2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Leveraging Stereo-Camera Data for Real-Time Dynamic Obstacle Detection and Tracking

Thomas Eppenberger[⋆†], Gianluca Cesari[⋆], Marcin Dymczyk[⋆], Roland Siegwart[†], and Renaud Dubé[⋆]

Abstract— **Dynamic obstacle avoidance is one crucial component for compliant navigation in crowded environments. In this paper we present a system for accurate and reliable detection and tracking of dynamic objects using noisy point cloud data generated by stereo cameras. Our solution is real-time capable and specifically designed for the deployment on computationally-constrained unmanned ground vehicles. The proposed approach identifies individual objects in the robot's surroundings and classifies them as either static or dynamic. The dynamic objects are labeled as either a person or a generic dynamic object. We then estimate their velocities to generate a 2D occupancy grid that is suitable for performing obstacle avoidance. We evaluate the system in indoor and outdoor scenarios and achieve real-time performance on a consumer-grade computer. On our test-dataset, we reach a MOTP of $0.07 \pm 0.07\mathrm{m}$, and a MOTA of 85.3% for the detection and tracking of dynamic objects. We reach a precision of 96.9% for the detection of static objects.**

I. INTRODUCTION

In order to safely and efficiently navigate in crowded public spaces, Unmanned Ground Vehicles (UGVs) need to reason about their static and dynamic surroundings and predict the occupancy of space to reliably perform obstacle avoidance [15]. Generally, static objects can be avoided by small safety distances, whereas for compliant navigation, dynamic objects need to be avoided by larger distances [45]. Moreover, robots should avoid crossing pedestrian paths, which not only requires to correctly classify dynamic objects as such, but also calls for accurate motion estimation and prediction. In addition to humans, other dynamic objects with varying size and speed, such as animals or other vehicles, may appear in the surroundings. Hence, the detection may not be restricted to humans only but needs to generalize.

We identified two major families of state-of-the-art techniques for detecting and tracking objects in crowded scenes. The first group uses point cloud data, often generated from highly-accurate LiDAR sensors, which allows for the detection of generic dynamic objects and is typically used for autonomous driving [1, 25]. The costs of LiDAR sensors make most of the algorithms in this first group not applicable for small, commercially used UGVs, such as delivery robots for hospitals or airports. The second group uses visual information from images and primarily focuses on the detection of a predetermined number of object classes, such as pedestrians and vehicles [22, 47]. However, these approaches often lack the ability to detect generic dynamic objects and do not run in real-time on computationally-constrained platforms [59]. In order to successfully deploy UGVs at large scale, low-cost sensor setups, such as stereo cameras, should be used along with efficient algorithms.

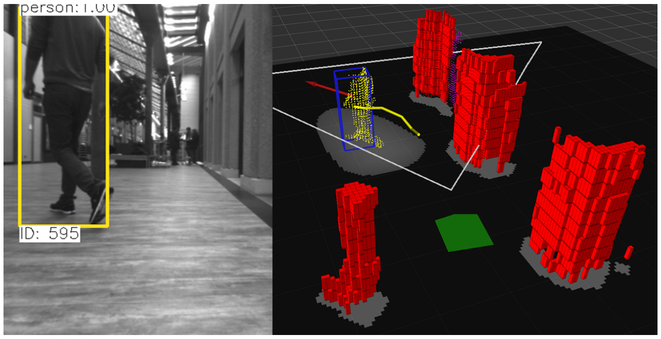

Fig. 1. Visualization of the output of the proposed dynamic object detection and tracking approach. **Left:** the input camera image overlaid with the output of a visual people detector, indicating the confidence of the detection and a tracking ID. **Right:** a resulting occupancy grid, with correctly identified static objects (red voxels) and a detected pedestrian (the yellow point cloud). The red arrow visualizes the pedestrian's estimated velocity, the yellow track shows the past trajectory, and the blue cuboid indicates that the visual people detector recognized this cluster as a person. Gray floor areas indicate high costs in the occupancy grid. The robot's footprint and field of view are shown in green and white, respectively.

We introduce a solution that leverages stereo camera data to reliably and accurately detect and track dynamic objects. To this end we first present a novel algorithm to detect generic dynamic objects based on their motion. For enhanced perceptual performance in crowded spaces, we use a visual people detector to classify humans motion-independently as a specific class of dynamic objects, as depicted in Figure 1. Our approach handles short-time occlusions using the estimated velocity of the dynamic objects. To the best of our knowledge this is the first work to propose a complete solution that uses stereo cameras for detecting and tracking generic dynamic objects by combining global nearest neighbor searches and a visual people detector. The system relies on noisy data of one stereo camera only and is designed to run on computationally-constrained platforms. As shown in the work of Liu [31], our perception system has been used for navigating an UGV in real life crowds. We encourage the reader to consult the supplementary video (`https://youtu.be/AYjgeaQR8uQ`) for more visualizations.

The contributions of this paper are as follows:

- a novel real-time algorithm to detect and track generic dynamic objects based on noisy stereo camera-data;
- a method to combine the aforementioned algorithm with a vision-based people detector to improve the detection and tracking performance and handle

[⋆] Sevensense Robotics AG, Zurich, Switzerland {firstname.lastname@sevensense.ch}

[†] ETH Zurich, Autonomous Systems Lab (ASL), Zurich, Switzerland

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

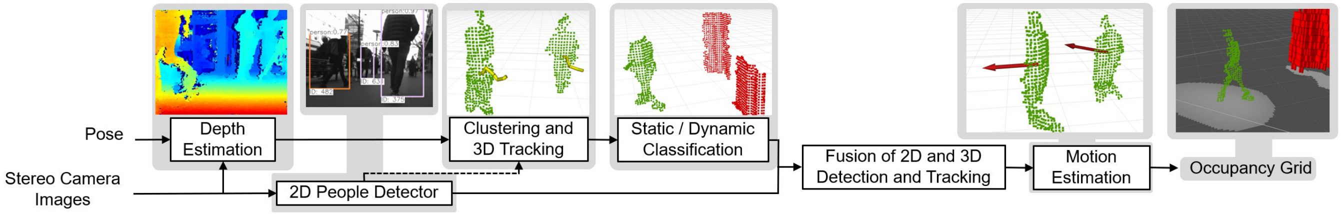
Pose
Stereo Camera Images
Depth Estimation
2D People Detector
Clustering and 3D Tracking
Static / Dynamic Classification
Fusion of 2D and 3D Detection and Tracking
Motion Estimation
Occupancy Grid

Fig. 2. Overview of our pipeline: the inputs are stereo images and the estimated pose of the robot from a visual SLAM module. The output is a 2D occupancy grid, which enables planning paths close to static objects and ensures avoidance of dynamic objects by a safety distance.

short-time occlusions;

- an evaluation of our pipeline on challenging datasets, demonstrating its performance and reliability for increased mobile robot safety.

II. Related Work

The task of detecting and tracking dynamic agents has led to a variety of different approaches, as robotic platforms range from small, commercial UGVs to autonomous cars. These approaches differ in computational load, robustness against noise, and required sensor data, namely point clouds, images, or a combination of both.

Image based algorithms: Assuming that a feature-based SLAM system is used, one approach is to utilize outliers of the feature tracking module to detect dynamic objects [4, 56]. Song et al. [52] argue that this approach should be favored over the usage of optical flow to estimate the velocities of visible objects [43]. This technique, however, requires the tracking algorithm to use dense feature points to ensure outliers point to all dynamic objects. This is not guaranteed for many visual SLAM systems and contradicts with our goal to design a generic dynamic object tracking pipeline.

Nonetheless, the optical flow methods can be enhanced to produce so-called Scene Flow, which computes per-pixel 3D velocities. However, this method does not run in real-time on computationally-constrained platforms [8, 48].

Another approach is the specific detection of pedestrians using visual data [12, 15, 39, 59], where deep neural networks receive much attention [47, 51, 58]. Segmentation networks [42, 50] are an alternative to detector networks, but do not imply object-instances, and hence, are impractical for differentiation between multiple people in crowded scenes. Object-instance segmentation networks overcome this drawback [18]. In this work, we do not limit ourselves to detecting certain object classes only, but pursue a generic dynamic object detection.

Point cloud based algorithms: detection algorithms based on Iterative Closest Point (ICP) match current segments of a point cloud to a previous point cloud to reveal their motion [11, 23, 30]. This approach works best for rigid objects like cars and considerably decreases in performance for deforming objects like humans. This property does not match the needs of our system, where we put a special emphasis on human detection and tracking.

When using a volumetric occupancy grid, the classification of points as static or dynamic can be based on the consistency of the occupancy of the voxels [1, 3, 7]. The performance of this approach depends on the voxel-size where smaller voxels allow for a more precise occupancy analysis, while requiring more computational effort. A probabilistic metric for the occupancy of the voxels is necessary to handle noise in the data, introducing a time-lag to the classification process.

Moreover, there are several algorithms designed for autonomous driving that rely on high-quality point clouds from LiDAR sensors and powerful GPUs to detect cars and pedestrians [25, 26, 37]. In contrast, our target platforms are small UGVs with low computational power and potentially the lack of an expensive LiDAR sensor.

Our approach directly compares point clouds among frames and is related to the work of Yoon et al. [57] and Shi et al. [49]. Yoon et al. [57] work with LiDAR data and need to introduce a time-lag of one frame to reliably cope with occlusions. We expand their idea to handle noisy stereo camera data with potentially incomplete depth information and additionally introduce a novel approach to differentiate between dynamic and previously occluded points without the need of a time-lag of one frame. Shi et al. [49] use RGB-D data and remove points during dense reconstruction in case they are spatially inconsistent between frames. In contrast, our method is able to classify all points of an object as dynamic, even though only their subset shows spatial inconsistency. In consequence, we obtain a more complete and robust classification of dynamic objects. Compared to the work of Osep et al. [41] we do not limit our system to a set of predefined detectable classes but implement a generic dynamic object detector instead.

III. Methodology

An overview of the proposed stereo camera-based perception approach is given in Figure 2 and the remainder of this section details its individual modules. To associate camera-based point clouds with the global frame we localize the robot using precise Visual Inertial Odometry (VIO).

A. Point Cloud Generation

The first module generates a 3D point cloud from undistorted and rectified stereo images. We designed our approach to be generic regarding the inputs, hence any algorithm extracting a disparity map from stereo images can be used in this module, from which we consider the well-established block-matching and deep neural networks.

1) Block-Matching: We use semi-global block-matching [19] and apply a weighted-least-squares filter [38] on the resulting disparity map.

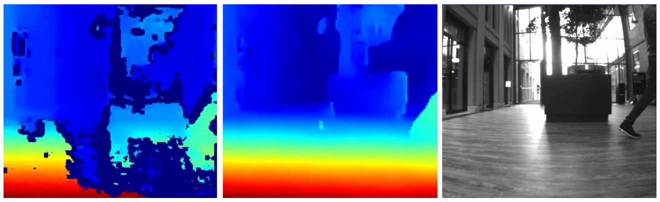

Fig. 3. Depth representations generated using stereo images. **Left:** Block-matching [19] cannot generate depth information in parts of the low-textured object on the right, or on the shiny surface of the floor. **Middle:** MADNet [53] captures most parts of the object and the floor. Hence, it delivers more complete depth information than block-matching in this scenario. **Right:** raw image.

2) Deep Stereo: Recently, deep neural networks that learn to infer disparity values from stereo images have emerged [9, 24, 35, 54]. We use MADNet [53], as we found this network to deliver a suitable trade-off between run-time and performance. Figure 3 shows an exemplary disparity map generated by both methods.

B. Point Cloud Filtering

We filter the point cloud generated by the previous module to reduce noise and down-sample the data for achieving real-time performance. We denote the point cloud after initial cropping as h^d and after filtering as h^s. We applying the following sequence:

- crop the point cloud at the depth limit l_d, up to which the measurements can be trusted[1];
- crop the point cloud at the heights l_g and l_h, to remove all ground plane and ceiling points, respectively.
- apply a voxel filter with leaf size l_l to reduce the size of the point cloud by an order of magnitude and to ensure even density of the points in 3D.
- apply a filter to remove all points with less than l_n neighbors within a radius l_r, to reduce noise points which occur most notably at the edges of objects.

C. Clustering and 3D Tracking

In this module we identify the individual objects through clustering and track them from frame to frame.

1) Clustering: We use DBSCAN [16] to cluster the point cloud, resulting in a set of m clusters $C = \{C^1, C^2, ..., C^m\}$. DBSCAN grows clusters from random seeds using dense points only. Compared to Euclidean clustering [13], we experienced that DBSCAN more precisely separates individual objects in cluttered point clouds, while introducing only a marginal computational overhead.

To refine the clusters, we use the bounding-boxes generated by the 2D people detector module. We separate any clusters which are associated with more than one bounding-box to distinguish individual humans. We also separate clusters whose fraction of points laying within the bounding-box is below a threshold to distinguish between humans and nearby static objects. In Section III-E we describe how the bounding-boxes are obtained and associated to the clusters.

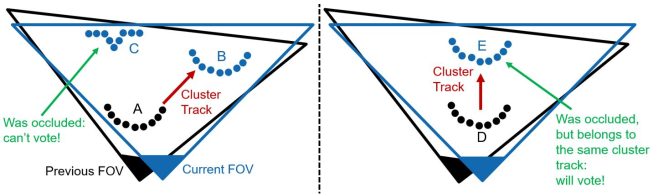
C
B
A
Cluster Track
Was occluded: can't vote!
Previous FOV
Current FOV
E
Cluster Track
D
Was occluded, but belongs to the same cluster track: will vote!

Fig. 4. Occlusion handling at dynamic object detection. **Left:** the current cluster C is excluded from voting, as it was occluded in the previous frame by cluster A, which belongs to a different cluster track $T^B_{t,1}$. **Right:** the current cluster E is *not* excluded from voting, as in the previous frame it was occluded by cluster D, which belongs to the same cluster track $T^E_{t,1}$.

2) 3D tracking: First, at time t we compute the centroids c_t of all current clusters C_t in the global frame as the average of all their points. Then, we associate them to their closest centroid $c^*_{t-\Delta t}$ of the clusters $C_{t-\Delta t}$ of the previous frame. Applying the tracking over k frames separated by Δt leads to a cluster track $T^i_{t,k} = \{c^*_{t-k\cdot\Delta t}, ..., c^*_{t-\Delta t}, c^i_t\}$ for a current cluster C^i_t. We mark current centroids c_t that cannot be related to any previous centroids $c^*_{t-\Delta t}$ as newly appeared objects and non-connected previous centroids as lost objects.

D. Classification as Dynamic or Static

In this module the clusters C^i_t identified in Section III-C are classified as either static or dynamic based on a voting strategy. First, we let the individual points of a cluster vote for the cluster's class. Then, the classification of the cluster is based on the voting information of all its points. Namely, we classify it as dynamic if the absolute or relative amount of votes for being dynamic surpass the respective thresholds l^{abs}_{dyn} or l^{rel}_{dyn}. We use two thresholds in order to correctly classify objects at different scales. In case the classification is inconsistent over a short time horizon τ, we mark the cluster as uncertain. Every cluster classified as dynamic is regarded as being an individual object.

Below, we describe the voting process of an individual point and indicate which points are excluded from voting.

1) Voting of an individual point: We measure the global nearest neighbor distance d^k from each point k of a cluster C^i_t in the current *filtered* point cloud h^s to a previous *dense, non-filtered* point cloud $h^d_{t-\delta}$. We found that it is key to measure d^k from the *filtered* to the *dense, non-filtered* point cloud $h^d_{t-\delta}$ in order to gain robustness against noise.

In order to assure that points of dynamic objects move substantially more than points corrupted by noise, we compare point clouds from frames being roughly δ seconds apart, similarly to the work of Yoon et al. [57]. For static objects, these measured nearest neighbor distances d will be in the magnitude of the noise. For points at the leading edge of moving objects, however, d will be substantially higher. We convert these distances to velocities and let each point k with $\frac{d^k}{\delta} < l_{NN}$ vote as being static, while points with $\frac{d^k}{\delta} \geq l_{NN}$ will vote as being dynamic. l_{NN} denotes the velocity threshold, which needs to be set marginally higher than the noise level in the filtered point cloud h^s.

2) Excluding points from voting: We only can infer knowledge about a point's movement if we could observe it in both frames used for voting. This observability-

[1] The limit l_d must be chosen based on the camera setup and the algorithm generating the disparity map.

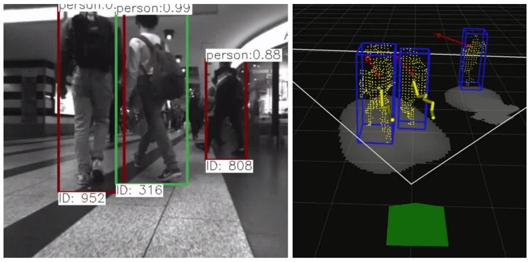

Fig. 5. **Left**: a sample output of the Mobilenet-SSD people detector [21, 32]. Every detection is rated by a confidence, shown on top. We assign an ID to every bounding-box being tracked on the image plane. **Right**: we associate detections to clusters, indicated by the blue cuboids.

requirement is not fulfilled in two cases which we detect for improving the voting performance.

First, if the Field of View (FOV) of a robot changes between the two frames, points might appear in the area of the current FOV that does not overlap with the previous FOV. As we have observed those points only once, we exclude them from voting.

Second, we exclude points of a current cluster C_t^i from voting if they previously were occluded by a *different* object j, i.e. by points from a cluster $C_{t-\delta}^j$ from a different cluster track $T_{t,k}^{j \neq i}$. Specifically, we distinguish between such occlusions and self-occlusions which happen when objects move away from the camera, as visualized in Figure 4. Occlusions are identified by first approximating the depth map of the previous frame $g_{t-\delta}$ by projecting randomly-sampled points of the non-filtered, dense point cloud $h_{t-\delta}^d$ onto the previous image plane. We then project a query point $q_t \in C_t^i$ onto $g_{t-\delta}$ and run a 2D nearest neighbor search. If a close nearest neighbor $n_{t-\delta}^{2D} \in g_{t-\delta}$ is found, we check for a potential occlusion, that is $depth[n_{t-\delta}] < depth[q_t]$, with $n_{t-\delta}$ being the associated 3D point of $n_{t-\delta}^{2D}$. To identify self-occlusions we associate $n_{t-\delta}$ to the cluster track of its nearest neighbor in $h_{t-\delta}^s$ and check if $T_{t-\delta}^{n_{t-\delta}} \in T_t^{q_t}$. In this case, the query point q_t is *not* excluded from voting.

E. 2D People Detector

Up to now, the system would classify a standing person as static and would only realize that it was a dynamic object once she starts walking. By adding Mobilenet-SSD [21, 32] as a visual 2D people detector to our pipeline, we achieve a motion-independent detection of pedestrians. We select this network as it delivers a suitable trade-off between run-time and performance on grayscale images.

Figure 5 shows an exemplary output of the Mobilenet-SSD people detector. We track the bounding-boxes over frames using Intersection over Union (IoU) as a metric. We use tracking by detection, as visual trackers have not yet reached a satisfying performance level while running in real-time on CPUs [28]. Similar to the cluster tracks in 3D, we generate bounding-box tracks $B_{t,k}$ in the image plane. In the subsequent Section III-F, we use $B_{t,k}$ to make the 3D tracking more robust.

The 2D bounding-boxes are associated to the 3D clusters C by linking the detection to the cluster having the highest amount of points within the bounding-box. Note that other approaches exist for this 2D-3D association [26, 27, 46, 49].

F. Fusion of 2D and 3D Detection and Tracking

The inputs to this module are the 3D cluster tracks T_t classified as static or dynamic, and the 2D bounding-box tracks B_t.

For every cluster track T_t^i, the frequency f_a^i of associated 2D-to-3D detections is computed. If f_a^i is above a certain confidence-threshold γ, we classify all clusters being added to this cluster-track as representing a pedestrian, and hence, representing a dynamic object regardless of their initial classification. Furthermore, this module checks if all bounding-boxes of a track B_t^j are consistently associated to clusters of the same cluster track T_t^i, and resets f_a^i otherwise.

G. Motion Estimation

Estimating the velocity and predicting the future trajectory of pedestrians is an active research field [10, 44, 45, 55]. Similarly to the work of Azim et al. [2], we adopt a conservative motion model to estimate the velocities and short-term future paths of dynamic objects. Assuming that the dynamic objects move on a horizontal plane, we estimate their velocity by using a constant 2D velocity model, based on a Kalman filter (KF). The measurement inputs $\vec{z}_i$ for the KF are the cluster centroids $\vec{c}_i = [c_x, c_y, c_z]_i^\top$ of a cluster track T_t^i measured in the x-y plane of the world frame: $\vec{z}_i = [c_x, c_y]_i^\top$. We define the state vector as: $\vec{x}_i = [x, y, \dot{x}, \dot{y}]_i^\top$. The system dynamics and measurement model are defined as:

$$\vec{x}_i[k+1] = A[k] \cdot \vec{x}_i[k] + N(0, Q)$$
$$\vec{z}_i[k] = H[k] \cdot \vec{x}_i[k] + N(0, R)$$

where Q and R model the system noise and measurement noise, respectively. H extracts the first two dimensions of $\vec{x}$ and A is defined as:

$$A = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where T_s denotes the time between two updates. Using the KF, we can catch short-time occlusions of dynamic objects. Specifically, when an object i is lost during tracking, we keep the KF running and compute for all new appearing objects j the probability $p(j = i)$ of being the same object as the lost one. We compute this probability as $p(j = i) = N(\vec{c}_j | \vec{c}_i, C_i(\vec{x}_i))$, with $C_i(\vec{x}_i)$ being the estimated covariance of the state $\vec{x}_i$. We connect the cluster tracks of object i and j if $p(j = i)$ surpasses a threshold.

H. 2D Occupancy grid

In order to allow for path planning and obstacle avoidance, we leverage occupancy grid representations [17, 20, 40] and specifically use the computationally efficient Costmap_2d implementation [33]. We create three maps in parallel for static, dynamic and uncertain objects, in which the obstacles are represented as positive costs at their respective location.

Fig. 6. Sample images used in our evaluation and the platform we used to collect them. Our system relies on the stereo camera mounted in the front, whereas the LiDAR was used for evaluation purposes only.

In the uncertain map, we create only short-living costs for which no static/dynamic classification was done yet. In the static map, we use raytracing [34] to clear free space once it was erroneously occupied. In the dynamic map, we expand the costs in the direction of the estimated velocity of the objects, such that a robot will not collide with it in the future. Safe paths can then be planned and executed by aggregating the costs of the three aforementioned layers.

IV. Evaluation

In this section, we evaluate the performance of the presented obstacle classification and tracking solution. The runtimes of the individual modules are finally presented.

A. Experimental Setup

We recorded multiple stereo-vision datasets that include pedestrians, featuring challenging indoor and outdoor scenes, shiny surfaces, low-textured objects, illumination changes, and empty as well as crowded spaces. To record our datasets, we used a Clearpath Robotics Jackal robot equipped with a stereo camera (grayscale, 752×480px), and an Ouster OS-1 LiDAR with 16 channels for the ground truth measurements used in Section IV-B and IV-C. Sample images of our datasets and our robot are shown in Figure 6. For all experiments, including the timing presented in Section IV-D, we run our pipeline on a Intel i7-8650U processor.

For our stereo camera setup we set the parameters introduced in Section III-B as $l_l = 0.05m$, $l_g = 0.15m$, $l_h = 1.8m$, $l_n = 30$, $l_r = 0.5m$ and $\delta = 0.4s$. Furthermore, we set $l_d = 5m$, as we chose a disparity-error of 1px leading to a depth-error of $0.5m$ to be the limit we can accept. Regarding the parameters of Section III-C, III-D, and III-F we set $l_{dyn}^{abs} = 100$, $l_{dyn}^{rel} = 0.8$, $\tau = 0.4s$, $l_{NN} = 0.45m/s$, and $\gamma = 1.5s^{-1}$. The robot's nominal speed is $1.2m/s$. We verified that our settings are sufficient for the objects to be reliably tracked up to the speed of a jogging person. Note that these parameters were identified using datasets independent of those used in the evaluations.

For precise localization of our robot we run a VIO algorithm similar to OKVIS [29]. For image processing we use OpenCV [6]. The people detector Mobilenet-SSD [21, 32] is implemented in Caffe and the deep stereo network MADNet [53] in Tensorflow.

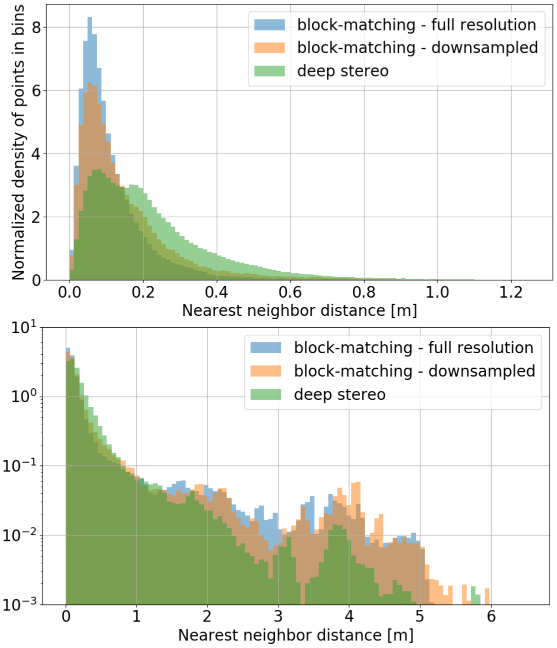

Fig. 7. Normalized histograms of nearest neighbor distances d between point clouds from the LiDAR and the stereo camera to analyze the *accuracy* and *completeness*. **Top/Accuracy:** d measured from the camera to the LiDAR. **Bottom/Completeness:** d measured from the LiDAR to the camera. The region above the accuracy-limit $l_c = 0.8m$ indicates points of objects not captured by the stereo camera point clouds.

B. Accuracy and Completeness of the Point Clouds

In this section, we briefly evaluate the quality of the stereo camera point cloud to identify the accuracy-limit l_c, which we will use in Section IV-C to evaluate the static object detection precision.

In order to evaluate the performance of the two point cloud generation methods identified in Section III-A, we collected ground truth 3D LiDAR data synchronized with the stereo images. To compare both the LiDAR and vision point clouds, we use nearest neighbor distances d as the metric, that is $d = ||p_1 - p_2||_2$, for two points p_1 and p_2. The *accuracy* of our stereo camera point clouds is estimated by computing d from each point of the camera clouds to its nearest neighbor in the LiDAR clouds. In the opposite manner, we measure the *completeness* of our camera clouds by computing d from each point of the LiDAR clouds to its nearest neighbor in the camera clouds. Note that, as the LiDAR clouds do not feature any measurements between beams, there will be non-zero nearest neighbor distances d, even in an ideal setting. In our setup[2], this distance is at most $0.08m$.

We evaluate the accuracy and completeness of the camera point cloud on a dataset where we drove down a crowded public sidewalk. In order to achieve real-time capability, we simplify the block-matching process by down-sampling images with a factor of two.

1) Accuracy: The histogram depicted in Figure 7 shows the normalized distribution of the distances d. The block-matching point cloud performs best and has an accuracy-limit of $l_c = 0.8m$, as all point are below this value. Clearly, the

[2] A 16-channels LiDAR with an opening angle of 30° and $l_d = 5m$.

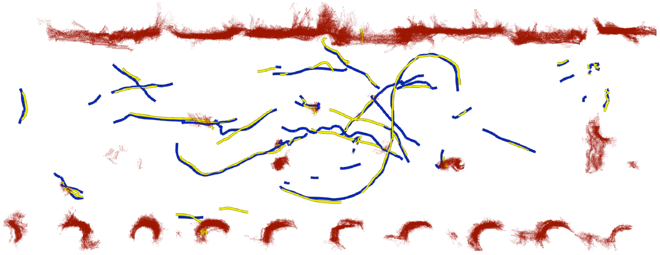

Fig. 8. The tracks of dynamic objects (classified by the module presented in Section III-D) are shown in yellow, the tracks of our ground truth based on LiDAR data in blue, and the stereo camera point cloud of static objects in red. The tracks are smooth and hence, suggest the applicability of our solution to motion tracking and prediction. Short tracks are caused by the obstacles leaving the limited FOV. The hall is of size 30 × $10m$.

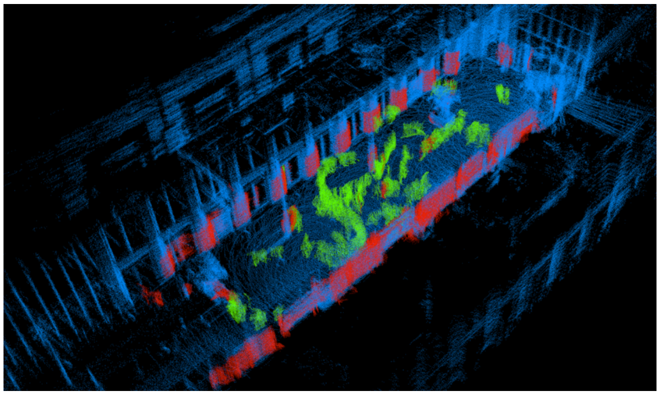

Fig. 9. Our static LiDAR ground truth is shown in blue and the accumulated stereo camera point cloud of static objects in red. For better visualization, we also show the accumulated dynamic camera point cloud in green and the LiDAR ground plane, which we remove for the comparison between clouds.

deep stereo network cannot compete with block-matching. Please refer to MADNet [53] for in-depth analysis of the performance of this network.

2) Completeness: The histogram of the normalized distribution of the distances d is shown in Figure 7. Measurements substantially larger than $l_c = 0.8m$ indicate points of objects that were not captured by the camera cloud. Deep stereo performs best, which was suggested by Figure 3. Overall, we observe that deep stereo is less exact, but more complete than block-matching. More detailed comparisons of methods for generating dense maps from stereo data can be found in the work of Menze [36]. For the remaining parts of the evaluation, we use block-matching on downsampled images, due to its real-time capability and accuracy.

C. Classification and Tracking Accuracy

To evaluate the classification and tracking accuracy of dynamic objects of the proposed system we compare the resulting object tracks to manually labeled tracks from LiDAR measurements that serve as our ground truth. We use the metrics MOTP and MOTA, as defined in the work of Bernardin [5].

For a robotic system navigating in real environments, a precise mapping of static objects matters. As MOTP and MOTA do not evaluate the detection of static objects we evaluate the similarity between the stereo-based and ground truth static clouds.

1) Classification and tracking of dynamic objects: We first generated a map m_l of a controlled, completely static environment D_1, using the measurements of the 3D LiDAR and an ICP-based algorithm similar to the work of Dubé [14]. Subsequently, we localized our robot within m_l and recorded a second dataset D_2, this time with dynamic objects present. We created a ground truth of obstacle tracks by first manually excluding from D_2 all LiDAR measurements that belong to static objects. We then extracted the upper bodies of the pedestrians in the remaining LiDAR clouds by cropping by height, in order to remove the influence of moving legs and hence, to attain smooth trajectories. Subsequently, we applied Euclidean clustering and tracked the clusters C^{lidar} using closest centroids, in the same manner as presented in Section III-C. We visually inspected and adjusted the resulting LiDAR ground truth tracks $T^{i,lidar}_{t,k} = \{c^{lidar}_{t-k\cdot\Delta t}, ..., c^{lidar}_{t-\Delta t}, c^{i,lidar}_{t}\}$ to ensure the absence of false positives, false negatives, or mismatches. D_2 is of 4min length and features 31 encounters with pedestrians leading to 1267 ground truth pedestrian positions c^{lidar}. Figure 8 shows a top-down view of our LiDAR-based ground truth tracks $T^{i,lidar}$ and the camera-based dynamic object tracks T^j extracted from D_2 by our proposed system. To compute the MOTP and MOTA, we set the threshold l_T for a correct match between object (object from the ground truth LiDAR track) and hypothesis (object from the camera track) as $l_T = 0.4m$, assuming a diameter of $0.4m$ for an average person and hence, an incorrect match if the object and the hypothesis do not have any overlap. Using this, we reach a MOTP of $0.07 \pm 0.07m$ and a MOTA of 85.3%, which is composed of a false negatives rate $f_n = 8.3\%$ (covering non-detected dynamic objects and dynamic objects erroneously classified as static or uncertain), a false positives rate $f_p = 3.0\%$ (covering ghost objects or static objects misclassified as dynamic), and a mismatch rate $f_m = 3.3\%$.

2) Classification of static objects: The map m_l from D_1 served as our ground truth for static objects. We then accumulated all stereo camera clusters classified as static in D_2 resulting in the point cloud m_s. Figure 9 visualizes m_l overlaid with m_s. To evaluate the similarity between the stereo-based and ground truth static clouds we measure the nearest neighbor distances d from points of the camera cloud m_s to the LiDAR cloud m_l. Ideally, the static camera cloud m_s coincides with the LiDAR ground truth m_l. Figure 10 shows the normalized distribution of these distances d. Ideally, d is zero for static objects. However, as shown in Section IV-B, in the extreme case static points can differ up to $l_c = 0.8m$ from the static ground truth. As there is no ground truth available for the per-point-classification, we estimate correct static classifications of our pipeline using one threshold l_e. We chose this threshold l_e to be the average of both limit cases of $l_c = 0.8m$ and zero, hence $l_e = 0.4$. We declare all static points below l_e of the camera cloud as correctly classified, reaching a precision of the classification of static objects of 96.9%.

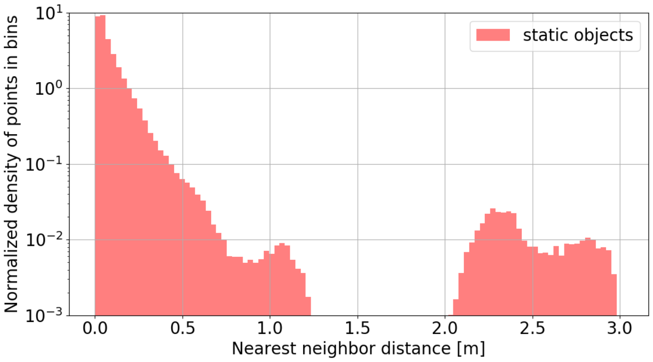

Fig. 10. Precision of static classification: normalized histogram of nearest neighbor distances d from the static part of the camera point cloud to our static LiDAR ground truth point cloud.

D. Run-time evaluation

Table I shows the timing of the modules of our pipeline when deployed on the embedded system described in Section IV-A. The two most significant contributors are the block-matching and the people-detector network. Overall, the pipeline runs at 8.5Hz with all features. Excluding the people detector, we reach 13.5Hz. The core part of our work, i.e. point cloud filtering, clustering and 3D tracking, classification, and motion estimation requires $21ms$ per stereo camera frame.

TABLE I

AVERAGE RUN-TIMES OF THE MAJOR MODULES OF OUR DYNAMIC OBSTACLE DETECTION AND TRACKING PIPELINE ON A STANDARD CPU.

Module	Timing [ms]	Portion [%]
Point Cloud Generation	55.2	46.4
2D People Detector	42.4	35.6
Point Cloud Filtering	9.9	8.3
Classification as Dynamic or Static	8.2	6.9
Remaining modules	3.3	2.8

V. CONCLUSION

In this paper we presented a method that reliably detects and tracks both generic dynamic objects and humans based on stereo images and thus provides accurate perception capabilities enabling compliant navigation in crowded places. Our novel algorithm detects generic dynamic objects based on motion and geometry in parallel to a detector network, which classifies humans as such based on their visual appearance. We handle short-time occlusions by estimating the velocities of the tracked objects and provide a 2D occupancy grid that is suitable for performing obstacle avoidance. We showed that our system is real-time capable on a computationally-constrained platform and despite the high noise level of stereo camera data we achieve a MOTP of $0.07 \pm 0.07m$, and a MOTA of 85.3% for the detection and tracking of dynamic objects, and a precision of 96.9% for the detection of static objects.

Future work will focus on advanced motion models, ground plane analysis and expanding the pipeline to use multiple perception sensors simultaneously.

REFERENCES

[1] Alireza Asvadi, Cristiano Premebida, Paulo Peixoto, and Urbano Nunes. 3d lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes. *Robotics and Autonomous Systems*, 83:299–311, 2016.

[2] Asma Azim and Olivier Aycard. Detection, classification and tracking of moving objects in a 3d environment. In *2012 IEEE Intelligent Vehicles Symposium*, pages 802–807. IEEE, 2012.

[3] Asma Azim and Olivier Aycard. Layer-based supervised classification of moving objects in outdoor dynamic environment using 3d laser scanner. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1408–1414. IEEE, 2014.

[4] Ioan Andrei Bârsan, Peidong Liu, Marc Pollefeys, and Andreas Geiger. Robust dense mapping for large-scale dynamic environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7510–7517. IEEE, 2018.

[5] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. In *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, volume 90, page 91. Citeseer, 2006.

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] Alberto Broggi, Stefano Cattani, Marco Patander, Mario Sabbatelli, and Paolo Zani. A full-3d voxel-based dynamic obstacle detection for urban scenario using stereo vision. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 71–76. IEEE, 2013.

[8] Sebastian Bullinger, Christoph Bodensteiner, Michael Arens, and Rainer Stiefelhagen. Stereo 3d object trajectory reconstruction. *CoRR*, abs/1808.09297, 2018.

[9] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[10] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

[11] Dennis Christie, Cansen Jiang, Danda Paudel, and Cédric Demonceaux. 3d reconstruction of dynamic vehicles using sparse 3d-laser-scanner and 2d image fusion. In *2016 International Conference on Informatics and Computing (ICIC)*, pages 61–65. IEEE, 2016.

[12] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1532–1545, 2014.

[13] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE, 2011.

[14] Renaud Dubé, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. An online multi-robot slam system for 3d lidars. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1004–1011. IEEE, 2017.

[15] Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc Van Gool. Moving obstacle detection in highly dynamic scenes. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 56–63. IEEE, 2009.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[17] Péter Fankhauser and Marco Hutter. A universal grid map library: Implementation and use case for rough terrain navigation. In *Robot Operating System (ROS)*, pages 99–120. Springer, 2016.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[19] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[20] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[21] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

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

[23] Cansen Jiang, Dennis Christie, Danda Pani Paudel, and Cédric Demonceaux. High quality reconstruction of dynamic objects using 2d-3d camera fusion. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2209–2213. IEEE, 2017.

[24] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 573–590, 2018.

[25] Stefan Kraemer, Christoph Stiller, and Mohamed Essayed Bouzouraa. Lidar-based object tracking and shape estimation using polylines and free-space information. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4515–4522. IEEE, 2018.

[26] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[27] Jean Lahoud and Bernard Ghanem. 2d-driven 3d object detection in rgb-d images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4622–4630, 2017.

[28] Laura Leal-Taixé, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, and Stefan Roth. Tracking the trackers: An analysis of the state of the art in multiple object tracking. *arXiv preprint arXiv:1704.02781*, 2017.

[29] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[30] Shile Li and Dongheui Lee. Rgb-d slam in dynamic environments using static point weighting. *IEEE Robotics and Automation Letters*, 2(4):2263–2270, 2017.

[31] Lucia Qi Liu, Daniel Dugas, Gianluca Ceasri, Roland Siegwart, and Renaud Dubé. Robot navigation in crowded environments using deep reinforcement learning. *EEE/RSJ International Conference on Intelligent Robots and Systems.*, 2020.

[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[33] David V Lu, Dave Hershberger, and William D Smart. Layered costmaps for context-sensitive navigation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 709–715. IEEE, 2014.

[34] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *2010 IEEE international conference on robotics and automation*, pages 300–307. IEEE, 2010.

[35] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[36] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

[37] Justin Miller, Andres Hasfura, Shih-Yuan Liu, and Jonathan P How. Dynamic arrival rate estimation for campus mobility on demand network graphs. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2285–2292. IEEE, 2016.

[38] Dongbo Min, Sunghwan Choi, Jiangbo Lu, Bumsub Ham, Kwanghoon Sohn, and Minh N Do. Fast global image smoothing based on weighted least squares. *IEEE Transactions on Image Processing*, 23(12):5638–5653, 2014.

[39] U Nguyen, F Rottensteiner, and C Heipke. Confidence-aware pedestrian tracking using a stereo camera. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4, 2019.

[40] Helen Oleynikova, Zachary Taylor, Marius Fehr, Juan Nieto, and Roland Siegwart. Voxblox: Building 3d signed distance fields for planning. *arXiv*, pages arXiv–1611, 2016.

[41] Aljoša Osep, Wolfgang Mehner, Markus Mathias, and Bastian Leibe. Combined image-and world-space tracking in traffic scenes. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1988–1995. IEEE, 2017.

[42] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[43] David Pfeiffer and Uwe Franke. Efficient representation of traffic scenes by means of dynamic stixels. In *2010 IEEE Intelligent Vehicles Symposium*, pages 217–224. IEEE, 2010.

[44] Mark Pfeiffer, Giuseppe Paolo, Hannes Sommer, Juan Nieto, Rol Siegwart, and Cesar Cadena. A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[45] Mark Pfeiffer, Ulrich Schwesinger, Hannes Sommer, Enric Galceran, and Roland Siegwart. Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2096–2101. IEEE, 2016.

[46] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[47] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[48] René Schuster, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. Combining stereo disparity and optical flow for basic scene flow. In *Commercial Vehicle Technology 2018*, pages 90–101. Springer, 2018.

[49] Wenjun Shi, Jiamao Li, Yanqing Liu, Dongchen Zhu, Dongdong Yang, and Xiaolin Zhang. Dynamic obstacles rejection for 3d map simultaneous updating. *IEEE Access*, 6:37715–37724, 2018.

[50] Mennatullah Siam, Mostafa Gamal, Moemen Abdel-Razek, Senthil Yogamani, and Martin Jagersand. Rtseg: Real-time semantic segmentation comparative study. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1603–1607. IEEE, 2018.

[51] Martin Simon, Karl Amende, Andrea Kraus, Jens Honer, Timo Samann, Hauke Kaulbersch, Stefan Milz, and Horst Michael Gross. Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[52] Wenjie Song, Yi Yang, Mengyin Fu, Fan Qiu, and Meiling Wang. Real-time obstacles detection and status classification for collision warning in a vehicle active safety system. *IEEE transactions on intelligent transportation systems*, 19(3):758–773, 2017.

[53] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-time self-adaptive deep stereo. *arXiv preprint arXiv:1810.05424*, 2018.

[54] Yan Wang, Zihang Lai, Gao Huang, Brian H Wang, Laurens van der Maaten, Mark Campbell, and Kilian Q Weinberger. Anytime stereo image depth estimation on mobile devices. *arXiv preprint arXiv:1810.11408*, 2018.

[55] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017.

[56] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic slam. *arXiv preprint arXiv:1812.07976*, 2018.

[57] David Yoon, Tim Tang, and Timothy Barfoot. Mapless online detection of dynamic objects in 3d lidar. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 113–120. IEEE, 2019.

[58] Liliang Zhang, Liang Lin, Xiaodan Liang, and Kaiming He. Is faster r-cnn doing well for pedestrian detection? In *European conference on computer vision*, pages 443–457. Springer, 2016.

[59] Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Towards reaching human performance in pedestrian detection. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):973–986, 2018.