Guindel, C., Martín, D. & Armingol, J. M. (2019). Traffic scene awareness for intelligent vehicles using ConvNets and stereo vision. *Robotics and Autonomous Systems*, vol. 112, pp. 109–122.

DOI: 10.1016/j.robot.2018.11.010

Traffic Scene Awareness for Intelligent Vehicles using ConvNets and Stereo Vision

Carlos Guindel*, David Martín, José María Armingol

Intelligent Systems Laboratory (LSI) Research Group
Universidad Carlos III de Madrid, Leganés, Spain

Abstract

In this paper, we propose an efficient approach to perform recognition and 3D localization of dynamic objects on images from a stereo camera, with the goal of gaining insight into traffic scenes in urban and road environments. We rely on a deep learning framework able to simultaneously identify a broad range of entities, such as vehicles, pedestrians or cyclists, with a frame rate compatible with the strict requirements of onboard automotive applications. Stereo information is later introduced to enrich the knowledge about the objects with geometrical information. The results demonstrate the capabilities of the perception system for a wide variety of situations, thus providing valuable information for a higher-level understanding of the traffic situation.

Keywords: object detection, pose estimation, deep learning, intelligent vehicles

1. Introduction

Technology has adopted an increasingly important role in transportation systems over the past decades. Advanced Driver Assistance Systems (ADAS) have been introduced in an attempt to deal with the fact that both wrong decision-making and driver distractions are factors involved in most traffic collisions. These systems represent an increase in the degree of automation

*Corresponding author

Email addresses: cguindel@ing.uc3m.es (Carlos Guindel), dmgomez@ing.uc3m.es (David Martín), armingol@ing.uc3m.es (José María Armingol)

Formal publication DOI: 10.1016/j.robot.2018.11.010

towards the future goal of fully autonomous driving, which is expected to lead to significant improvements in several issues associated with transportation systems; i.e., energy consumption, exploitation of transport infrastructures and, last but not least, traffic safety.

Traffic environments, particularly in urban areas, entail a higher degree of complexity for automated driving. Thus, while automated cars have been already successfully tested [1, 2], they were in most cases utterly dependent on off-line-built maps. Navigation with scarce or non-existent prior knowledge remains an open challenge due to the broad range of complex situations which is required to be handled (e.g., occluded landmarks and unexpected behaviors) within a highly dynamic, semi-structured environment.

Forthcoming self-driving systems will have to be endowed with the capability to understand complex traffic situations by themselves. This requirement relies on a robust inference of the position and motion of every traffic participant in the surrounding scene. Not only the presence of obstacles must be accounted for, but also an accurate estimation of the class to which every obstacle belongs is essential to understand and eventually predict the traffic situation.

Whereas semantic segmentation has gained traction in recent years in the intelligent vehicles literature [3], dynamic obstacles are still typically described by an object-based representation. Therefore, detection remains a critical component of the perception subsystem of an automated vehicle. Going further, semantic instance segmentation, which is still an ongoing research issue, often relies on a previous robust object detection stage [4].

Vision-based approaches [5] have been proved to be highly cost-effective while enabling close-to-production assemblies given their compact size and ease of integration. Even though active sensors, such as those based on laser or radar measurements, are more robust and can be used in different weather and lighting conditions (e.g., fog or nighttime), its high cost and its notorious impact on the vehicle styling are often considered as significant obstacles preventing widespread adoption. Furthermore, video frames provide a rich data source from which additional information can be extracted.

In this paper we present a vision-based approach aimed to perform detection and localization of the different road participants in the surroundings of a movable platform. We have focused on building a system able to fulfill the requirements that should be expected in a typical use case:

- Robust and accurate detection of the objects in the field of view of

the camera, including those partially occluded by another instance. A modern deep convolutional network–based framework is employed to perform the critical inference steps according to appearance features.

- Extraction of additional features to enable high-level inference about short-term behaviors. On top of the object detection inference, we perform viewpoint estimation, which may be used by downstream perception modules to obtain more accurate predictions of the movement of the objects.

- 3D localization of the objects within the plane on which the ego vehicle is moving. Here, we use stereo images to introduce spatial reasoning into the system.

- Real-time performance. The design choices along the different steps of the algorithm have been made accordingly to enable the execution of the complete pipeline in a fraction of a second on commercially-available hardware.

Previous works relying on deep convolutional networks usually fail to comply with all the constraints stated above, particularly the limited detection time. We present a practical-oriented application aimed to satisfy the needs that arise on real automated vehicles, and tested on our research platforms.

This paper is an extended version of the one presented at the Third Iberian Robotics Conference (ROBOT'2017), in Seville, Spain [6]. In this manuscript, we offer a further detailed description of the different steps of the algorithm, as well as more experiments that prove the effectiveness of the method under different setups, e.g., using a new stereo matching method and a new *backbone* architecture. Additionally, we offer a more refined approach for the object localization step.

The remainder of this paper is organized as follows. In Section 2, we briefly discuss related works. Section 3 gives a general overview of the work and how it fits into our research platforms. Section 4 describes the obstacle detection approach, while Section 5 introduces the scene modeling procedure. Results are reported in Section 6. Finally, Section 7 presents our conclusions about the work.

2. Related Work

As noted above, obstacle detection is an essential feature for automated driving systems. Consequently, a large number of algorithms have been historically developed to that end. Effort often focused on vehicle and pedestrian detection as these agents are the most commonly found ones in traffic scenes.

A variety of vision-based sensors have been used for object detection, including omnidirectional [7] or infrared cameras [8]; however, most approaches fall into two main categories: mono and stereo vision methods. Stereo vision provides depth information about the scene and thus is commonly used in driving applications [9].

Stereo vision algorithms are often bounded to make some assumptions about the ground or the expected free space on it [10]. However, detailed geometry information about the scene can be recovered, enabling the building of representations such as probabilistic occupancy maps [11], elevation maps [12] or full 3D models [13], where obstacles can be identified. Some methods emulate stereo vision using pairs of consecutive frames [14].

On the other hand, monocular obstacle detection is commonly based on appearance features. Selection of suitable features was traditionally the most crucial step in the performance of the detection pipeline and thus a lot of application-specific features, such as HOG-DPM [15], have been proposed to perform detection of traffic participants (e.g., cyclists [16]). Traditional features have also been used to estimate the orientation of the detected objects [17].

Representation learning based on deep neural networks has delivered a paradigm shift in recent years, showing vast improvements over hand-crafted features in several kinds of recognition tasks [18]. In particular, convolutional neural networks (CNNs) can learn hierarchical data representations that have been shown useful in a variety of tasks involved in autonomous driving, such as lane detection [19], semantic segmentation [20], stereo reconstruction [21], optical flow [22] and, of course, object detection [23].

Instead of using the classical sliding-window approach, detection with CNNs frequently relies on proposal algorithms which reduce the search space. Within this tendency, Girshick et al. introduced the now widely used region-based convolutional network (R-CNN) framework [24]. Regions can be selected according to classical similarity-based segmentation methods; however, end-to-end pipelines where every stage, including region proposal, can be effectively learned, are more frequently used nowadays. Faster R-CNN

[25] takes advantage of a region proposal network (RPN) which feeds the R-CNN responsible for the classification task. Due to its fast and accurate performance, we build our inference subsystem on top of this meta-architecture. The method has been recently extended to become Mask R-CNN [26], which includes instance-level semantic segmentation. This approach has been proved to deliver top-performing results, thus providing new insight into the relationship between object detection and semantic segmentation.

In some cases, orientation is also predicted to increase the information about the detected instances. In [27], a CNN is used for object detection and viewpoint estimation. Viewpoint is also estimated in [28] through keypoint likelihood models. In [29], a simple regression is introduced into the CNN-based detection pipeline to estimate the orientation.

Regarding the 3D localization of obstacles, Palazzi et al. [30] proposed a CNN able to map the 2D coordinates of the bounding boxes in the image to bird's eye coordinates, leveraging the training samples provided by a video game. However, accurate estimations usually rely on the measurements provided by high-resolution lidar devices [31, 32]. We argue that stereo vision systems can provide competitive results at localizing obstacles in the near environment, as will be shown in this work, while retaining the advantages of vision-based systems that were reported in Sec. 1.

3. System Overview

This work presents an approach for object detection and localization, aimed to become the core element of the perception module of an intelligent vehicle. To assess its adequacy, we have implemented the system in the IVVI 2.0 (Intelligent Vehicle based on Visual Information) research platform [33], shown in Fig. 1. IVVI 2.0 is a manned vehicle, equipped with cutting-edge automotive sensors, which is meant for development and testing of ADAS.

Visual sensing units in the IVVI 2.0 include a trinocular stereo camera covering the field of view in front of the vehicle (Fig. 1b), which is the source of the images used by our pipeline. The support system used for fixing the camera to the windshield allows for a degree of freedom (pitch) to adapt its pose to the restrictions imposed by specific modules, e.g., the lane detection system [34]. For that reason, we cannot assume negligible the pitch angle between the camera and the ground plane. In any case, calibration is a recurring issue in most vehicle sensor setups.

Figure 1: IVVI 2.0, Intelligent Vehicle based on Visual Information: (a) external view; (b) detail of the trinocular stereo vision system mounted on the windshield.

The processing unit includes a high-performance GPU which provides resources for high-parallel processing, therefore enabling CNN-based inference in real time. Robot Operative System (ROS)[1] is used for inter-module cooperation. Our implementation makes use of its highly modular structure, so separated nodes perform the different functions involved in the process.

The method presented here provides a self-contained pipeline that maps from the visual inputs to the object-based representation of the environment. The approach consists of two main branches that are intended to be performed in parallel, as shown in Fig. 2:

1. Object detection and viewpoint estimation based on appearance. Features are extracted exclusively from the left image of the stereo pair.
2. Object localization, robust to changes in position and orientation of the vision system resulting from the vehicle movement. A stereo-based 3D reconstruction is performed, and the extrinsic camera parameters are extracted under a flat ground assumption.

We designed our approach in such a way that these separated branches extract information to be fused only at the latest stages of the pipeline. In this way, the object detection task can be performed almost entirely on the GPU, without having to share resources with the spatial localization subsystem. The latter can instead make intensive use of either a typical multi-core

[1] `http://www.ros.org/`

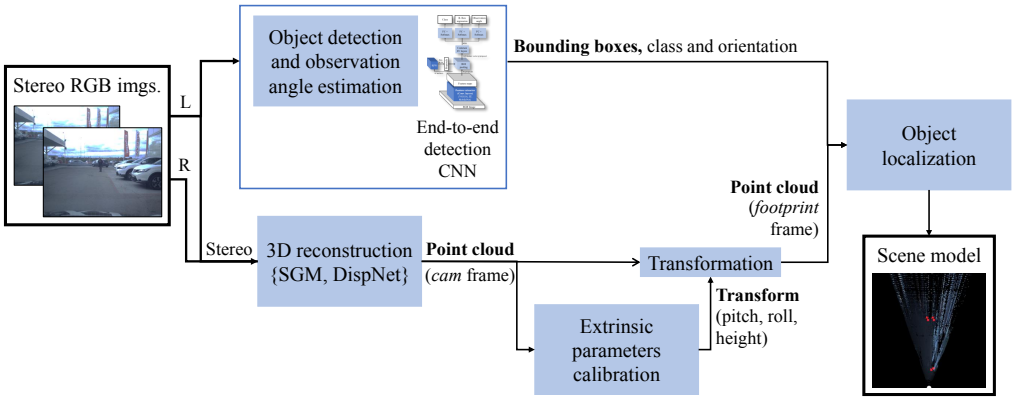


Figure 2: Proposed approach overview. The CNN architecture will be shown in detail in Fig. 3.

CPU or another GPU, depending on the stereo matching algorithm in use (as will be detailed in Sec. 5). This twofold process has been designed to fill the available computing capability, thus allowing to comply with the time requirements inherent to the application.

4. Obstacle Detection

Automated vehicles have to deal with a wide variety of dynamic obstacles in the highly unstructured traffic environments. Object classification, which aims to identify predefined image regions, can be performed in myriad ways using deep convolutional networks; however, object detection additionally requires locating every object within the image coordinates.

Among the modern detection meta-architectures that have been developed over the last years, we adopt Faster R-CNN [25] for our identification branch, due to its particularly compelling set of features:

- It is an end-to-end trainable framework spanning from image pixels to the final prediction, so no assumptions must be made about the position of the objects in the image.
- It can deal with a large number of classes with no major impact on performance; in fact, the number of classes only affects the number of parameters of the last set of layers.

- It is suitable for real-time tasks while achieving state-of-the-art accuracy results, and provides the basis for some modern instance-level semantic segmentation algorithms.

This framework admits potentially infinite *backbone* architectures, but always involves two well-differentiated stages: a region proposal network (RPN) and a classification step. The former is responsible for identifying image regions where objects are likely to be located, while the latter assigns a category to those proposals. Both components share the same set of convolutional layers, enabling real-time frame rates.

In its original setup, Faster R-CNN provides a bounding box refinement along with the classified regions. In this work, we additionally adopt the strategy introduced in [35] to incorporate the inference of the *observation angle* (viewpoint) into the detection framework. Hence, we benefit from the already computed convolutional features to obtain an estimation of the orientation of the objects with respect to the ego-car, therefore providing valuable information to build the scene model. According to the requirements of the application, only the yaw angle (i.e., azimuth) from which objects are visible is to be estimated, since both relevant obstacles and the ego-vehicle are assumed to move on the same ground plane.

As with the region proposals from the RPN, the viewpoint can be estimated at almost no cost during test-time given that convolutions are computed only once. Fig. 3 schematically illustrates the architecture in use.

Features are extracted through a *backbone* architecture and subsequently shared across the different inference tasks. The selection of that feature extractor is a critical choice with a notable impact on the overall performance of the algorithm, as will be demonstrated in Sec. 6. On the other hand, the *head* structure, made of fully connected layers, is a well-established architecture that has proven more effective than other one-shot approaches in significant cases (e.g., distant objects) [36]. The custom viewpoint extension is described in detail in the following subsections.

4.1. Discrete Viewpoint Approach

A discrete approach is adopted for the orientation estimation so that the full range of possible viewpoints (2π rad) gets divided into N_b bins which span $2\pi/N_b$ rad each one. One-hot encoding is used, so only one of these bins is used to describe each object's viewpoint. In this way, we can pose the problem as a multinomial classification where the class labels are the angle

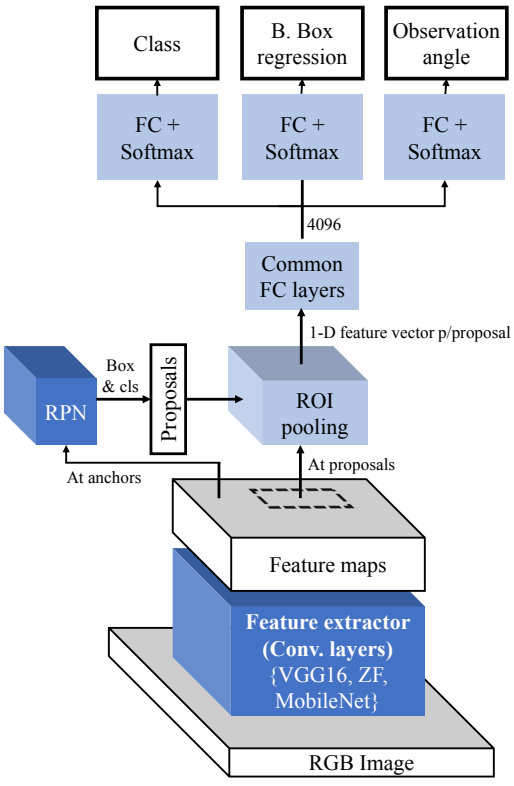


Figure 3: Proposed object detection and viewpoint estimation approach.

bins themselves. We have found beneficial to make this prediction class-aware; therefore, we modify the network architecture to provide a vector $\mathbf{r}$ representing K separated categorical distributions $\mathbf{r}^k$ over the N_b possible bins, one per category:

$$\mathbf{r}^k = (r_0^k, \ldots, r_{N_b-1}^k) \text{ for } k = 0, \ldots, K-1 \tag{1}$$

Each element r_i of this sub-vector expresses the probability of the object's orientation being within the range spanned by the bin with index i. Since a numerical prediction is eventually required to enable high-level interpretation, we finally assign to each detection the value of the center of the bin with the highest probability, b^*. If we choose the bin with index 0 to start at 0 rad, that numerical estimation would be computed as:

$$\hat{\theta} = \frac{\pi(2b^* + 1)}{N_b} \tag{2}$$

Note, however, that we usually choose the endpoints of the angle intervals such as their centers are aligned with the common relative directions, i.e., left, right, forward and backward; hence, such offset should be taken into account in the calculation as an angular offset.

4.2. Joint Detection and Viewpoint Estimation

As established by the R-CNN paradigm and depicted in Fig. 3, features corresponding to each proposal are pooled and propagated downstream to allow the final layers of the network to perform the inference. In the original framework, the 4 096-element feature vector provided by the common set of *fully connected* (FC) layers is employed to provide a class and a bounding box regression for each proposal. Additionally, we also use this outcome to perform viewpoint inference, based on the intuition that appearance features can discriminate among different classes and orientations at the same time. This design fits naturally into the Faster R-CNN meta-architecture and bears some resemblance to the implementation of the native bounding box refinement task.

As shown in the upper part of Fig. 3, the feature vectors from the common FC layers are finally fed into three sibling FC layers. As in the original approach, the first two sibling layers provide classification and a bounding box regression, respectively. On the other hand, the new third layer is responsible for giving an estimation of the viewpoint, which is ultimately normalized through a softmax function to provide K different probability distributions, as expressed in Eq. 1. It is important to note that here K includes a catch-all background class, $k = 0$, even when $\mathbf{r}^0$ is obviously not used since samples classified as background are not actual detections.

4.3. Loss Function

According to the results reported in [25], we adopt an approximate joint training strategy, which has been shown to offer the best time-precision trade-off. Both the RPN and the classification stage are trained simultaneously. The loss function used for backpropagation at every iteration is composed of the sum of individual losses which account for the respective tasks (i.e., region proposal, classification, and bounding box regression).

Likewise, viewpoint estimation is introduced as a new component of the loss. As we are dealing with a one-of-many classification task, we use a multinomial logistic loss that only adopts non-zero values for the ground-truth class. Hence, from the $N_b K$-dimensional output $\mathbf{r}$, we only take into account the N_b elements belonging to the ground-truth class, with index u^*, when computing the loss. The loss is aggregated over the N_B samples of the mini-batch B:

$$L_v = \frac{1}{N_B} \sum_{j \in B} [u^* > 0] L_{cls}(\mathbf{r}_j^{u^*}, b_j^*) \tag{3}$$

where $L_{cls}(\mathbf{r}_j^{u^*}, b_j^*)$ is the multinomial logistic loss which accounts for the divergence between the prediction $\mathbf{r}^{u^*}$ and the index of the ground-truth bin, b^*. Note the use of the Iverson bracket to ignore samples that are classified as *background* ($u^* = 0$).

On the other hand, we use a weighted version of the multinomial logistic loss for the classification task, implemented as an *infogain* loss with a diagonal *infogain matrix*. This way, we intend to counter the effect of the significant class imbalance that can be observed in real-world datasets. We assign higher weights to those samples belonging to classes which are underrepresented in the training set to increase their contribution to the total loss.

The loss function is ultimately composed of the sum of five components, one per task, normalized by the size of the respective batches. Although different weights might be applied to the components of the loss function, we obtain satisfactory results assigning the same weight to every component.

4.4. Implementation Details

We use a custom set of RPN *anchors* obtained through statistical analysis to fit the objects in traffic environments. The number of anchors is set to a total of nine (three scales and three aspect ratios) to limit the computational requirements.

At the end of the detection pipeline, a non-maximum suppression (NMS) is applied, taking into account the classification score assigned to each detection by the network. As we train our models to detect a broad set of classes, and some of them are very similar to each other, we perform the suppression considering detections from similar classes together. Thus, we avoid obtaining duplicate detections corresponding to the same instance, as is often the case for neighbor classes such as *Car* and *Van*, or *Pedestrian*

and *Cyclist*. An intersection-over-union (IoU) overlap of 30% is required to assume redundant detections.

5. Scene Modeling

The second branch of the algorithm aims to gather geometrical information about the scene to augment the object detection from the first part and build an instantaneous, local model of the traffic participants in front of the vehicle. To that end, we solely rely on data from the two cameras in the stereo rig, which are used to build a dense 3D reconstruction of the environment.

We consider two different coordinate frames in our system: the *camera* frame, which is attached to the optical center of the left camera, and the *footprint* frame, which is placed just underneath the *Camera* frame and oriented so that the xy-plane and the ground plane are coincident. Both coordinate frames move together with the ego-car.

Initially, the 3D point cloud of the scene is represented in *camera* coordinates. However, the most valuable piece of information is instead provided by the location of the obstacles on the ground plane, where their movement can be represented as a 2D displacement.

If the ground is assumed to be flat in a relatively small neighborhood from the vehicle, extrinsic parameters of the stereo system can be estimated in an online fashion. This way, it is possible to consider and properly remove not only the relative orientation of the camera with respect to the car but also the camera pose changes with respect to the ground plane that take place due to the vehicle movement (e.g., while traveling on uneven road surfaces). The flat ground assumption holds for most driving environments, as abrupt road slope changes are rare.

Through the following process, obstacles first detected through the object detection stage can be located in *footprint* coordinates and assigned an absolute yaw angle making use of the depth information.

5.1. Stereo 3D Reconstruction

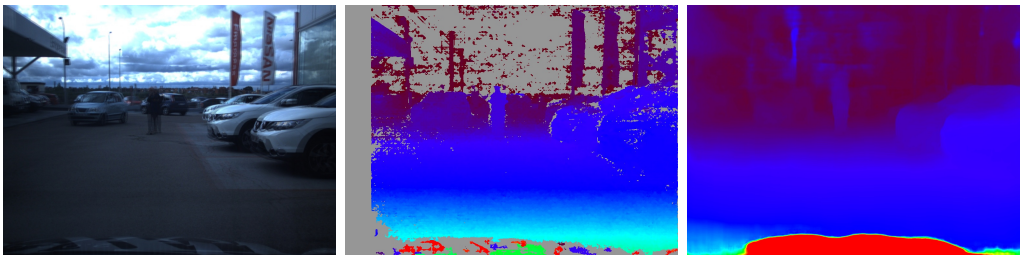
Different stereo matching algorithms can be used to retrieve the geometrical information from the stereo image pair. As usual, the features demanded by our algorithm for this component include good accuracy, since errors are propagated to the object positioning; high density, so that 3D points can be

assigned to every object detected in the image; and, finally, a low computational burden to allow real-time processing.

We have tested two alternative stereo matching algorithms:

1. A semi-global matching approach [37]. Despite being overcome by more recent approaches, the SGM method still shows a good trade-off between accuracy and computation time, with a decent density and good generalization capabilities.
2. A state-of-the-art, CNN-based model: DispNet [38]. This architecture features a simple design which achieves competitive results for a fraction of the cost of virtually every other CNN-based approach. Unlike SGM, this method provides a 100% dense disparity map. We use the publicly available model with a 1D correlation layer and fine-tuning on the KITTI dataset.

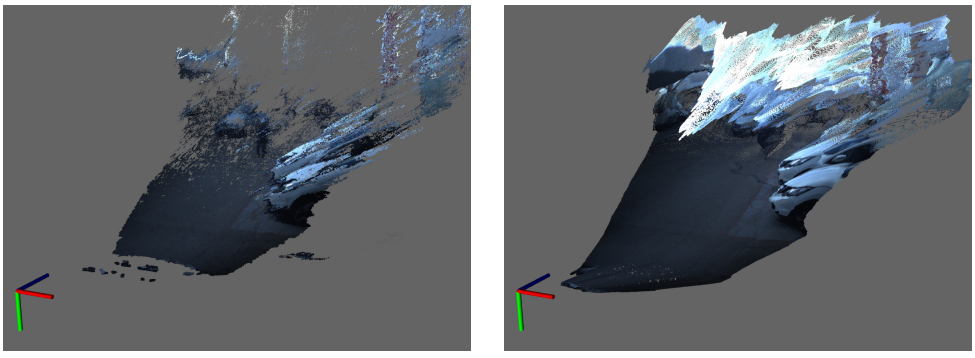
In general, both methods cope well with challenges posed by road environments, e.g., lack of texture or changes of illumination, which makes them suitable for the intended application. We will analyze the performance of these two algorithms in the context of our approach in Sec. 6. An example of the resulting disparity maps, which encode the distance between corresponding points in both images and whose values are therefore inversely proportional to the depth estimation, is provided in Fig. 4, using the IVVI 2.0 stereo camera, with a baseline of 12 cm. Although the results from the DispNet are qualitatively better, we have found that they usually present some spurious estimations in the upper parts of the image, probably due to the uneven distribution of the ground-truth points used to train the model (obtained with a down-looking lidar).



(a) (b) (c)

Figure 4: Stereo matching algorithms comparison: left and right images overlapped (a); SGM disparity map (b); DispNet disparity map (c)

Whichever algorithm is used, stereo matching provides a dense depth estimation. Therefore, it is possible to assign a depth value to virtually every pixel of the image and then to build a *projectable* 3D point cloud which preserves the relationship between pixels and 3D points. Fig. 5 shows the clouds obtained with the depth estimations from Fig. 4. Depth artifacts introduced by the DispNet matching are visible in the sky in Fig. 5b.



(a) (b)

Figure 5: 3D point clouds obtained with the two tested stereo matching algortihms: (a) SGM; (b) DispNet.

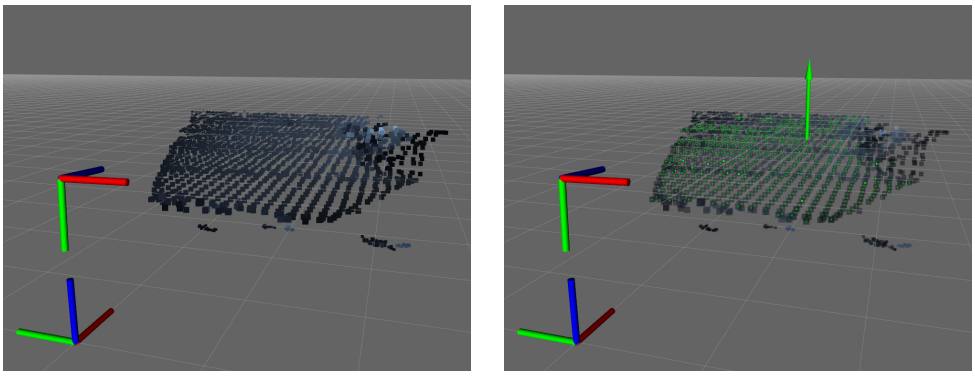
5.2. Extrinsic Parameters Auto-Calibration

The auto-calibration procedure is based on finding the plane of the road in front of the vehicle. To avoid spurious detections, we perform the search on a subset of the original point cloud. Pass-through filters are applied to the point cloud in order to limit the search area to the region in front of the vehicle where the ground plane is likely extractable. We filter out the points closer than 2 m or further than 20 m, as well as those outside a width range of 12 m around the depth axis of the *camera* frame. Within those ranges, the flatness assumption is fulfilled with a high probability.

Later, the point cloud is downsampled using a voxel grid with a voxel size of 20 $\times$ 20 $\times$ 20 cm. In addition to reducing the amount of data to be processed, this filtering is aimed to normalize the point density along the depth axis. The resulting cloud from the exemplary image pair is depicted in Fig. 6a.

Coefficients defining the ground plane must be estimated as a first step to obtain the extrinsic camera parameters. Points comprising the filtered point-

cloud are fitted to a plane using a sample consensus approach, RANSAC [39]. Following [40], a tight threshold of 1.5 cm is used. The search is reduced to planes perpendicular to the vertical axis of the camera, with a tolerance of 0.35 rad, as we assume the depth axis of the camera frame to be roughly parallel to the ground plane. Fig. 6b illustrates the ground plane obtained from the voxel-filtered point cloud.



(a) (b)

Figure 6: Extrinsic parameters estimation pipeline: (a) cropped, downsampled cloud; (b) plane inliers (in green) over the cloud, with the normal to the estimated plane represented as an arrow. *Camera* (top) and *footprint* (bottom) frames are represented.

It can be shown [40] that, given a road plane defined by $ax_c+by_c+cz_c+d = 0$, with (x_c, y_c, z_c) being the coordinates of a point belonging to the plane, roll (ψ), pitch (ϕ) and height (h) defining the camera pose can be obtained as:

$$\psi = \arcsin(a) \qquad \phi = \arctan\left(\frac{-c}{b}\right) \qquad h = d \tag{4}$$

Yaw angle cannot be extracted solely from the plane, and thus it is assumed to be nil. We do not translate the *footprint* coordinate frame along x and y camera axes, although that displacement may be arbitrarily chosen (e.g., the origin might be centered at the front end of the vehicle).

It is noteworthy that this method also produces as a by-product a rough estimation of the free space in front of the vehicle. This outcome is straightforwardly given by the inliers of the plane segmentation.

5.3. Object Localization

Both branches of the algorithm merge at the object localization module, where detections are provided with an estimation of their x and y coordinates on the ground plane. To that end, the correspondence between points in the image and points in the 3D cloud is exploited. Data association between both structures is preserved due to the point cloud used in the pipeline being *organized* and *projectable*.

At this point, the transform obtained from the previous step is applied to the point cloud to express the points' coordinates in the *footprint* frame. Then, points belonging to the ground, as well as those too close to the camera (< 3 m), which are assumed to belong mainly to the hood of the car, are removed.

Henceforth, the estimation of the location of each detected object relies on the 3D points falling into its bounding box. However, the association between points and object is not a trivial task. Firstly, it is easily noticeable that not all points enclosed by a bounding box belong to the detected instance. For instance, in Fig. 7 is apparent that part of the background is necessarily included within the detection box.

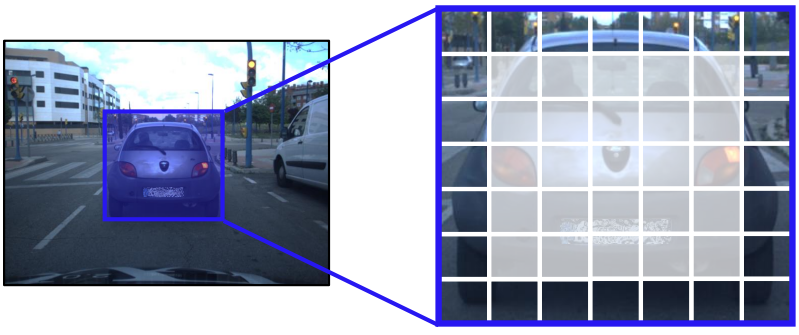


Figure 7: Grid division of the bounding box representing a detection, where the sub-boxes used for localization in our approach are filled in semi-transparent white.

Inspired by the concept of ROI pooling [41], we solve this problem by dividing every $h \times w$ bounding box into a $H \times W$ grid of sub-boxes of size $h/H \times w/W$. Then, only a subset of the sub-boxes is considered for the task (Fig. 7). We use $H = 7$ and $W = 7$, and take into account all the boxes, except those in the outermost rows and columns of the ROI (i.e., the 5×5

inner grid). Our decision is based on statistical analysis and seeks to maximize the generalization ability of the method against the notable intraclass variability in shape. Different setups might be used for the various categories or estimated viewpoints, but we did not observe significant improvement. It is noteworthy that this approach deals naturally with the scale variability.

Another relevant issue is related to the nature of the stereo matching process itself. Stereo methods are intended to recover the 3D location of the points on the surface of the objects. Nonetheless, we aim to estimate the position of the center of the objects, enabling the use of meaningful higher-level models of the dynamics of every instance.

Here we assume a fixed size for each of the possible classes, based on aggregated statistical data from the KITTI dataset [42], and approximate the objects to a prismatic shape with these dimensions. From Fig. 8, assuming that the surface point $\mathbf{p}_s$ (in footprint coordinates) can be obtained from the point cloud using the sub-boxes method defined above, the central position of the object, $\mathbf{p}_0$ can be computed extending the direction defined by the vector $\mathbf{p}_s$ a distance $\Delta s = \sqrt{\Delta x^2 + \Delta y^2}$. Δx and Δy are only dependent on the dimensions of the object ($L_0 \times W_0$) and the point of view (θ), which is conveniently provided by the detection branch.

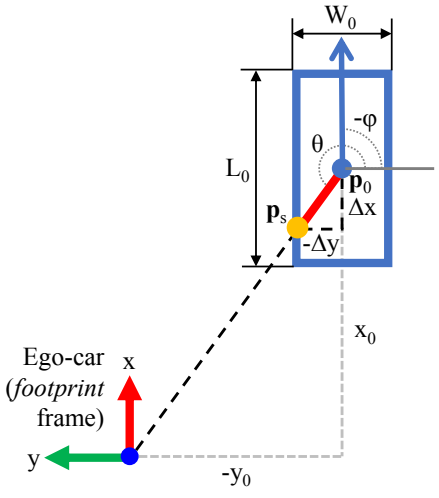


Figure 8: Schematic representation of the top-down view of the scene with a detected object (a car facing forwards).

To obtain $\mathbf{p}_s$, we rank separately two lists containing the x and y co-

ordinates of every point within the selected sub-boxes and extract the first quartile. The resulting two values are used as an estimation of the x and y coordinates of $\mathbf{p}_s$. The use of the first quartile intends to remove the influence of the outliers and to ensure that the retrieved point actually belongs to the object's surface.

Once determined $p_0 = (x_0, y_0)$, and considering the angle and coordinate frame definitions in Fig. 8, the actual yaw angle of the object, φ, is obtained as follows:

$$\varphi = \alpha + \text{atan2}(x_0, y_0) + 3\pi/2; \tag{5}$$

Note that this yaw value represents the rotation of the object around a local vertical axis.

As an unavoidable consequence of using stereo vision to retrieve geometrical information, it is assumed that the accuracy in locating nearby objects will be greater than for those farther away. The evolution of the error in the depth estimation, δz, with the distance, z, can be approximated [43] by:

$$\delta z = \frac{z^2}{fB}\delta d \tag{6}$$

where f is the focal length, B the baseline, and δd the error made in the stereo matching process. According to existing stats, we assume $\delta d \approx 2$ for SGM and $\delta d \approx 1$ for the DispNet method.

6. Results

For assessing our approach, we use a twofold validation procedure which aims to prove the adequacy of the method at meeting the strict demands of automotive applications. Hence, we first quantitatively evaluate the accuracy of the critical parts of the algorithm using a well-established image benchmark and later, we use our research platform to test the method in a variety of real traffic situations, thus demonstrating its generalization ability.

6.1. Object Detection and Viewpoint Estimation

Experiments to quantitatively assess the object detection and viewpoint estimation branch have been conducted on the KITTI object detection benchmark [42], taking advantage of the per-instance class and orientation labels available for evaluation. This widely-used dataset features a large number of

labeled objects in a variety of pose, occlusion, and truncation circumstances, thus ensuring the representativeness of the obtained results.

Since annotations for the testing set are not publicly available, we have divided the training set into two splits for training and validation. We use a custom split which makes sure that images from the same sequence are not included simultaneously in both subsets. As we aim at a ratio of 70:30, 5 415 labeled pictures are used in training whereas 2 065 are subsequently employed to validate our algorithm. The number of instances for each split is tabulated in Tab. 1. Please note that samples with requirements further than the established for the *Hard* difficulty level (min bounding box: 25 pixels, max. occlusion level: difficult to see, max. truncation: 50%) are not considered either for training or validation.

	Car	Pedest.	Cyclist	Van	Truck	Per. sit.[1]	Tram[1]
train samples	20 609	2 476	1 042	2 407	965	222	511
val. samples	8 120	2 010	584	506	129	-	-
total	28 729	4 486	1 626	2 913	1 094	222	511

[1] Categories used only for training

Table 1: Object occurrence statistics of the train and validation splits.

To explicitly assess the performance of the algorithm under close-to-production conditions, where a variety of categories must be identified, we use the seven available foreground classes, all except for *DontCare* and *Misc*, to train the network. Special care was taken to avoid using regions belonging to those two excluded labels at training time, as they might harm the learning process. On the other hand, *Person sitting* and *Tram* categories are only used for training as the low number of samples discourages its use in validation. Confusion between neighbor classes is not accounted as an error. It is important to note that most works in the field limit the validation stage to the *Car*, *Pedestrian* and *Cyclist* classes, due to the high intra-class variability and the low number of samples featured by the rest of categories.

We mostly follow the KITTI benchmark regarding the evaluation metrics. Therefore, average precision (AP) and average orientation similarity (AOS) are used to assess object detection and orientation estimation, respectively. We occasionally use the mean across the different categories (mAP

and mAOS) to summarize the inference performance. The minimum required intersection-over-union (IoU) overlap between detections and ground-truth is 70% for motor vehicles and 50% for people and cyclists.

Our experimental setup is based on a python-based implementation built upon Caffe [44], and NVIDIA Titan Xp GPUs.

From now on, we study the effect of the *backbone* architecture and other hyperparameters (e.g., the input scale and the number of proposals) in the performance of the algorithm. Later, we offer a breakdown of the results for the top-performing choices.

6.1.1. Hyper-Parameters and Feature Extractor Selection

As our approach is agnostic to the particular architecture used at the feature extractor stage (i.e., the convolutional layers) of the network, we investigate three different models, which are known for their high efficiency. Whereas we use different initial learning rates and numbers of iterations for each model, we always use an asynchronous SGD with a momentum of 0.9 and reduce the learning rate by 10x every time a 1/3 of the total iteration count has been completed. The stride of the feature maps is 16 in all cases. A model pre-trained on ImageNet is used to initialize the weights.

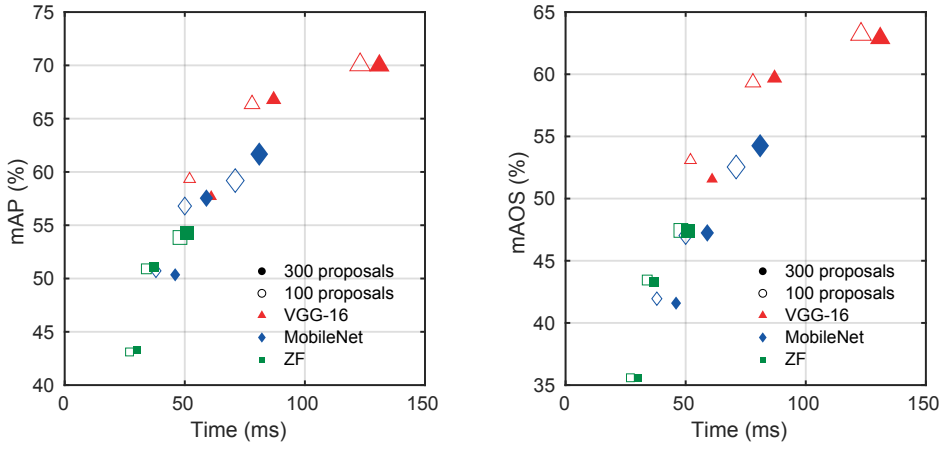
1. Zeiler and Fergus (ZF) [45], an improved variant of AlexNet, with five convolutional layers. 256 features are used by the RPN, and a 6×6 ROI pooling is performed. The initial learning rate is 5e-4 and training lasts for 90k iterations.
2. VGG-16 [46], with 13 convolutional layers. 512 features are used by the RPN, and a 7×7 ROI pooling is performed. The initial learning rate is 5e-4 and training lasts for 150k iterations.
3. MobileNet [47], recently released with the focus on embedded applications. We extract the features from the *Conv2d_11* layer, after 18 convolutional layers (half of them, in its depthwise version). 512 features are used by the RPN, and a 7×7 ROI pooling is performed. The initial learning rate is 1e-3, and training lasts for 150k iterations.

Training is performed with a random horizontal flip as data augmentation. We do not freeze any weights (not even the shallowest ones), nor use dropout.

Following suggestions from recent studies [36], we also study the number of proposals that the RPN send to the box classifier at test time and the scale of the input image. In this work, we perform the experiments in such a

way that the input scales at train and test times concur; in other words, we train a different model for each input scale.

This set of hyperparameters has an impact on the accuracy of the detection and viewpoint estimation, but also on the output frame rate of the algorithm. Consequently, we study the relationship between both magnitudes for the main object categories in Fig. 9. Time is given as the median of the processing time for every frame in the validation split.



(a) (b)

Figure 9: Accuracy vs. time for different number of proposals, architectures and input scales (375, 500 and 650, indicated through the marker size): (a) detection performance; (b) orientation estimation performance. Only samples in the *Moderate* difficulty level and belonging to *Car*, *Pedestrian* and *Cyclist* are considered.

Results are given for the *Moderate* difficulty level subset defined by the KITTI dataset (min. bounding box height: 25 Px, max. occlusion level: "partly occluded", max. truncation: 30%), but comparable results are obtained used the *Hard* difficulty level. In this experiment, the number of orientation bins is set to $N_b = 8$.

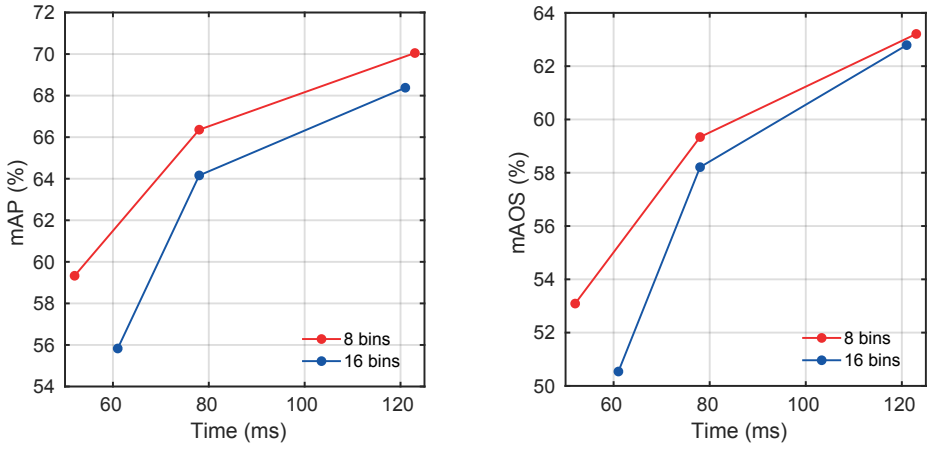
Surprisingly, reducing the number of proposals does not harm mAP or mAOS and even improves the results in some cases, due to a reduction in the occurrence of false positives. VGG-16 achieves the best accuracy but is also the slowest architecture; at the other end of the scale, ZF is the most lightweight model but produces suboptimal results. Finally, the image size has a very significant influence on both sides. As expected, applying a

scale factor over the baseline of 375 pixels in height improves the accuracy, although the precision plateaus at 1.77× (650 pixels).

6.1.2. Viewpoint Estimation

A particular case of hyperparameter is the number of bins, N_b, used for the viewpoint estimation. Even when results for $N_b = 8$ are satisfactory regarding orientation similarity, it may be advisable to provide a finer-grain estimation in some particular cases. Yaw angle has a major influence on the estimation of the 3D location of the objects, and the resolution of $\pi/4$ obtained when $N_b = 8$ might not be enough for the higher-level modules to apprehend the structure of the traffic scene.

We have tested that $N_b = 16$ provide similar detection results regarding average precision. In Fig. 10 we compare the performance of the algorithm with the VGG-16 *backbone* for the three different input scales tested and two variants of the number of bins N_b: 8 and 16.



(a) (b)

Figure 10: Accuracy vs. time for VGG-16, with different number of bins and different input scales: (a) detection performance; (b) orientation estimation performance. Only samples in the *Moderate* difficulty level and belonging to *Car*, *Pedestrian* and *Cyclist* are considered.

The detection performance is minimally reduced when N_b is increased, as is also the mAOS. An example of the effect of this parameter in the resulting scene model will be provided in Sec. 6.2.

6.1.3. Class-Disaggregated Analysis

According to the results above, the VGG *backbone*, with 100 RPN proposals, offers the best performance among the analyzed alternatives. In Table 2, we provide a breakdown of the results for the analyzed scales given $N_b = 8$.

It is noteworthy that for the *Easy* difficulty level, made of samples larger than 40 pixels, the effect of the scale is significantly lower than for the *Moderate* and *Hard* samples. The weak results that we obtain for the *Van* and *Truck* classes can be explained by several factors: the reduced number of training samples, the high intraclass variability (e.g., box trucks vs. dump trucks) and the demanding IoU required for these categories.

Comparison with other existing algorithms is not entirely fair because of two reasons: on the one hand, top-performing algorithms are not trained to detect all the seven available categories; and on the other hand, we perform our experiments using a custom train/validation split. Nevertheless, we provide a brief comparative in Table 3 against two comparable methods (i.e., providing multi-class detection and orientation estimation), but it should be noted that train/test sets, as well as the hardware used to extract the computation times, are not homogeneous. Despite this, it is remarkable that the accuracy level achieved by our method is on-par with these more sophisticated approaches.

6.2. Object Localization

Tests for the assessing of performance of the object localization module were also performed on the KITTI dataset, where a precise annotation of the 3D location of the labeled instances is available. It should be noted that ground-truth locations are provided in the camera frame, so the evaluation of the extrinsic parameters calibration cannot be included here.

For this evaluation, we rely on the best-performing model among the different ones that we tested on the previous section; that is, VGG-16, with scale 650 and 100 proposals. We only consider true positive detections according to the criteria established before; we also set a minimum threshold of 20% for the detection score. Fig. 11 shows the distribution of the errors per category for each stereo matching method. The error is represented as the absolute Euclidean distance in the xz-plane of the stereo camera between the estimated location and the ground-truth position.

Despite the presence of a substantial amount of outliers, the median of the localization error is 0.771 m when using SGM and only 0.517 m with the DispNet method. However, it is important to note that we are using a

diffic.	scale	Car	Pedest.	Cyclist	Van	Truck	time (ms)
		Average Precision (AP)					
	375	83.86	77.53	56.88	37.16	10.58	52
Easy	500	88.54	79.73	73.79	44.39	15.03	78
	650	90.14	85.57	74.32	40.08	10.75	123
	375	71.59	64.48	41.91	30.25	9.96	52
Moderate	500	77.83	67.77	53.47	38.87	14.77	78
	650	84.49	70.39	55.28	34.78	12.30	123
	375	56.33	59.68	40.96	30.32	6.11	52
Hard	500	60.40	61.46	52.12	37.52	10.56	78
	650	67.11	66.34	54.23	35.58	9.17	123
		Average Orientation Similarity (AOS)					
	375	82.24	71.13	42.02	36.59	9.04	52
Easy	500	86.88	73.42	54.12	43.90	11.52	78
	650	88.75	78.76	57.19	39.66	10.62	123
	375	69.78	58.31	31.17	29.00	8.31	52
Moderate	500	76.14	61.42	40.45	37.16	9.94	78
	650	82.92	64.28	42.42	32.76	10.35	123
	375	54.37	53.74	30.68	28.60	5.05	52
Hard	500	58.86	55.70	39.47	36.06	7.21	78
	650	65.53	60.24	41.78	33.60	7.19	123

Table 2: Detection and orientation estimation performance for different scales and difficulty levels using the VGG-16 *backbone* and 100 proposals.

DispNet model fine-tuned explicitly for the KITTI stereo dataset; overall, the difference is not very significant.

As stated above, the depth estimation error increases with the distance to the camera. In Fig. 12 we study the distribution of the error as a function of the distance in the depth axis of the *camera* frame. We also represent the quadratic error of the depth estimation, from Eq. 6, as a reference.

As can be observed in both representations, localization error is acceptably low, particularly within the 0-20 m range. Truck localization seems to

diffic.	Car	Pedest.	Cyclist	Van	Truck	time (s)
	Average Precision (AP)					
SubCNN [48]	88.86	71.34	70.77	-	-	2
Pose R-CNN [49]	75.74	63.38	68.04	-	-	2
Ours (VGG, 650)	84.49	70.39	55.28	34.78	12.30	0.123
	Average Orientation Similarity (AOS)					
SubCNN [48]	88.43	66.28	63.41	-	-	2
Pose R-CNN [49]	75.35	59.89	62.25	-	-	2
Ours (VGG, 650)	82.92	64.28	42.42	32.76	10.35	0.123

Table 3: Comparison with other methods of the detection and viewpoint estimation performance (%). Results for *Moderate* samples are shown.

be affected by unusually high errors, although the significance of the results is diminished by the low number of samples in the category. On the other hand, the occurrence of pedestrian outliers in the >20 m range is also especially notorious, due to the incidence of occlusions (e.g., pedestrians behind vehicles).

To study the effect of the technique introduced in Sec. 5.3 to estimate the center of the object, we compare our method with a naive implementation where the predicted location is directly given by the average of the coordinates of the points within the selected ROI. Results of the signed difference between the estimation and the ground-truth for the depth coordinate are provided in Fig. 13. It is noticeable that the median of the error is closer to 0 using the proposed approach, whereas the naive approach introduces a negative bias in the measurements.

We also posed the problem as a classification in bird's eye view, as in the recent KITTI benchmark[2], and compute detection precision-recall stats. Results are presented in Table 4 for two different minimum IoU thresholds.

The resulting local scene model can be suitably represented as a top-down view over the ground plane, with the position and class of the objects. Thus, Fig. 14 shows examples of the performance of our method on the

[2] `http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=bev`

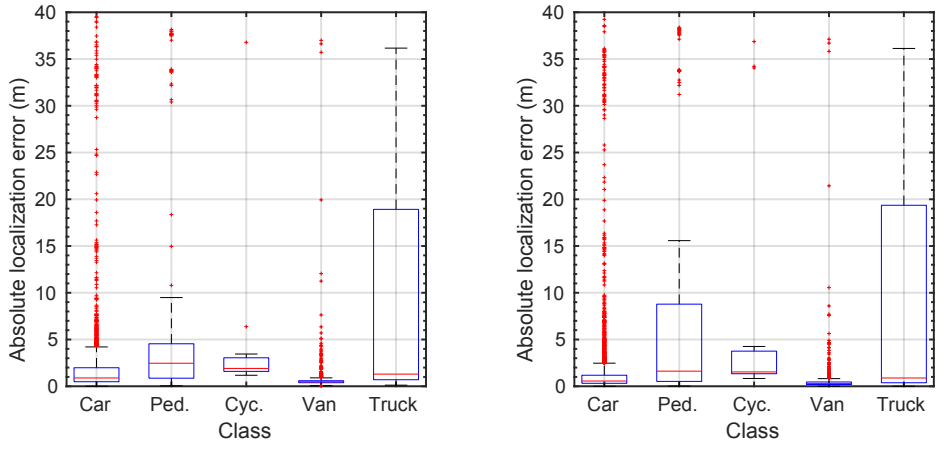


Figure 11: Absolute Euclidean error in the location estimation for the different classes, using two different stereo matching algorithms: (a) SGM; (b) DispNet. The central mark represents the median, and the bottom and top edges of the box indicate the first and third percentiles, respectively; outliers are represented outside the whiskers.

min IoU	diffic.	Car	Pedest.	Cyclist	Van	Truck
		Average Precision (AP)				
20%	Easy	77.67	38.55	32.72	25.55	18.68
	Moder.	66.49	31.71	22.22	21.09	12.73
	Hard	49.88	31.42	22.02	15.38	11.16
40%	Easy	70.64	12.64	8.35	22.16	8.68
	Moder.	53.23	12.95	5.19	18.84	8.18
	Hard	38.18	9.17	5.17	13.77	4.55

Table 4: Detection performance evaluated in bird's eye view.

KITTI dataset for the two variants of the number of bins (N_b). Each sample depicts the detections in the image, as well as the corresponding local scene models, where obstacles are located on a top-down view of the reconstructed point cloud. KITTI ground-truth detections are depicted in a faded color. Additionally, points falling in a 10 cm interval around the ground plane are projected in green on the image, as a rough free space estimation.

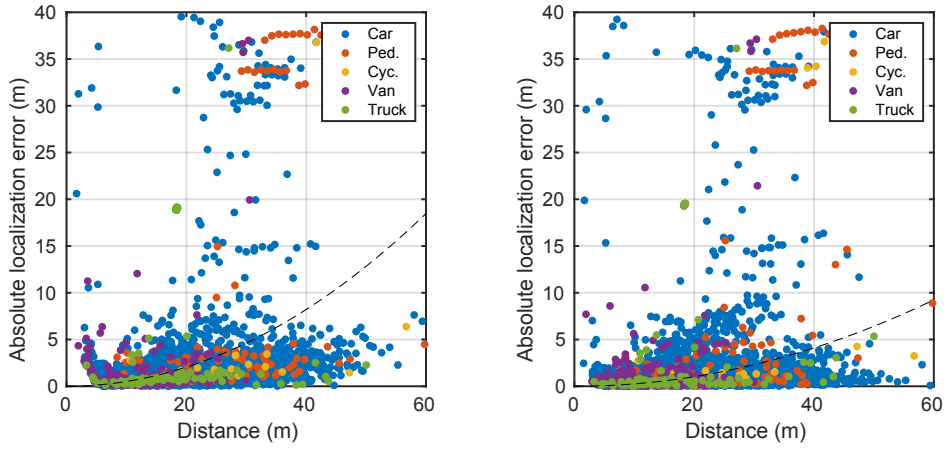


Figure 12: Localization error vs. distance using two different stereo matching algorithms: (a) SGM; (b) DispNet. The dashed line represents the estimated depth estimation error.

The difference in the pose estimation of the closest car in Fig. 14c and Fig. 14d shows the effect of the increase in angular resolution. Nonetheless, it is apparent that, in both cases, our method enables an accurate understanding of the traffic scene in front of the vehicle as an object-based representation.

6.3. Tests on the IVVI 2.0 Platform

Further testing in real traffic situations was performed using the IVVI 2.0 platform, as a way to prove the universality of the algorithm in automotive applications with different sensor devices and setups. In our experiments, we use the narrow baseline of the available trinocular system (12 cm) to maximize the accuracy at short ranges and employ the 1280 × 960 images from the two cameras. For the stereo matching, we use the DispNet option.

According to the results in Sec. 6.1, the detection stage uses a VGG-16 model trained for a scale of 650 pixels in height, 100 proposals, and $N_b = 8$. During inference, a region of interest of 1280 × 650 is extracted from the central area of the left image. Thus, topmost and bottommost rows, which mainly contain sky and road areas and therefore do not convey meaningful information for this branch, are discarded.

Fig. 15 shows four examples of detections and top-view scene models at urban and peri-urban traffic scenarios near our campus. IVVI 2.0 sensing

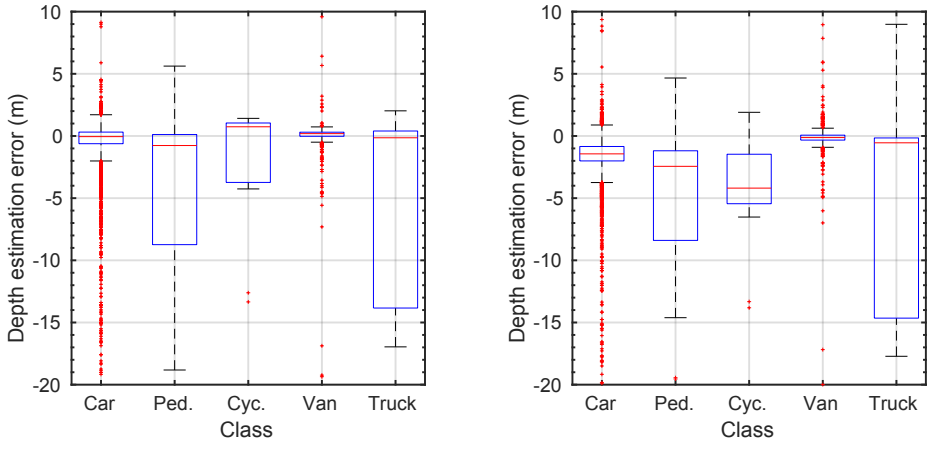


Figure 13: Localization error (detection minus ground-truth) along the depth axis when estimating the center of the object using: (a) our approach; (b) a naive approach. DispNet disparity used.

devices present significantly different properties than the one used for training the model, concerning both the field of view and the sensitivity of the CCD sensor. Despite this, results prove that the method can generalize well.

Regarding the computation times needed for processing the 1280 × 960 image, DispNet stereo matching takes around 80 ms, while the detection on the chosen 1280 × 650 ROI takes less than 100 ms on average. The system is therefore capable of delivering information to the decision-making modules at an acceptable rate of 10 Hz, enabling a fast response to unexpected situations.

7. Conclusion

A computer vision-based framework designed as a step towards a full traffic scene understanding has been presented. Traffic participants are identified by a CNN-based method, showing the potential of this approach within automotive applications. Viewpoint estimation is introduced as an additional inference task to endow the system with further insight into the object features. Conveniently, image recognition is not based on prior knowledge, which makes the method robust to a variety of possible situations.

Because of the nature of the adopted approach, joint object detection and viewpoint estimation can be performed simultaneously over all classes. Since

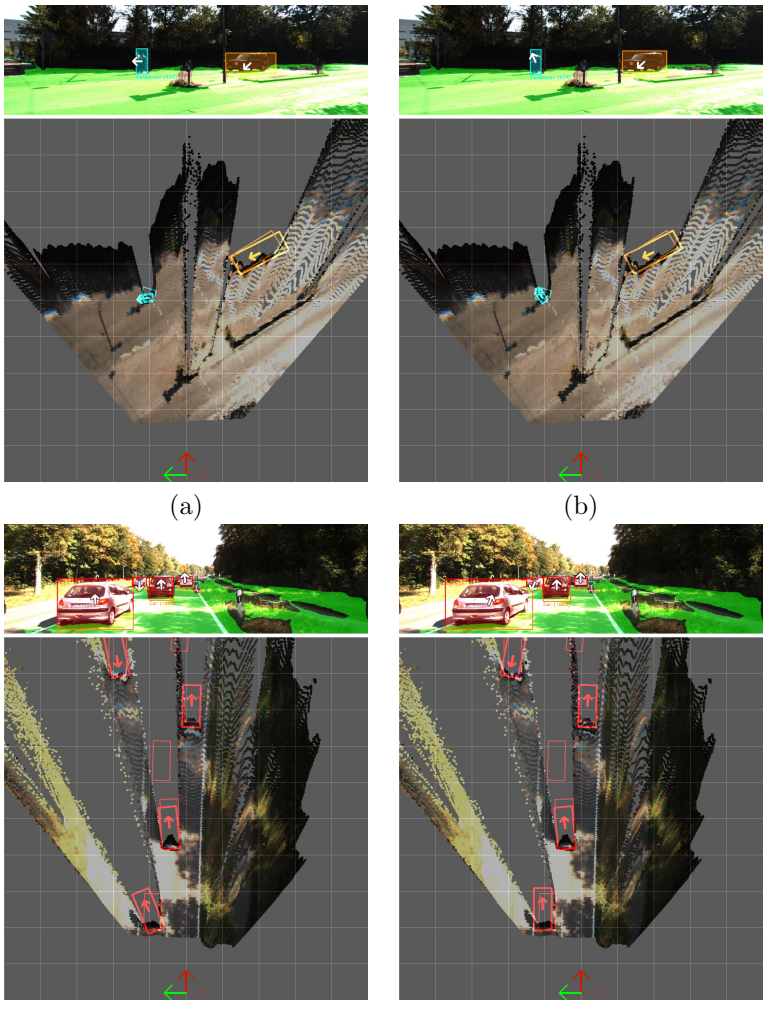


Figure 14: Detections and local scene model for different values of N_b: 8 bins (left) and 16 bins (right). Color code: red for *Car*, blue for *Pedestrian* and orange for *Van*. Grid size in the top-view scene model is 10/3 (3.33) m.

CNN parameters are shared across all tasks and feature vectors computed by the CNN are low-dimensional, computation times are compliant with real-time requirements, yet achieving accurate results. We have also shown that the input scale and the feature extractor architecture are hyperparameters which can be tuned to reach an optimal accuracy/time trade-off according to the requirements of the specific application.

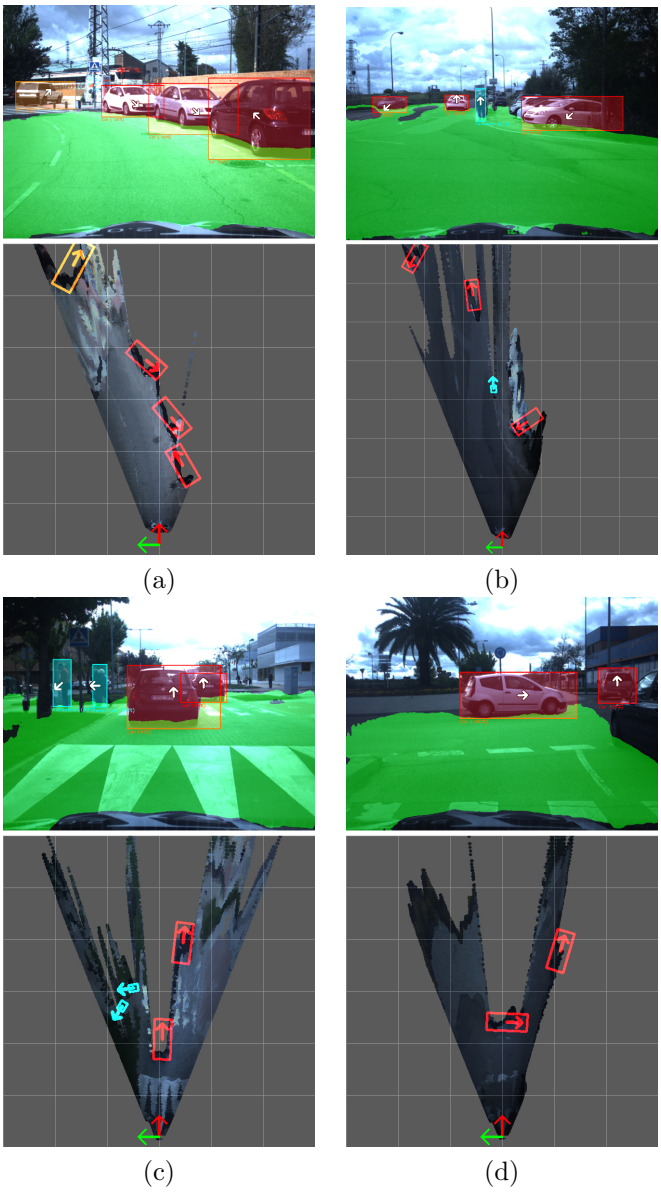


Figure 15: Detections and local scene model in our IVVI 2.0 platform. Color code: red for *Car*, blue for *Pedestrian* and orange for *Van*. Grid size in the top-view scene model is 5 m.

The output from the CNN is merged with information retrieved from a stereo-vision-based 3D reconstruction to gather an accurate situation assessment in complex traffic situations. The accuracy in the depth estimation

has been proven to be adequate for the short-to-medium range. Besides, the modular design of the method makes it possible to employ different depth estimation algorithms. We rely exclusively on visual information, but the proposal serves as a basis for future developments including alternative data sources; e.g., lidar scanners.

Future works may also include new categories of traffic elements, even those belonging to the infrastructure, to enrich the scene model. Semantic segmentation might also be a valuable cue to get a more comprehensive understanding of the different elements of the traffic scene, either as a per-agent semantic mask or as a global estimation of the whole field-of-view. On the other hand, we aim to extend the method to properly handle extreme cases of road slope changes by detecting them beforehand.

However, the primary focus of future development will be on the extension of the method to the time domain to make predictions about future behaviors of agents involved in the scene. In this regard, viewpoint estimation provided by the presented method will play a fundamental role to enable a robust inference.

Acknowledges

This research was supported by the Spanish Government through the CICYT projects (TRA2015-63708-R and TRA2016-78886-C3-1-R), and the Comunidad de Madrid through SEGVAUTO-TRIES (S2013/MIT-2713). We gratefully acknowledge the support of the NVIDIA Corporation with the donation of the GPUs used for this research.

References

References

[1] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, M. Panciroli, A. Prioletti, PROUD-public road urban driverless test: architecture and results, in: Proc. IEEE Intelligent Vehicles Symposium (IV), 2014, pp. 648–654.

[2] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoeppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke,

M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, E. Zeeb, Making Bertha Drive — An Autonomous Journey on a Historic Route, IEEE Intelligent Transportation Systems Magazine 6 (2) (2014) 8–20.

[3] E. Romera, L. M. Bergasa, R. Arroyo, Can we unify monocular detectors for autonomous driving by using the pixel-wise semantic segmentation of CNNs?, in: IEEE Intelligent Vehicles Symposium (IV) - DeepDriving Workshop, 2016.

[4] J. Dai, K. He, J. Sun, Instance-aware Semantic Segmentation via Multi-task Network Cascades, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3150–3158.

[5] H. Zhu, K. V. Yuen, L. Mihaylova, H. Leung, Overview of Environment Perception for Intelligent Vehicles, in: IEEE Transactions on Intelligent Transportation Systems, 2017, pp. 2584 – 2601.

[6] C. Guindel, D. Martín, J. M. Armingol, Modeling Traffic Scenes for Intelligent Vehicles Using CNN-Based Detection and Orientation Estimation, in: ROBOT 2017: Third Iberian Robotics Conference: Volume 2, Springer International Publishing, 2018, pp. 487–498.

[7] I. Markovic, F. Chaumette, I. Petrovic, Moving object detection, tracking and following using an omnidirectional camera on a mobile robot, in: Proc. IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 5630–5635.

[8] D. Olmeda, A. de la Escalera, J. M. Armingol, Contrast invariant features for human detection in far infrared images, in: Proc. IEEE Intelligent Vehicles Symposium (IV), 2012, pp. 117–122.

[9] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, R. G. Herrtwich, Making Bertha See, in: IEEE International Conference on Computer Vision Workshops (ICCVW), 2013, pp. 214–221.

[10] B. Musleh, A. de la Escalera, J. M. Armingol, U-V disparity analysis in urban environments, in: Computer Aided Systems Theory - EUROCAST 2011, Springer Berlin Heidelberg, 2012, pp. 426–432.

[11] H. Badino, U. Franke, R. Mester, Free space computation using stochastic occupancy grids and dynamic programming, in: IEEE International Conference on Computer Vision Workshops (ICCVW), 2007.

[12] F. Oniga, S. Nedevschi, Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection, IEEE Transactions on Vehicular Technology 59 (3) (2010) 1172–1182.

[13] A. Broggi, S. Cattani, M. Patander, M. Sabbatelli, P. Zani, A full-3D Voxel-based Dynamic Obstacle Detection for Urban Scenario using Stereo Vision, in: Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC), 2013, pp. 71–76.

[14] I. Zubiaguirre-Bergen, M. Torres-Torriti, M. Flores-Calero, Generación de Regiones con Potencial de Contener Peatones usando Reconstrucción 3D No Densa a partir de Visión Monocular, Revista Iberoamericana de Automática e Informática Industrial 15 (3) (2018) 243–251.

[15] P. F. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (9) (2010) 1627–1645.

[16] W. Tian, M. Lauer, Fast Cyclist Detection by Cascaded Detector and Geometric Constraint, in: Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC), 2015, pp. 1286–1291.

[17] B. Pepik, M. Stark, P. Gehler, B. Schiele, Teaching 3D geometry to deformable part models, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3362–3369.

[18] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: Proc. Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[19] J. Li, X. Mei, D. Prokhorov, Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene, IEEE Transactions on Neural Networks and Learning Systems 28 (3) (2017) 690–703.

[20] E. Romera, J. M. Álvarez, L. M. Bergasa, R. Arroyo, ERFNet: Efficient Residual Factorized ConvNet for Real-time Semantic Segmentation, IEEE Transactions on Intellent Transportation Systems 19 (1) (2018) 263–272.

[21] J. Žbontar, Y. LeCun, Computing the Stereo Matching Cost with a Convolutional Neural Network, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1592–1599.

[22] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbas, V. Golkov, P. van der Smagt, D. Cremers, Thomas Brox, FlowNet: Learning Optical Flow with Convolutional Networks, in: Proc. IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2758–2766.

[23] F. Yang, W. Choi, Y. Lin, Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2129–2137.

[24] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580–587.

[25] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (6) (2017) 1137–1149.

[26] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: Proc. IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988.

[27] L. Yang, J. Liu, X. Tang, Object detection and viewpoint estimation with auto-masking neural network, in: Computer Vision - ECCV 2014, Springer, Cham, 2014, pp. 441–455.

[28] S. Tulsiani, J. Malik, Viewpoints and Keypoints, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1510–1519.

[29] C. C. Pham, J. W. Jeon, Robust object proposals re-ranking for object detection in autonomous driving using convolutional neural networks, Signal Processing: Image Communication 53 (2017) 110–122.

[30] A. Palazzi, G. Borghi, D. Abati, S. Calderara, R. Cucchiara, Learning to Map Vehicles into Bird's Eye View, in: Proc. International Conference on Image Analysis and Processing (ICIAP), 2017, pp. 233–243.

[31] X. Chen, H. Ma, J. Wan, B. Li, T. Xia, Multi-View 3D Object Detection Network for Autonomous Driving, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6526–6534.

[32] S.-l. Yu, T. Westfechtel, R. Hamada, K. Ohno, S. Tadokoro, Vehicle Detection and Localization on Bird's Eye View Elevation Images Using Convolutional Neural Network, in: Proc. IEEE International Symposium on Safety and Rescue Robotics (SSRR), 2017, pp. 102–109.

[33] D. Martín, F. García, B. Musleh, D. Olmeda, G. A. Peláez, P. Marín, A. Ponz, C. H. Rodríguez Garavito, A. Al-Kaff, A. de la Escalera, J. M. Armingol, IVVI 2.0: An intelligent vehicle based on computational perception, Expert Systems with Applications 41 (17) (2014) 7927–7944.

[34] C. H. Rodríguez Garavito, J. Carmona, A. de la Escalera, J. M. Armingol, Stereo Road Detection Based on Ground Plane, in: Computer Aided Systems Theory - EUROCAST 2015, Springer, Cham, 2015, pp. 748–755.

[35] C. Guindel, D. Martin, J. M. Armingol, Joint object detection and viewpoint estimation using CNN features, in: Proc. IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2017, pp. 145–150.

[36] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, K. Murphy, Speed/accuracy trade-offs for modern convolutional object detectors, in: Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3296–3305.

[37] H. Hirschmüller, Stereo Processing by Semiglobal Matching and Mutual Information, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2) (2008) 328–341.

[38] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, T. Brox, A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4040–4048.

[39] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, Computer Graphics Forum 26 (2) (2007) 214–226.

[40] A. de la Escalera, E. Izquierdo, D. Martín, B. Musleh, F. García, J. M. Armingol, Stereo visual odometry in urban environments based on detecting ground features, Robotics and Autonomous Systems 80 (June) (2016) 1–10.

[41] R. Girshick, Fast R-CNN, in: Proc. IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448.

[42] A. Geiger, P. Lenz, R. Urtasun, Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3354–3361.

[43] O. Faugeras, Three-dimensional computer vision: a geometric viewpoint, The MIT Press, 1993.

[44] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional Architecture for Fast Feature Embedding, in: Proc. ACM International Conference on Multimedia, 2014, pp. 675–678.

[45] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Computer Vision - ECCV 2014, Springer International Publishing, 2014, pp. 818–833.

[46] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR abs/1409.1 (2014).

[47] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv:1704.04861 [cs.CV] (2017).

[48] Y. Xiang, W. Choi, Y. Lin, S. Savarese, Subcategory-aware Convolutional Neural Networks for Object Detection, in: Proc. IEEE Winter Conference on Applications of Computer Vision (WACV), 2017, pp. 924–933.

[49] M. Braun, Qing Rao, Y. Wang, F. Flohr, Pose-RCNN: Joint object detection and pose estimation using 3D object proposals, in: Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 1546–1551.