a sufficient amount of time (which is related to the values of τ_{bck} and W), it will eventually be marked as free space and excluded from $\tilde{B}$. Therefore, once the obstacle creating the shadow cone is passed, that pixel will be recognized as occupied and mistakenly labeled as dynamic until the moving average filter includes it once again in the background image. In order to cope with this problem, estimated shadow cones are created for each dynamical object detected in the scene. Then, pixels in $\tilde{B}$ which fall under any shadow cone are kept constant until they are visible again. This is done supposing that shadowed background doesn't change during the passage of the shadow cone, which proved to be an acceptable hypothesis in the testing phase.

3.3 Low Level Target Tracking

Starting from the processed BEV images, the problem of estimating the speed of incoming vehicles can be cast into an extended-target tracking framework. The term extended-target indicates the fact that one obstacle can in general produce more than one measure. In case of our BEV images, this correspond to recognize that each obstacle can produce more than one blob of white, or

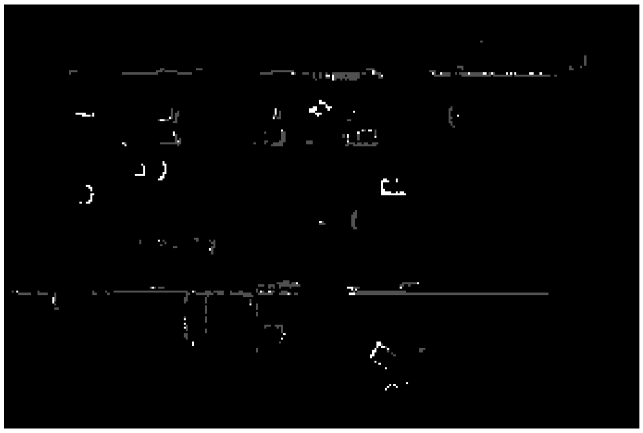
Fig. 4. BEV image after background removal: white pixels indicate dynamical objects detected in the scene. Background is reported in gray for demonstration purposes only.

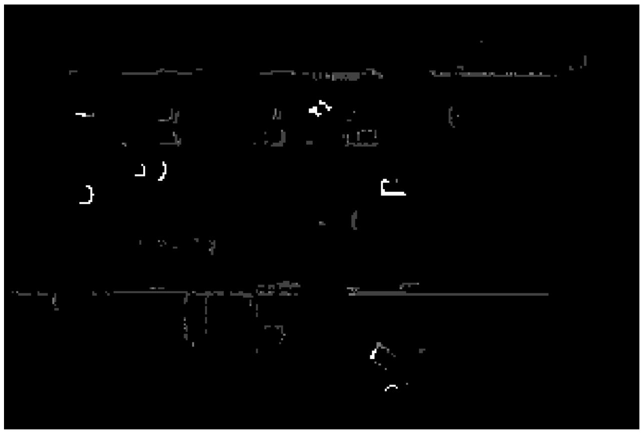
Fig. 5. BEV image after image processing.

occupied, pixels. Our solution to this problem passes trough a layered structured tracking algorithm. In the first layer, each pixel blob is considered as a separate target. The goal is that of associating each of them with a speed estimate to be used in the successive clustering layer. The measures used to track each blob are the x and y coordinates of its geometrical center. As most traditional target tracking algorithm, our first-layer target tracking is based on a specific motion model, and it is composed of a prediction step, a data association and tracks management algorithm, and a correction step.

Motion Model. The motion model for the first-layer target tracking is a Coordinated Turn (CT) model, which is based on the hypothesis of planar motion with constant rotational speed. The state vector associated to each track is:

$$\boldsymbol{x} = (x\ y\ v_x\ v_y\ \omega)^\top. \tag{10}$$

Discrete time evolution of the state is regulated by the following discrete time dynamical system:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f(\boldsymbol{x}_k) + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= H\boldsymbol{x}_k + \boldsymbol{v}_k \end{aligned} \tag{11}$$

in which $f(\boldsymbol{x})$ depends on the chosen motion model. In case of CT it can be expressed as:

$$f(\boldsymbol{x}) = \begin{pmatrix} x + \frac{v_x}{\omega}\sin(\omega T) - \frac{v_y}{\omega}(1-\cos(\omega T)) \\ y + \frac{v_x}{\omega}(1-\cos(\omega T)) + \frac{v_y}{\omega}\sin(\omega T) \\ v_x\cos(\omega T) - v_y\sin(\omega T) \\ v_x\sin(\omega T) + v_y\cos(\omega T) \\ \omega \end{pmatrix}. \quad (12)$$

The measurement matrix H is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (13)$$

Leveraging this model it is possible to set up an Extended Kalman Filter, which evolves under the usual prediction-correction scheme. However, two further steps are needed between the filter prediction and correction phase, namely a data association step and a tracking management step.

Data Association. The key concept is that at each time step, new measurements are acquired from the obstacles on the scene, but there is no prior knowledge on which new measure should be associated to each existing track. Tackling this problem requires the solution of two subtasks, namely gating and association. Gating refers to a process of selection of those measures which are potentially compatible with each track. Once those measures are selected an association problem must be solved trying to associate each track with the most likely measure. We base our gating step on the Mahalanobis distance (see Mahalanobis (1936)). At each time instant, for each measurement y and predicted track a distance measure is computed as:

$$d_M(\boldsymbol{y}, \hat{\boldsymbol{y}}) = (\boldsymbol{y} - \hat{\boldsymbol{y}})^\top \boldsymbol{S}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{y}}) \quad (14)$$

where $\boldsymbol{y}$ is the measure, $\hat{\boldsymbol{y}}$ is the predicted track position and S is the innovation covariance defined as $S = HP'H^\top + R$. A measure $\boldsymbol{y}$ can be associated to $\hat{\boldsymbol{y}}$ only if:

$$d_M(\boldsymbol{y}, \hat{\boldsymbol{y}}) \leq \tau_G. \quad (15)$$

where τ_G is the *gating threshold*. Notice that in general more than one measure can satisfy (15) for the same track, and conversely the same measure can be acceptable for more than one track. It is therefore necessary to solve a data association problem which in our case was done using the Global Nearest Neighbor (GNN) algorithm. This requires each track-measurement couple to be associated with a weight which is proportional to the Mahalanobis distance. Measures which fall outside the gating threshold are given infinite cost. The problem is then formulated as a typical combinatory optimization problem known as association problem, which is solved using Munkres algorithm (Hungarian algorithm, see Kuhn (1955)).

Tracks Management. In general, the number of active tracks at each time instant is different from the number of newly acquired measures. This difference can have different causes:

- A previously tracked object could have exited the scene, so that there is a track with no associated measure that should be eliminated from the list.
- A new object could have entered the scene, so that a new track should be created to follow it.
- A previously tracked object could now be hidden by the shadow produced by another obstacle in the scene, so that there is a track with no associated measure that should be kept alive until the tracked object exits the shadow cone.
- A faulty measure could have survived the filtering process, so that there is a measure with no associated track that should be ignored.

In any of the previous cases, the data association step will leave either measures with no track or tracks with no measure. Both these cases are managed by introducing a confirmation and an elimination logic for the tracks. First of all, a new track is initiated every time there is a measure with no associated track. However this is initially labeled as a *tentative track*, and it is confirmed only if it receives at least M_c measures in the next N_c time instants, otherwise it is eliminated. This logic, which is known as *M/N logic*, is also applied to track elimination, so that a track is removed from the list if it doesn't receive at least M_e measures in the last N_e instants.

3.4 Clustering

In order to group the targets produced by the low level target tracking we developed a threshold based clustering algorithm in which two distance metrics are used. In particular, if we define a cluster C as a set of tracks q each characterized by a state vector as in (10), the first distance measure accounts for position, and is defined as:

$$\delta_d(p, C) = \min_{q \in C} \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}. \quad (16)$$

Conversely, the second distance measure is computed in the velocity domain according to:

$$\delta_v(p, C) = \max_{q \in C} \sqrt{(v_{x_p} - v_{x_q})^2 + (v_{y_p} - v_{y_q})^2}. \quad (17)$$

The clustering algorithm then develops as follows:

(1) Randomly select a track from those generated by the low lever target tracking algorithm. This is the seed of the first cluster C_1.
(2) For every unclustered track p compute $\delta_d(p, C_1)$ and $\delta_v(p, C_1)$. The track is included in C_1 if both $\delta_d(p, C)$ and $\delta_v(p, C)$ are below two prescribed thresholds
(3) If there are still unclustered track, randomly select one to be the seed of the next cluster, then repeat step 2 for the new cluster.
(4) The algorithm ends when all tracks have been assigned to a cluster.

Using a threshold on the speed allows to deal with cases in which two objects are very close to each other in the scene (which happens very often on two-lanes roads), so that a pure position based clustering would lead to mistakenly aggregate blobs which are not produced by a single obstacle. Notice that the minimum distance is considered for track position in order to make the inclusion criterion independent from the object dimension, whereas the maximum distance is considered for track speed in order to obtain a bounded speed variation within the same cluster.

3.5 High Level Target Tracking

The goal of the clustering algorithm is to create exactly one cluster for each obstacle in the scene. Moreover, by averaging the position and speed of each track belonging to a cluster, an averaged state vector of the same form of (10) can be associated to each cluster. In this way, the original extended target tracking problem can been reduced to a standard target tracking one, which considers the averaged state vectors as measures. An additional target tracking layer is therefore implemented, on the same line of the first-layer target tracking. In particular, the same CT motion model is used, together with the EKF structure. The only difference lies in the measurement matrix, which is defined as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{18}$$

where cluster average x and y position and speeds are considered. The data association step is the exact equivalent of that implemented in the low level target tracking, based on Mahalanobis distance and GNN association. Finally, concerning tracks management, no confirmation logic is implemented, whereas the usual M/N logic is used for cluster elimination. The rationale is that only confirmed tracks are included in the clustering procedure, therefore adding a further confirmation step would only increase the detection delay without adding anything to the noise reduction. On the other hand, cluster measures could be subject to shadowing or misdetection, therefore an elimination logic is required to keep the cluster alive in those cases.

4. EXPERIMENTAL RESULTS

The algorithm described in the previous section was experimentally validated with field tests in Milan. The robot was manually operated in proximity of urban crosswalks and Lidar data were acquired and processed. During the operations the robot was interacting with a variety of different agents, including pedestrian, cyclist, cars, trucks and buses. This allowed us to assess the performance of the algorithm in presence of large variations in the obstacles size and speed. As expected, size plays a role in determining the distance at which an obstacle produces at least one measurement point: with our Lidar sensor we found the first detection distance for a car to be of approximately 30 meters with unobstructed view. The selected scenario was actually particularly challenging in this respect due to the presence of parked cars which which were partially obstructing the sensor field of view at the beginning and at the end of the crosswalk. Since the experiments took place in a real world scenario, no ground-truth measurement was available for the obstacle trajectories. The overall performance of the algorithm was therefore assessed using estimated ground-truth trajectories which were obtained in post processing by manually selecting the Lidar points belonging to each vehicle in each frame. Fig. 6 shows a visual comparison between the estimated reference and the trajectories obtained by our tracking algorithm, highlighting the good performance of the algorithm in tracking both vehicles proceeding on a straight line and vehicles performing turns. In particular, Fig. 6 reports the results obtained in tracking a car, a bicycle and a pedestrian respectively, proving the capability of our algorithm to deal obstacles with relevant differences in size and speed. Finally, a comparison between the reference and estimated speed of one of the vehicles is reported in Fig. 7, once again showing the effectiveness of the filter.

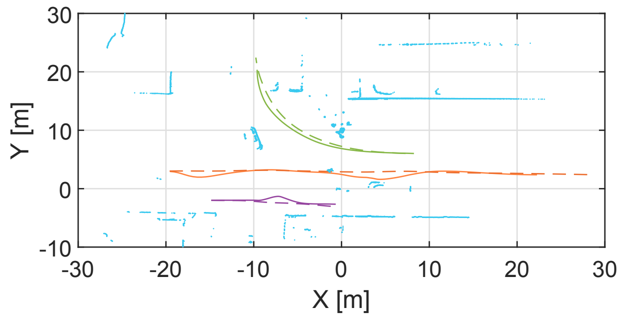

Fig. 6. Ground truth (dashed line) and estimated trajectories (solid line) for a car (green) a bicycle (orange) and a pedestrian (purple).

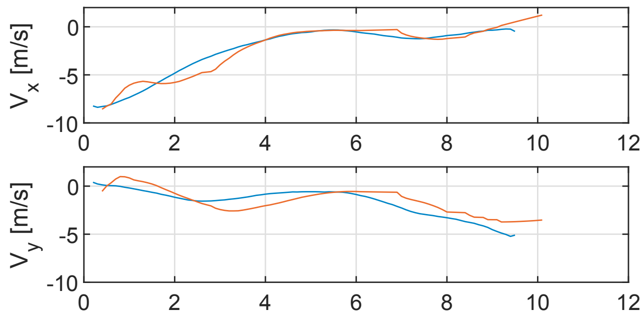

Fig. 7. Speed reference (red) and EKF estimate (blue) for vehicle number 3

REFERENCES

Bar-Shalom, Y. and Osborne, R.W. (2015). Data association. *Encyclopedia of Systems and Control*, 251–259.

Granstrom, K., Baum, M., and Reuter, S. (2016). Extended object tracking: Introduction, overview and applications. *arXiv preprint arXiv:1604.00970*.

Haralick, R.M., Sternberg, S.R., and Zhuang, X. (1987). Image analysis using mathematical morphology. *IEEE transactions on pattern analysis and machine intelligence*, (4), 532–550.

Kuhn, H.W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2), 83–97.

Li, X.R. and Jilkov, V.P. (2003). Survey of maneuvering target tracking. part i. dynamic models. *IEEE Transactions on aerospace and electronic systems*, 39(4), 1333–1364.

Mahalanobis, P.C. (1936). On the generalized distance in statistics. National Institute of Science of India.

Myronenko, A. and Song, X. (2010). Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12), 2262–2275.

Sabatini, S., Corno, M., Fiorenti, S., and Savaresi, S.M. (2018). Improving occupancy grid mapping via dithering for a mobile robot equipped with solid-state lidar sensors. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, 1145–1150. IEEE.

Vo, B.n., Mallick, M., Bar-shalom, Y., Coraluppi, S., Osborne III, R., Mahler, R., and Vo, B.t. (1999). Multitarget tracking. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 1–15.