ON THE FUSION OF CAMERA AND LIDAR FOR 3D OBJECT DETECTION AND CLASSIFICATION

N. Kozonek, N. Zeller, H. Bock, M. Pfeifle

Visteon Electronics Germany, 76227 Karlsruhe, Germany - (nkozone2, nzeller, hbock, mpfeile)@visteon.com

ICWG II/III: Pattern Analysis in Remote Sensing

KEY WORDS: autonomous driving, object detection, classification, Lidar, camera, sensor fusion

ABSTRACT:

In this paper we present a sensor fusion framework for the detection and classification of objects in autonomous driving applications. The presented method uses a state-of-the-art convolutional neural network (CNN) to detect and classify object from RGB images. The 2D bounding boxes calculated by the CNN are fused with the 3D point cloud measured by Lidar sensors. An accurate sensor cross-calibration is used to map the Lidar points into the image, where they are assigned to the 2D bounding boxes. A one-dimensional K-means algorithm is applied to separate object points from foreground and background and to calculated accurate 3D centroids for all detected objects. The proposed algorithm is tested based on real world data and shows a stable and reliable object detection and centroid estimation in different kind of situations.

1. INTRODUCTION

Save navigation of autonomous vehicles in urban and highway scenarios can only be guaranteed by providing an accurate and reliable environmental model. This model must contain detailed information about 3D position and motion of all traffic participants. Most autonomous vehicles are equipped with a front facing camera and 3D sensors to perceive information about their environment. This offers the possibility to exploit the advantage of different sensor characteristics to achieve an accurate and reliable detection of 3D objects.

With the rapid development of deep learning and convolutional neural network (CNN) in the past years, 2D object recognition from camera images has seen significant progress. Especially for the task of detection and classification of vehicles, pedestrians or bicyclists, image based deep learning approaches outclass algorithms based on other sensors like Radar or Lidar, in both accuracy and speed (Li et al., 2017, Kim et al., 2017). This is mainly due to the high resolution and uniform sampling of camera images, as well as the feasibility to perceive chromatic channels. Furthermore, the ability to solve complex object detection tasks, makes CNNs superior to traditional computer vision techniques, that are often not robust due to varying road scenes and lightning conditions. However, one major issue with image based object detection is that a single camera is not able to provide precise 3D locations and 3D dimensions of the detected objects, but rather predicts 2D bounding boxes or provides pixel-wise segmentation. Even if we assume a perfect camera calibration matrix which defines the projection from image space to the road surface, this mapping will fail if the ground-plane changes (e.g. ego vehicle brakes, accelerates or on a sloping uphill).

On the other hand, Lidar sensors supply very accurate and reliable 3D measurements and therefore are suitable for accurate 3D obstacle detection. Although, mainly due to the low ground sampling distance, as well as the missing ability to perceive colors, Lidar based object boundary detection and classification are less reliable than camera based approaches. This is especially the case in distant, sparsely sampled regions of the Lidar point cloud, as well as for small objects which consist of only few sampling points.

In this article, we present an approach which combines the advantages of both, 2D object detection from chromatic images using deep learning techniques and accurate 3D measurements from Lidars, in order to obtain an accurate 3D object detection.

1.1 Related Work

In the past years, several approaches have been proposed to produce accurate and reliable object detection from 3D Lidar point clouds. Some traditional approaches use clustering algorithms to segment the data and assign the resulting group into different classes (Douillard et al., 2011, Mertz et al., 2013). With the rise of deep learning techniques, convolutional neural network (CNN) have been widely used to tackle the task of 3D object detection. However, due to the high dimensionality and sparsity of 3D point clouds, applying deep learning techniques to 3D point clouds implies high computational burden. Therefore, most of the deep learning approaches transform the 3D point cloud into an equivalent 2D representation, on which then a 2D CNN is applied. (Li et al., 2016) projects the 3D Lidar point cloud onto a camera's perspective and then applies a 2D full convolutional network to this representation in order to generate 3D detections. Other lines of work exploit a bird eye view representation from the Lidar point cloud and apply their CNN on this representation (Li, 2016). However, these approaches suffer especially at long range and when dealing with occluded objects due to the sparsity of the Lidar points in these regions (Liang et al., 2018).

Camera images on the other hand, are able to provide dense measurements. CNN approaches have been proven to be very effective to perform object detection and have surpassed many traditional methods in both accuracy and speed. Some prominent examples like R-CNN (Girshick et al., 2014) and its variants (Ren et al., 2015) are based on region proposals. These networks consist of two separated parts, one that proposes a

region for possible detections and another part that does classification to these proposed regions. The drawback of this kind of architecture is, that it significantly slows down the whole processing speed. Another example, that overcomes this issue is YOLO (You Only Look Once) (Redmon et al., 2015), which uses one network for both, prediction of potential regions and classification. The image is therefore divided into a coarse grid. For each grid a predefined number of bounding boxes are predicted together with a class probability. Although all of these methods perform well in detecting objects in the image space, 3D localizations of detected objects are not accurate, due to the loss of depth information caused by the perspective projection (Chen et al., 2016a, Chen et al., 2016b).

Recently, many techniques have been explored to exploit both cameras and Lidar jointly. One common approach is to apply 2D deep learning techniques on both camera image and representations of the Lidar point cloud in the bird eye view and fuse the intermediate convolutional feature maps of both (Chen et al., 2016c, Ku et al., 2017). As this is usually done at a coarse level, the accuracy of the finally obtained 3D object decreases. Another line of work performs depth image based processing. The Lidar point cloud is therefore projected in the image space and encoded as an additional image channel (Gupta et al., 2014, Silberman et al., 2012). The fused output can then be passed to any 2D object detection architecture.

Our approach also follows the idea of projecting the Lidar point cloud into the image space, but unlike the methods that encode the depth information from Lidar as an additional image channel, we treat the task of object detection and depth estimation seperately. A CNN is used to localize and classifiy objects from a given RGB image. These 2D bounding boxes are then fused with the projected 3D points using an accurate sensor cross-calibration. Finally, the fused output is clustered, and a centroid of the 3D object is calculated. This centroid then can be used as an input for an object tracking algorithm.

1.2 Notations

In the following, we denote vectors by bold, lower-case letters $\boldsymbol{x}$ and matrices by bold, upper case letters $\boldsymbol{G}$. Scalars are represented by normal letters λ which may be either lower or upper case.

2. THE OBJECT DETECTION SYSTEM

2.1 System Overview

In addition to the algorithm proposed in this article, we present the entire process chain for the accurate cross-calibration of a front facing camera with a collection of Lidar sensors based on a simple checkerboard pattern. The calibration process, as well as the entire detection framework can easily be extended to multiple cameras and Lidars to obtain a full 360 degrees coverage. Figure 1 shows an overview of the proposed object detection and classification algorithm.

In the proposed algorithm a CNN is used to perform the task of object detection on a recorded image. CNNs have been proven to be very effective in the area of image recognition and classification. A CNN consists of an input and output layer, as well as at least one hidden layer in between. The hidden layers in this type of neural network are convolutional layers, that consist of a set of learnable filters and are able to extract features from

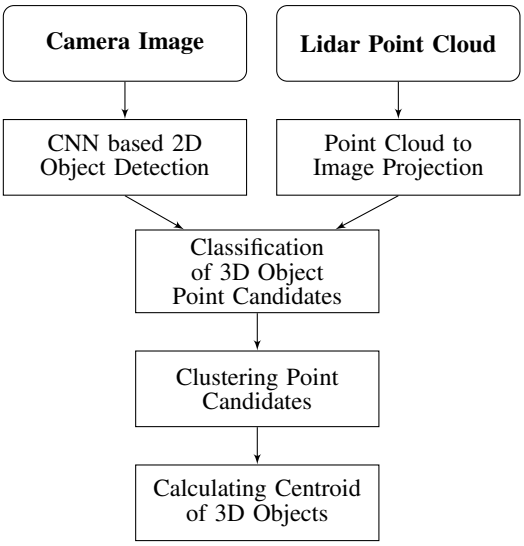


Figure 1. Flowchart showing the processing chain of the proposed algorithm.

an input volume. In the first layer, usually low-level features, such as edges, color, texture, et cetera are extracted and learned from an image input. By adding more convolutional layers, the network is able to also capture some high-level feature representations and finding even more sophisticated patterns. For the task of object detection, these learned features are finally fed into one or more fully connected output layers, to obtain the final result. They all have in common that the network propose several bounding boxes in the image and predicts if any of them actually belongs to an object. In this way, the network is able to detect multiple objects within an image.

For projecting the obtained Lidar point cloud from 3D space into the 2D image space of the camera, an accurate sensor cross-calibration between camera and Lidar is used, that will be explained in 2.2. The 2D bounding boxes predicted by the CNN are used to select potential 3D object point candidates for each of the detected objects. All Lidar points that are projected inside a predicted 2D bounding box in the image space are therefore selected as candidates to potentially contribute as 3D points to a certain object. However, the rectangular bounding boxes, that can be obtained from the CNN are in general larger than the actual object. As a result, the group of points that are assigned to a certain object, do not only contain all desired object corresponding points, but furthermore also foreground and background points.

To distinguish between actual object points, and foreground or background points, the 3D point candidates are clustered based on their spatial distribution. The clustering of points is not done in the 3D space, but in a 1D space, i.e. the z-domain in the camera frame. A K-means algorithm (MacQueen, 1967) with a predefined number of three clusters is used for clustering. These three clusters can be considered as foreground (e.g. a tree or pole occluding the object), object and background. While clustering along only one dimension is much faster than clustering in the 3D space, one can also overcome the problem of multiple separated clusters, e.g. due to occlusions, which in fact correspond to the same object. For instance, a truck or a car standing behind a tree would be represented by at least two clusters if the points would be clustered in the 3D space. However, if they are clustered in a 1D z-domain they will be represented by one cluster only. As we assume, that the object occupies most space in

the bounding box, all points corresponding to the largest cluster are classified to actually represent the object.

Algorithms that perform tracking of moving objects, generally track the centroid of the object or rather the centroid of the facing side e.g. of a car. Such a centroid can be easily tracked over time, for instance, by an extended or unscented Kalman filter (Wan , Van Der Merwe, 2000). Due to that reason, the presented approach also focuses on the calculation of a reliable centroid of an object. From the 2D bounding boxes in image space, it is already possible to obtain a reliable estimation of the x- and y-dimension of the objects centroid. However, as a reliable distance estimation is also key for an accurate tracking of moving objects, we are using the clustered points from the Lidar to make a more precise estimation of the missing dimension. Apart from that, the framework can easily be extended to 3D object boundaries calculated from the Lidar point cloud.

2.2 System Calibration

The accuracy of the presented fusion approach highly depends on an accurate calibration of the sensors. Therefore, we present a straight-forward approach for the intrinsic camera calibration and the cross-calibration between camera and Lidar.

2.2.1 Intrinsic Camera Calibration The intrinsic camera parameters are calibrated following the well-known method of Zhang (Zhang, 2000). Based on multiple recordings of a planar checkerboard pattern, the intrinsic matrix $\boldsymbol{K}$ (see eq. (1)) as well as radial symmetric and tangential distortion (eqs. (2) to (6)) parameters are solved.

$$\lambda \begin{bmatrix} x_I \\ y_I \\ 1 \end{bmatrix} = \boldsymbol{K} \cdot \boldsymbol{x}_C = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} \tag{1}$$

$$\begin{aligned} \tilde{x}_d &= \tilde{x}(1 + k_0 r^2 + k_1 r^4 + k_2 r^6) \\ &\quad + p_0(r^2 + 2\tilde{x}^2) + 2p_1 \tilde{x}\tilde{y} \end{aligned} \tag{2}$$

$$\begin{aligned} \tilde{y}_d &= \tilde{y}(1 + k_0 r^2 + k_1 r^4 + k_2 r^6) \\ &\quad + p_1(r^2 + 2\tilde{y}^2) + 2p_0 \tilde{x}\tilde{y} \end{aligned} \tag{3}$$

$$\tilde{x} = \frac{x_I - c_x}{f_x} = \frac{x_C}{z_C} \quad \tilde{y} = \frac{y_I - c_y}{f_y} = \frac{y_C}{z_C} \tag{4}$$

$$r^2 = \tilde{x}^2 + \tilde{y}^2 \tag{5}$$

$$x_{Id} = \tilde{x}_d \cdot f_x + c_x \quad y_{Id} = \tilde{y}_d \cdot f_y + c_y \tag{6}$$

For later use, the image coordinates are directly undistorted. This allows to work with an ideal pinhole camera model as defined in eq. (1) throughout the entire system. Furthermore, the undistorted image can be obtained by using a static remapping, which can be solved by a simple look-up-table in combination with an interpolation step.

2.2.2 Camera-to-Lidar Cross-Calibration For the camera-to-Lidar cross-calibration we follow a semi-automated approach. As for the intrinsic calibration of the camera, we are using a checkerboard as a calibration target and align it in a way, that it can be seen by Lidar and camera at the same time. We are recording the camera image as well as the Lidar point cloud. Afterwards, the four corner points of the checkerboard are selected manually in the camera image (using intensity differences) and in the Lidar point cloud (using depth differences). The detection of the checkerboard corners, could be also done automatically. The relationship between the checkerboard corners detected in the recorded image with coordinates (x_I, x_I) and the corresponding metric checkerboard coordinates $(x_{cb}, x_{cb}, z_{cb} = 0)$ is defined by the combination of a rigid body transformation and the intrinsic camera matrix as follows:

$$\lambda \begin{bmatrix} x_I \\ y_I \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{t} \end{bmatrix} \cdot \begin{bmatrix} x_{cb} \\ y_{cb} \\ z_{cb} \\ 1 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix} \in \mathrm{SO}(3) \quad \text{and} \quad \boldsymbol{t} \in \mathbb{R}^3 \tag{7}$$

The rigid body transform from 2D checkerboard coordinates $\boldsymbol{x}_{cb}$ to normalized image coordinates $\tilde{\boldsymbol{x}}$ results in a so-called planar homography defined by the matrix $\boldsymbol{H}$, as given in eq. (9).

$$\begin{aligned} \lambda \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} &= \lambda \boldsymbol{K}^{-1} \begin{bmatrix} x_I \\ y_I \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{t} \end{bmatrix} \cdot \begin{bmatrix} x_{cb} \\ y_{cb} \\ z_{cb} = 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \cdot \begin{bmatrix} x_{cb} \\ y_{cb} \\ 1 \end{bmatrix} \end{aligned} \tag{8}$$

$$\lambda' \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} = \boldsymbol{H} \cdot \begin{bmatrix} x_{cb} \\ y_{cb} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \boldsymbol{h}_3 \end{bmatrix} \cdot \begin{bmatrix} x_{cb} \\ y_{cb} \\ 1 \end{bmatrix} \tag{9}$$

Up to an ambiguous scaling factor, the matrix $\boldsymbol{H}$ can be solved from at least four points. Instead of using only four points, we solve the homography in a least square sense from a total number of N detected checkerboard points (corners between adjacent checkerboard fields). However, what one is interested in, is not the homography itself, but the underlying rigid body transform instead. An estimation $\hat{\mu}$ of the missing scaling factor can be obtained as follows:

$$\hat{\mu} = \frac{2}{\|\boldsymbol{h}_1\| + \|\boldsymbol{h}_2\|} \tag{10}$$

$$\tag{11}$$

This is since for a rotation matrix $\|\boldsymbol{r}_1\| = \|\boldsymbol{r}_2\| = \|\boldsymbol{r}_3\| = 1$ holds by definition. Another definition of a rotation matrix is that all three column vector $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$ are orthogonal to each other. Hence $\boldsymbol{r}_3$ can be calculated from $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ as follows:

$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2 \tag{12}$$

From the least squares estimate of the matrix $\boldsymbol{H}$, one can not guarantee that $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ will be orthogonal to each other. Therefore the estimated rotation matrix $\boldsymbol{R}$ has to be orthonormalized based on its sigular value decomposition (SVD) as given in eqs. (13) and (14).

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T \tag{13}$$

$$\hat{\boldsymbol{R}} = \boldsymbol{U}\boldsymbol{V}^T \tag{14}$$

Here, $\boldsymbol{D}$ represents the diagonal matrix with the singular values of $\boldsymbol{R}$. This results in the estimate $\hat{\boldsymbol{M}}$ of the rigid body transfor-

mation between checkerboard and camera coordinates $\boldsymbol{x}_C$.

$$\hat{\boldsymbol{M}} = \begin{bmatrix} \hat{\boldsymbol{R}} & \hat{\boldsymbol{t}} \\ \boldsymbol{0} & 1 \end{bmatrix} \in \mathrm{SE}(3) \quad \text{with} \quad \hat{\boldsymbol{t}} = \hat{\mu} \cdot \boldsymbol{h}_3 \tag{15}$$

Based on the matrix $\hat{\boldsymbol{M}}$, one obtains the checkerboard plane in camera coordinates $\boldsymbol{x}_C$ as given in eq. (16).

$$\boldsymbol{n} = \hat{\boldsymbol{R}} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \boldsymbol{s} = \hat{\boldsymbol{t}} \tag{16}$$

Here, $\boldsymbol{n}$ defined the planes normal vector, while $\boldsymbol{s}$ is the respective support vector.

Using the plane defined by $\boldsymbol{n}$ and $\boldsymbol{s}$ in camera coordinates one is able to calculate the 3D corners of the calibration target by calculating the intersection between the detected image points and the defined plane. For later use, we will denote these four corner points by the set $X_C^{cb} = \{\boldsymbol{x}_C^{cb(0)}, \cdots, \boldsymbol{x}_C^{cb(3)}\}$.

While the selection of the four corner points of the checkerboard in the image is straight forward it is more difficult in the Lidar point cloud. From the recorded Lidar point cloud we calculate an orthogonal 2D projection onto the y-z-plane. In this projection the 2D corners of the calibration target are selected by finding the respective depth continuities. Based on all Lidar points lying within the selected rectangular region, the 3D target plane is estimated while removing outliers from the set of points using singular value decomposition. Finally the set of calibration target corners $X_L^{cb} = \{\boldsymbol{x}_L^{cb(0)}, \cdots, \boldsymbol{x}_L^{cb(3)}\}$ in the Lidar frame is obtained by projecting the selected 2D points onto the estimated plane.

The rigid body transformation between both sets of points X_C^{cb} and X_L^{cb} can be obtained in a closed form. The respective mean values as well as the estimation of the cross-covariance matrix is calculated as follows:

$$\overline{x}_C^{cb} = \frac{1}{4} \cdot \sum_{i=0}^{3} x_C^{(i)} \tag{17}$$

$$\overline{x}_L^{cb} = \frac{1}{4} \cdot \sum_{i=0}^{3} x_L^{(i)} \tag{18}$$

$$\boldsymbol{C} = \sum_{i=0}^{3} \left(x_L^{(i)} - x_L\right) \cdot \left(x_C^{(i)} - x_C\right)^T \tag{19}$$

From the singular value decomposition of $\boldsymbol{C}$ one obtains the rotation matrix from Lidar to camera frame as follows:

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T \rightarrow \boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^T \tag{20}$$

In case that $\boldsymbol{R}$ has a negativ determinate it has to be recalculated as follows:

$$\tilde{\boldsymbol{V}} = \boldsymbol{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \rightarrow \boldsymbol{R} = \tilde{\boldsymbol{V}}\boldsymbol{U}^T \tag{21}$$

Based on the rotation matrix $\boldsymbol{R}$ and the means $\overline{x}_C$ and $\overline{x}_L$ the translation vector t is obtained as follows:

$$t = \overline{x}_C - \boldsymbol{R} \cdot \overline{x}_L \tag{22}$$

In fact, the system can easily be extended multiple Lidar sensors and cameras. While multiple Lidars can be cross-calibration based on standard interative closest points (ICP) approaches, the camera-to-camera cross-calibration can be obtained from the homography between two views.

2.3 2D Object Detection

For the task of 2D object detection in the recorded image, each CNN with the purpose of object detection can be used. We decided to use the so called YOLO (You Only Look Once) network, which was first introduced in 2015. YOLO is capable of super real-time object detection, obtaining a maximum of 45 FPS on a GPU. For each image, it predicts spatially seperated bounding boxes in the image space and additionally outputs an associated class probability for each of the predicted bounding boxes. The model is trained on the Berkely driving dataset (Yu et al., 2018), which consists of a total of 100K images and contains 10 classes (e.g. car, bike, bus, motor, person, traffic sign/light), that are all relevant in the field of autonomous driving application. Before we can start to train the model, some preliminary work has to be done. This involves resizing the image (according to the input shape accepted by the model) and converting the ground truth labels from the dataset into an appropriate form. As it is usually hard to train a model from the scratch, we make also use of transfer learning. Transfer learning is the process of taking a pre-trained model (that was already trained on a large dataset) and re-use some of its first layers. As these layers usually learn general features like edges or curves, they can even be taken from a model that was trained for a complete different purpose. The trained model is afterwards evaluated on the validation dataset with the proposed method (mean average presicion) of Berkely driving dataset.

2.4 3D Object Modeling

2.4.1 World to Image Projection Based on a proper sensor cross-calibration as described in Section 2.2 the recorded 3D point cloud measured by a Lidar sensor is projected into the image space. For any Lidar point $\boldsymbol{x}_L = [x_L, y_L, z_L]^T$ this projection is defined as follows:

$$\lambda \begin{bmatrix} x_I \\ y_I \\ 1 \end{bmatrix} = \boldsymbol{K} \cdot \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \cdot \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix} \tag{23}$$

Here $[\boldsymbol{R}\ \boldsymbol{t}]$ defines the rigid body transform from Lidar to camera frame, obtained from the cross-calibration.

2.4.2 3D Object Point Candidate Selection The Lidar points, that have been projected into the image space are fused with the information obtained by the CNN. In order to select the 3D object points that belong to a particular object, all Lidar points that are inside a certain bounding box are treated as potential candidates. As these bounding boxes are rectangular, but the detected objects usually have a different shape, a bounding box spans over a bigger volume than the object. Therefore, some of the points, that were beforehand selected to be potentially belonging to the object, actually represent foreground or background.

2.4.3 3D Object Segmentation In autonomous driving application, one is generally interested in the centroid of the object, as this can then serve as an input for tracking algorithms. Hence, it is desirable to exclude all points corresponding to foreground or background in order to calculate the centroid of the object. To obtain all points which really correspond to the detected object, the 3D point candidates are clustered based on their spatial distribution. From the 2D bounding boxes in the image space, it is already possible to obtain a reliable estimation of the x- and y-dimension of the objects centroid. Hence, instead of doing a complete 3D clustering of points, we decided to apply the clustering only in a 1D space, i.e. the z-domain of the camera frame. Aside the fact, that 1D clustering is much faster than clustering in the 3D space, we also overcome the problem of dealing with multiple seperated clusters of one object, due to occlusion. As we assume the object to occupy most of the space inside the 2D bounding box, all points corresponding to the largest cluster are classified to actually represent the object. These object points are thereafter used to estimate the centroid of the object. This allows a much more accurate motion estimation, compared to a centroid that was only calculated based on an 2D detection.

3. EXEMPLARY RESULTS

Finally, the proposed algorithm is applied to sensor recordings from our own test vehicle. Figures 2 to 6 show some exemplary results for different kind of scenarios. It can be seen, that our method offers a stable and reliable object detection and centroid estimation in various situations. Due to the clustering in the 1D space, centroids of objects that are (partly) occluded can still be calculated properly (see Figure 5). However, one drawback is the accuracy of detections at long range distance. Although the camera predicts an object, centroids calculated from Lidar points cannot be estimated, due to the sparsity of the Lidar points (figures 2, 5). Although the accuracy in distance estimation will drop in such cases, we are still be able to detect the object, which makes this approach superior compared to pure Lidar based detections. Additionally, our algorithm simultaneously offers a object classification, that distinguish between 10 classes. A commonly used metric to evaluate the performance of an object detection algorithm is Average Precision (AP) (Wilson et al., 2019). To obtain this metric, the Intersection-over-Union (IoU) between predicted bounding box b_{pred} and ground truth bounding box b_t , which is defined as the overlapping area of the two boxes, is calculated. A detection is only considered to be true positive, if it has an IoU greater than a given threshold. On the validation dataset our trained model achieves the top 5 precision, as stated in table 1.

Class	Car	Truck	Bus	Person	Bike
AP [%]	43.21	39.35	37.01	22.64	17.33

Table 1. Average Precision of the top 5 classes obtained by our trained CNN model. Following the Berkeley Driving Dataset challenge, an AP_{75} is used which correpsonds to a threshold of 0.75.

Please note, that the Berkeley Driving Dataset Challenge, from which we derived our benchmark, applies a relatively strict localization goal of AP_{75} (corresponding to an threshold of 0.75). Therefore, the achieved precisions might seem low compared to results from other object detection challenges. From table 1 it can be derived, that our model generally performs better on larger objects, like cars, trucks or buses. This is due to the limitations of the YOLO architecture. As the input images are being resized to a resolution of 448x448, some of the already small objects are sized down to a very few pixels, which the network is not able to learn anymore. Additionally, the model struggles with small objects that appear as a cluster (e.g. group of people).

4. CONCLUSION

In this paper a sensor fusion framework for 3D object detection in autonomous driving applications was shown. The proposed method uses a state-of-the-art CNN to detect and classify objects from camera images. An accurate sensor-cross-calibration approach is used, to map 3D Lidar points into the image space. The fused points are then assigned to the 2D bounding boxes, that were predicted from the CNN. A 1D K-means algorithm is thereafter applied, to seperate actual objects points from foreground and background points. From the resulting object points, 3D centroids are calculated, which can be used as an input for tracking algorithms. On real data, obtained from one of our test vehicles, the method shows superior detection accuracy compared to pure Lidar based algorithms. By fusing depth information from the Lidar, with the detections obtained by the camera, we can also overcome the loss of depth information caused by the perspective projection of the camera. Our approach therefore offers reliable and accurate distance and position information from objects, that is necessary for a precise motion estimation. Apart from that, our algorithm simultaneously offers a object classification, that distinguish between 10 classes. Due to the CNN architecture that we used, the model generally performs better on larger objects. The performance can be easily enhanced, by using another CNN architecture, e.g. YOLOv3 (Redmon , Farhadi, 2018) or SqueezeDet (Wu et al., 2016). Furthermore, the framework can easily be extended to multiple cameras and Lidars to obtain a full 360 degrees coverage. For a future work our method could be extended, by fusing 3D boundaries calculated from the Lidar point cloud into the image space and estimate 3D bounding boxes. Another approach is to do semantic or instance segmentation instead of object detection, which would make the clusterer redundant.

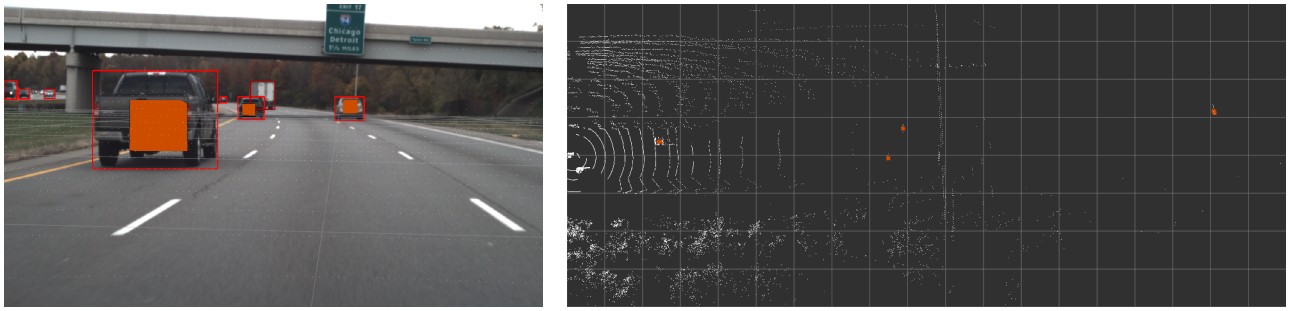
REFERENCES

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R., 2016a. Monocular 3D Object Detection for Autonomous Driving. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2147-2156.

Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S., Urtasun, R., 2016b. 3D Object Proposals Using Stereo Imagery for Accurate Object Class Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP.

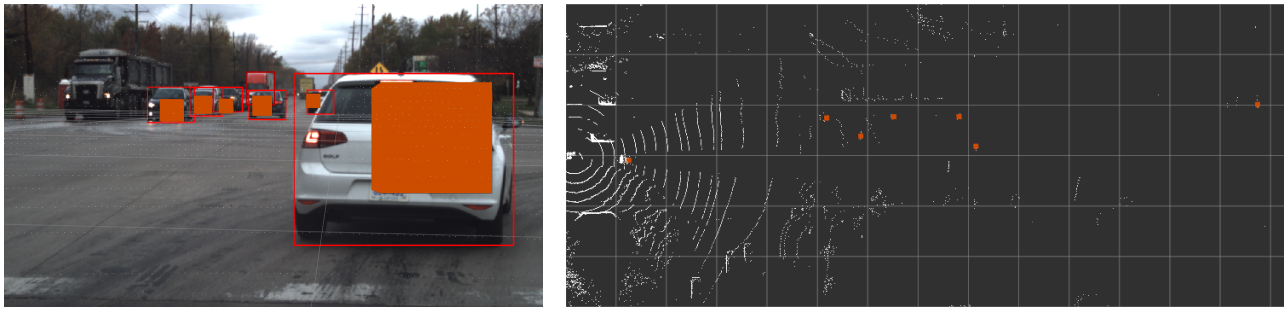
Chen, X., Ma, H., Wan, J., Li, B., Xia, T., 2016c. Multi-View 3D Object Detection Network for Autonomous Driving. *CoRR*, abs/1611.07759.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., Frenkel, A., 2011. On the segmentation of 3d lidar point clouds. *2011 IEEE International Conference on Robotics and Automation*, 2798–2805.



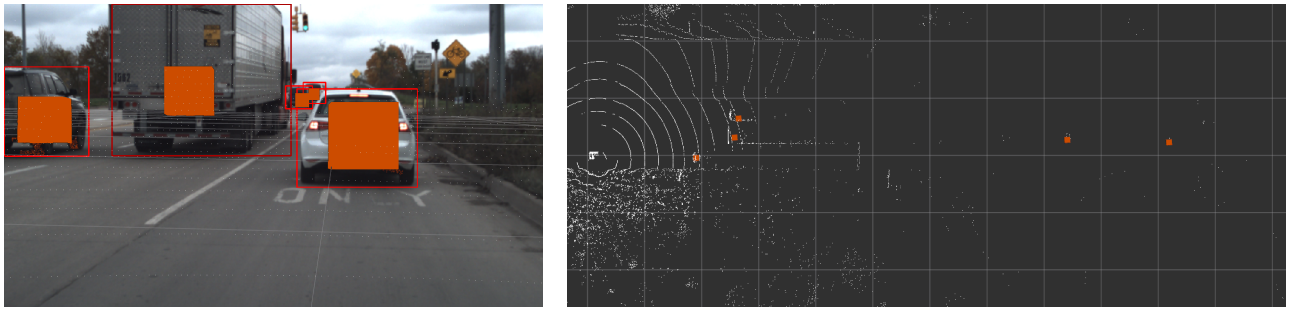
(a) camera image (b) point cloud top view

Figure 2. Object detection results on a highway scenario. (a) Camera image overlayed with bounding boxes calculated by the CNN (red rectangles around vehicles) and 3D centroids projected into the image (orange cubes). (b) Top view of the Lidar point cloud (white dots) with calculated object centroids (orange cubes). Ego vehicle is located at the left boundary of the figure. Grid cells have a size of 10 by 10 meters. Centroid of the truck in a distance ob about 170 m is not shown in the image.



(a) camera image (b) point cloud top view

Figure 3. Object detection results on a junction. (a) Camera image overlayed with bounding boxes calculated by the CNN (red rectangles around vehicles) and 3D centroids projected into the image (orange cubes). (b) Top view of the Lidar point cloud (white dots) with calculated object centroids (orange cubes). Ego vehicle is located at the left boundary of the figure. Grid cells have a size of 10 by 10 meters.



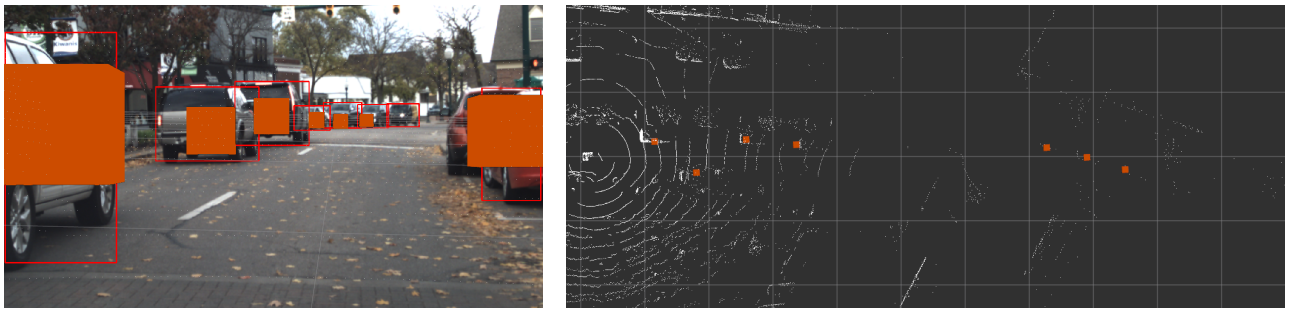
(a) camera image (b) point cloud top view

Figure 4. Object detection results in heavy traffic. (a) Camera image overlayed with bounding boxes calculated by the CNN (red rectangles around vehicles) and 3D centroids projected into the image (orange cubes). (b) Top view of the Lidar point cloud (white dots) with calculated object centroids (orange cubes). Ego vehicle is located at the left boundary of the figure. Grid cells have a size of 10 by 10 meters. Even the two cars between the truck and the leading vehicle in a distance of about 80 m and 100 m are detected correctly.

This contribution has been peer-reviewed.
https://doi.org/10.5194/isprs-archives-XLII-2-W16-149-2019 | © Authors 2019. CC BY 4.0 License.

(a) camera image (b) point cloud top view

Figure 5. Object detection results for an urban scenario. (a) Camera image overlayed with bounding boxes calculated by the CNN (red rectangles around vehicles) and 3D centroids projected into the image (orange cubes). (b) Top view of the Lidar point cloud (white dots) with calculated object centroids (orange cubes). Ego vehicle is located at the left boundary of the figure. Grid cells have a size of 10 by 10 meters. Due to the K-means clustering the centroid of the car on the left occluded by a tree is still calculated correctly. For the far distant car detected by the CNN no centroid was calculated since it is confirmed by only little number of Lidar points.



(a) camera image (b) point cloud top view

Figure 6. Object detection results for a challenging urban scenario. (a) Camera image overlayed with bounding boxes calculated by the CNN (red rectangles around vehicles) and 3D centroids projected into the image (orange cubes). (b) Top view of the Lidar point cloud (white dots) with calculated object centroids (orange cubes). Ego vehicle is located at the left boundary of the figure. Grid cells have a size of 10 by 10 meters. For the most distant car detected by the CNN no centroid was calculated since it is confirmed by only little number of Lidar points.

https://doi.org/10.5194/isprs-archives-XLII-2-W16-149-2019

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, IEEE Computer Society, Washington, DC, USA, 580–587.

Gupta, S., Girshick, R. B., Arbelaez, P., Malik, J., 2014. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. *CoRR*, abs/1407.5736.

Kim, J., Kim, J., Jang, G. J., Lee, M., 2017. Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection. *Neural networks : the official journal of the International Neural Network Society*, 87, 109-121.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S. Lake, 2017. Joint 3D Proposal Generation and Object Detection from View Aggregation. *CoRR*, abs/1712.02294.

Li, B., 2016. 3D Fully Convolutional Network for Vehicle Detection in Point Cloud. *CoRR*, abs/1611.08069.

Li, B., Zhang, T., Xia, T., 2016. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. *CoRR*, abs/1608.07916.

Li, J., Mei, X., Prokhorov, D., Tao, D., 2017. Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene. *IEEE Transactions on Neural Networks and Learning Systems*, 28, 690-703.

Liang, M., Yang, B., Wang, S., Urtasun, R., 2018. Deep continuous fusion for multi-sensor 3d object detection. V. Ferrari (ed.), *Computer Vision – ECCV 2018*, Springer International Publishing, 663–678.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 281–297.

Mertz, C., Navarro-Serment, L. E., MacLachlan, R., Rybski, P., Steinfeld, A., Suppe, A., Urmson, C., Vandapel, N., Hebert, M., Thorpe, C., Duggins, D., Gowdy, J., 2013. Moving Object Detection with Laser Scanners. *J. Field Robot.*, 30, 17-43.

Redmon, J., Divvala, S. K., Girshick, R. B., Farhadi, Ali, 2015. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, abs/1506.02640.

Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.02767.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems 28*, 91–99.

Silberman, N., Hoiem, D., Kohli, P., Fergus, R., 2012. Indoor segmentation and support inference from rgbd images. *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, ECCV'12, Springer-Verlag, Berlin, Heidelberg, 746–760.

Wan, E. A., Van Der Merwe, R., 2000. The unscented kalman filter for nonlinear estimation. *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 153–158.

Wilson, B., Hoffman, J., Morgenstern, J., 2019. Predictive Inequity in Object Detection. *CoRR*, abs/1902.11097.

Wu, B., Iandola, F. N., Jin, P. H., Keutzer, K., 2016. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. *CoRR*, abs/1612.01051.

Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., Darrell, T., 2018. BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling. *CoRR*, abs/1805.04687.

Zhang, Z., 2000. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 1330-1334. https://www.microsoft.com/en-us/research/publication/a-flexible-new-technique-for-camera-calibration/. MSR-TR-98-71, Updated March 25, 1999.