3D Detection of Vehicles from 2D Images in Traffic Surveillance

M. H. Zwemer[1,2][a], D. Scholte[1,2], R. G. J. Wijnhoven[2][b] and P. H. N. de With[1][c]
[1]Department of Electrical Engineering, Eindhoven University, Eindhoven, The Netherlands
[2]ViNotion BV, Eindhoven, The Netherlands

Keywords: 3D Object Detection, Traffic Surveillance, Vehicle Detection.

Abstract: Traffic surveillance systems use monocular cameras and automatic visual algorithms to locate and observe traffic movement. Object detection results in 2D object boxes around vehicles, which relate to inaccurate real-world locations. In this paper, we employ the existing KM3D CNN-based 3D detection model, which directly estimates 3D boxes around vehicles in the camera image. However, the KM3D model has only been applied earlier in autonomous driving use cases with different camera viewpoints. However, 3D annotation datasets are not available for traffic surveillance, requiring the construction of a new dataset for training the 3D detector. We propose and validate four different annotation configurations that generate 3D box annotations using only camera calibration, scene information (static vanishing points) and existing 2D annotations. Our novel Simple box method does not require segmentation of vehicles and provides a more simple 3D box construction, which assumes a fixed predefined vehicle width. The Simple box pipeline provides the best 3D object detection results, resulting in 51.9% AP3D using KM3D trained on this data. The 3D object detector can estimate an accurate 3D box up to a distance of 125 meters from the camera, with a median middle point mean error of only 0.5-1.0 meter.

1 INTRODUCTION

Vehicle traffic congestion is a major problem worldwide. Traffic surveillance systems can help to improve traffic management in the city and ensure road and traffic safety. Such traffic surveillance systems commonly use networks of monocular cameras and employ computer vision algorithms to observe traffic movement. These camera systems employ object detection and tracking to collect statistics such as the number of vehicles per traffic lane, the type of vehicles and the traffic flow.

Typical traffic analysis systems for surveillance cameras use an object detector to localize vehicles followed by a tracking algorithm to follow vehicles over time. Object locations in computer vision are typically represented by a 2D box fully capturing the object. The bottom midpoint of the box is then used to define the vehicle position in a single location (see Figure 1). The ground surface of the object is thereby represented in a single point only. Using the camera calibration, this pixel-position point can be converted

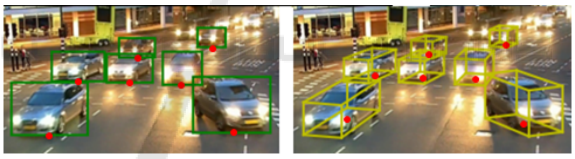


Figure 1: Localization using 2D and 3D bounding boxes. Note the difference between the estimated ground-plane location (red dot) for both bounding boxes.

to a real-world (GPS) location for further use in the traffic application. A more accurate detection method would represent the object location by a 3D bounding box such that the complete ground plane is estimated in contrast to the current single point. The use of more accurate 3D location estimation enables more accurate measurements of vehicle speed, size and inter-vehicle distances.

In this paper, we propose to use a 3D bounding-box detector on monocular fixed-camera images. We assume that the cameras are fully calibrated such that the camera locations can be converted to 3D-world positions. Detectors that directly estimate 3D boxes (Li, 2020) from 2D images, require 3D-labeled training datasets, which are not available for our traffic surveillance application. Available traffic surveillance datasets lack camera calibrations

[a] https://orcid.org/0000-0003-0835-202X
[b] https://orcid.org/0000-0001-8837-3976
[c] https://orcid.org/0000-0002-7639-7716

Zwemer, M., Scholte, D., Wijnhoven, R. and de With, P.
3D Detection of Vehicles from 2D Images in Traffic Surveillance.
DOI: 10.5220/0010783600003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 5: VISAPP*, pages 97-106
ISBN: 978-989-758-555-5; ISSN: 2184-4321

and 3D box annotations. Such 3D annotations are available in datasets for autonomous driving such as KITTI (Geiger et al., 2012), but the in-car camera viewpoints are different from typical road-side surveillance viewpoints. We propose a novel annotation processing chain that converts existing 2D box labels to 3D boxes using labeled scene information. This labeled scene information is derived from 3D geometry aspects of the scene, such as the vanishing points per region and the estimated vanishing points per vehicle depending on the vehicle orientation. The use of vanishing points combined with camera calibration parameters enables to derive 3D boxes capturing the car geometry in a realistic setting. For validating the concept, the existing KM3D (Li, 2020) CNN object detector is evaluated on our automatically annotated traffic surveillance dataset to investigate the effect of different processing configurations, in order to optimize the system into a final pipeline.

The remainder of this paper is structured as follows. First, a literature overview is given in Section 2. Second, the 3D box annotation processing chain is elaborated in Section 3. The results of the experiments are presented in Section 4 where also the optimal pipeline is determined. Section 5 discusses the conclusions.

2 RELATED WORK

There are two categories of 3D object detectors for monocular camera images: (A) employing geometric constraints and labeled scene information and (B) direct estimation of the 3D box from a single 2D image.
A. Geometric Methods. In (Dubská et al., 2014), the authors propose an automatic camera calibration from vehicles in video. This calibration and vanishing points are then used to convert vehicle masks obtained by background modeling to 3D boxes. They assume a single set of vanishing points per scene, as the road surface in the scenes are straight and the camera position is stationary. The authors of (Sochor et al., 2018) improve vehicle classification using information from 3D boxes. Because their method works on static 2D images and motion information is absent to estimate the object orientation, they propose a CNN to estimate the orientation of the vehicle. With the vehicle orientation and camera calibration the 3D box is estimated based on work of (Dubská et al., 2014). In our case, we adopt the 3D box generation from Dubská et al., but instead of using the single road direction (per scene) for each vehicle, we calculate the orientation for each vehicle independently. This enables to extend the single-road case to scenes with multiple vehicle orientations, such as road crossings, roundabouts and curved roads.
B. Direct Estimation by Object Detection. Early work uses a generic 3D vehicle model that is projected to 2D given the vehicle orientation and camera calibration and then matched to the image. The authors of (Sullivan et al., 1997) use an 3D wire-frame model (mesh) and match it on detected edges in the image and Nilsson and Arnö (Nilsson and Ardö, 2014) use foreground/background segmentation. Vehicle matching is sensitive to the estimated vehicle position, the scale/size of the model, and the vehicle type (stationwagon vs. hatchback). Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005) generalizes the viewpoint specific wire-frame model to a single detection model. Wijnhoven et al. (Wijnhoven and de With, 2011) divide this single detection model into separate viewpoint-dependent models.

State-of-the-art CNN detectors generalize into a multi-layer detection system. Here, the vehicle and its 3D pose are estimated directly from the 2D image. Most methods use two separate CNNs or additional layers. Depth information is used as an additional input to segment the vehicles in the 2D image in (Brazil and Liu, 2019; Ding et al., 2020; Chen et al., 2016; Cai et al., 2020). Although the depth is estimated from the same 2D images, it requires an additional depth-generation algorithm. Mousavian et al. (Mousavian et al., 2017) use an existing 2D CNN detector and add a second CNN to estimate the object orientation and dimensions. The 3D box is then estimated as the best fit in the 2D box, given the orientation and dimensions. RTM3D (Li et al., 2020) and KM3D (Li, 2020) estimate 3D boxes from the 2D image directly into a single CNN. They utilize CenterNet with a stacked hourglass architecture to find 8 key points and the object center, to define the 3D box. Whereas RTM3D utilizes the 2D/3D geometric relationship to recover the dimension, location, and orientation in 3D space, KM3D estimates these values directly, which is faster and can be jointly optimized. We adopt the KM3D model as the 3D box detector because it is fast and can be trained end-to-end on 2D images only.

3 SEMI-AUTOMATIC 3D DATASET GENERATION

This section presents several techniques to semi-automatically estimate 3D boxes from scene knowledge and existing 2D annotated datasets for traffic surveillance. Estimating a 3D box in an image is carried out in the following four steps (see Figure 2).

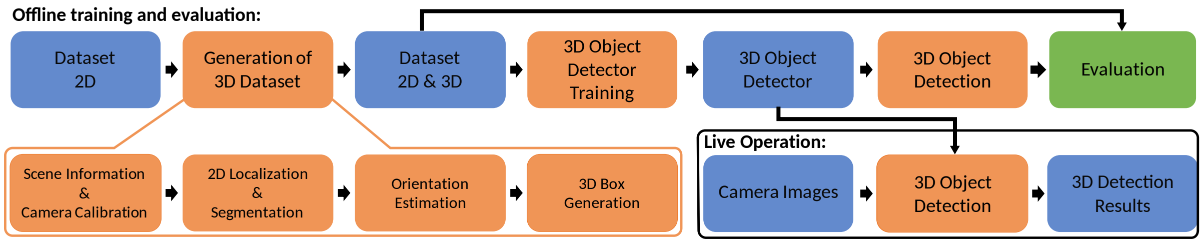


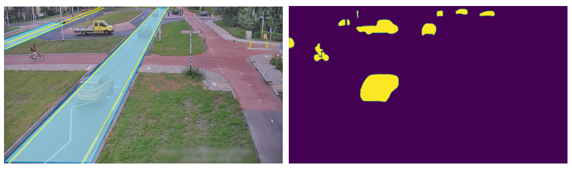
Figure 2: System overview of automatic 3D annotation of 2D images and training the 3D object detector, using inference with live camera images.

First, a camera calibration and scene information is created for each camera scene. Second, the 2D location and segmentation for each vehicle is determined. Third, the orientation of each vehicle is estimated, and finally the 3D box is constructed. Each step is now explained individually.

A. Prior Knowledge. The first step is to gather scene information from an image set and create a corresponding camera calibration. Since datasets often contain many images from the same scene, this prior knowledge has to be generated often only once for each unique scene, resulting in limited manual effort.

The camera calibration involves estimation of all internal and external camera parameters to form a camera projection matrix (Brouwers et al., 2016). This matrix describes the mapping of a pinhole camera model from 3D points in the world to 2D image coordinates and vice versa.

The scene information consists of annotated vanishing points for each straight road section. The yellow lines in the example of Figure 3a show the lines from each straight road section (blue area), which will converge to the corresponding vanishing point (outside the image). The vanishing points are used in our orientation estimation and 3D box-generation steps.



(a) Annotated scene. (b) Segmentation map.

Figure 3: Example scene with configured horizon lines and 2D annotated bounding-boxes (a). Blue areas show the straight road segments, yellow lines are used to find the vanishing points (partially blurred for privacy reasons). The image in (b) shows the corresponding segmentation map of the scene.

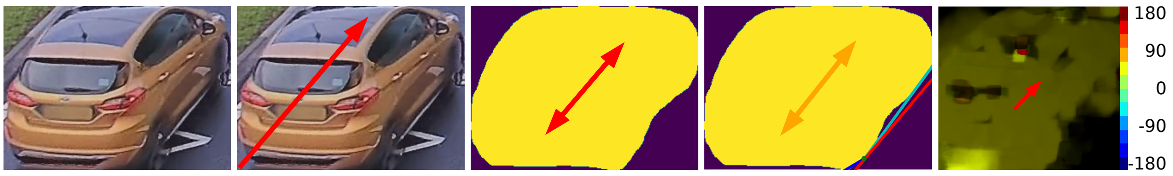
B. Segmentation and 2D Box Locations. Object segmentation for each vehicle is required for some of the techniques for orientation estimation and 3D box-generation steps. Although any algorithm that produces a segmentation mask can be used, we employ the existing Mask-RCNN object detector with instance segmentation (He et al., 2017) for its efficiency. Figure 3b show an example segmentation map. The 2D box detections are matched with the 2D box annotations based on an Jaccard overlap (IoU) threshold. If sufficient overlap is found, the segmentation is matched to our ground-truth annotation. If a ground-truth object is not matched with a segmentation, it is removed from the training set. Detections not matching ground-truth are ignored.

3.1 Orientation Estimation

In this subsection, the goal is to estimate the orientation of the object. The input is the scene information (static vanishing points), camera calibration and the 2D box, while in some techniques also an object segmentation is used. For each object, the output is an orientation (angles) in the real-world domain.

The orientation estimation of all vehicles within a straight road section is based upon the direction of this straight section because we assume all vehicles driving in the correct direction and orientation with respect to these road sections. To this end, an overlap is calculated between each 2D object bounding-box and all the straight road sections (see blue areas in Figure 3a). If the overlap value exceeds a threshold, the orientation of the vehicle is parallel with the direction of the considered road section (see Figure 4b). If an object is not in a straight road section, we apply one of the techniques explained in the following paragraphs. The techniques discussed are based on Principal Component Analysis (PCA), PCA with ray casting technique, and an Optical flow-based method.

A. Principal Component Analysis (PCA). PCA is applied to the collection of foreground points in the segmentation mask. The first principal component corresponds to our orientation estimation which corresponds to the inertia axis of the segmentation mask. Figure 4c illustrates an estimated orientation. If multiple sides of the vehicle are observed (e.g. front, top and side of the vehicle in Figure 4c), the inertia axis of the segmentation will be based on several sides of the vehicle. This introduces an offset in the orientation



(a) Image crop. (b) Prior knowledge. (c) PCA. (d) Ray-Casting. (e) Optical Flow.

Figure 4: Four methods (in (b)-(e)) of estimating orientation for the annotation shown in (a).

estimation. This offset depends on the orientation of the vehicle, the tilt of the camera, and the location in the image (e.g. far field).

B. PCA with Ray-casting. This method is similar to the previous implementation using PCA analysis, but extends this method by additional post-processing using a ray-casting algorithm. This ray-casting algorithm operates in the following two steps.

First, a line with the same orientation as the orientation from the above PCA is fitted to the bottom contour of the segmentation mask (when tilted right (left), this is at the bottom-right (left) corner). This line is shown by the red line in Figure 4d. The point where this line intersects the segmentation contour is the starting point of the ray-casting algorithm.

Second, a ray-casting algorithm is applied to determine the edges of the contour as observed from the intersection point (see the green dot in Figure 4d). The ray-casting algorithm searches for a line from the intersection point that intersects the segmentation contour at another point by evaluating multiple lines with small orientation changes. These orientation changes are carried out in two directions, clockwise and counter-clockwise. The two lines resulting from ray-casting are depicted in the bottom-right corner with dark blue and cyan lines in Figure 4d. The orientation of the two resulting lines are compared to the original estimated direction of the principal components. If any of these two is close in orientation to the starting orientation, the minimum change in orientation is taken, otherwise the ray-casting-based orientation is neglected and the starting orientation is inevitably used (ray-casting did not work).

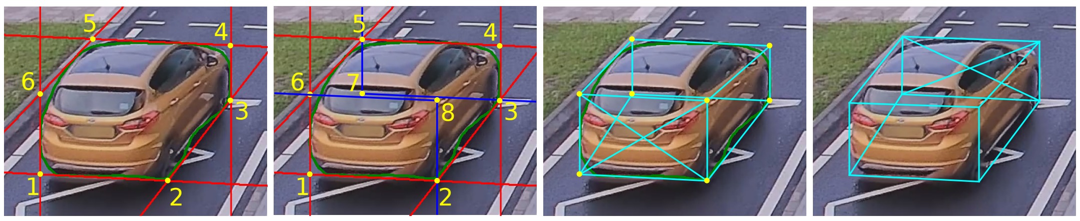
C. Optical Flow. The last orientation estimation method is based upon Optical flow. The Optical flow algorithm measures the apparent motion of objects in images caused by displacement of the object or movement of the camera between two frames. The result is an 2D vector field, where each vector shows the displacement of movement of points between the first and the second frame. The optical flow algorithm is based on Farnebäck's algorithm (Farnebäck, 2003).

To compute the orientation for all vehicles, we first apply a threshold on the magnitude of the optical flow to only select pixels with sufficient movement. Next, the 2D box of each object is used to select the flow vectors that correspond to each object. The orientation is then estimated as the median of the orientations of the Optical flow vectors. An example of Optical flow-based orientation estimation is visualized in Figure 4e, where the background colors indicate the motion directions of points.

3.2 3D Box Generation

The outputs of the prior knowledge, segmentation and orientation estimation steps are the input for the construction of the actual 3D boxes. Note that the methods for creating the 3D boxes in this section are only used to create annotations to train the actual detection model. The final detection for live operation directly estimates the 3D box using the KM3D detector (see Figure 2). This section describes two methods for generating 3D box annotations. The first method is dependent on a correct segmentation map of each object in the scene and is called 'Segmentation fitting', while the second method 'Simple Box' relies on prior information only and does not require segmentation of vehicles. Note that both these methods rely on an estimated orientation, provided by one of the methods described in Section 3.1. On straight road sections this estimation is based on prior knowledge. For curved roads an estimation based on PCA, PCA with ray-casting, or Optical flow is created. Hence, although the estimated orientation differs when a vehicle is not on a straight road section, the 3D box generation methods are similar.

A. Segmentation Fitting. The generation of 3D boxes based on a segmentation uses the box-fitting algorithm of (Dubská et al., 2014), where a 3D box is constructed using three vanishing points. The first vanishing point is located at the crossing of the horizon line and the line in the direction of the estimated vehicle orientation. The second point is the crossing of the horizon line with the line orthogonal to the vehicle orientation. This orientation is estimated in the real-world domain using the camera-projection matrix. The third point is the vertical vanishing point of the scene, which results from camera calibration.



(a) Fitted vanishing lines. (b) Remaining points. (c) Resulting 3D box. (d) Average height 3D box.

Figure 5: Segmentation fitting as the work of Dubská et al.where the vanishing lines (red) are fit to the contour (green). The intersections provide corner-points (yellow) for the 3D box. The blue lines are the remaining vanishing lines used to find the last corner-points.

The algorithm is now as follows (based on (Dubská et al., 2014)). First, two lines are drawn originating at each of the three vanishing points along the sides of the contour, shown as the six red lines in Figure 5a. At the intersections of the lines, six corner-points of the 3D box are found, denoted by the numbered yellow dots (1-6). The last two corner points (7,8) are found as the intersection of lines from corner points 2, 3, 5 and 6 towards the vanishing points (denoted by the blue lines in Figure 5b). The height of the different points of the top surface can vary between corner points because they are estimated individually. However, we assume the roof of a vehicle to be parallel to the ground surface and therefore use the average height of these roof points in the final 3D box, as shown in Figure 5d.

B. Simple Box. Instead of using a segmentation mask for each vehicle, we propose a novel method to estimate a 3D box based on statistical vehicle dimensions. The method requires a 2D bounding box, camera calibration and the vehicle orientation, to convert this information to a 3D box. We assume a fixed vehicle width and determine length and height dynamically from the 2D box fit. Furthermore, the three vanishing points are defined in an identical way as in the segmentation fitting method. The 3D box generation is carried out by the following steps.

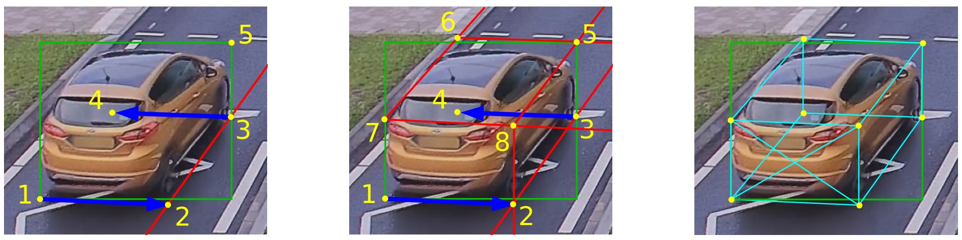
First, we define one of the two bottom corners of the annotated 2D box as the starting point, depending on the estimated orientation (left- or right-facing car). The second point is located at the end of a line segment from the first point towards the second vanishing point (orthogonal to the vehicle orientation) with the statistical vehicle width as its length. This is depicted by the blue line and the points 1 and 2 in Figure 6a. The back-corner point (3) is found by defining the orientation line along the length of the car at the ground plane that crosses the 2D box, resulting in point (3). The last ground-plane point (4) is computed from the third point, by drawing a line segment with fixed length (statistical vehicle width) towards the second vanishing point. The corner-point of the 2D box (point 5) is used as the top of the 3D box. The vehicle height is defined as the distance between points 5 and 3 (Figure 6a). The remaining points of the top of the 3D box can be constructed from point 5 (using the vehicle width/length), or from the ground-plane points (using the height) (see Figure 6b). The final obtained 3D box is depicted in Figure 6c.

We have found that the vehicle height is often estimated too low, caused by small variations in the 2D box locations and the orientation estimations. The estimated height is pragmatically adjusted if it is below a threshold T_{height}, by slightly modifying the orientation. The orientation estimation is updated with one degree in a direction such that the estimated height becomes closer to the average vehicle height. The adapted direction depends on the orientation of the vehicle. The threshold T_{height} is defined as the statistical average vehicle height H_{avg} minus the $H_{\text{variation}}$ value. This process of updating the orientation is repeated until the estimated vehicle height exceeds the threshold. The selected threshold values are discussed in Section 4(A).

4 EXPERIMENTS

The 3D box generation processing is evaluated with two different approaches. First, we evaluate a direct approach where the output of the annotation process (without the 3D detector) is considered. Second, we train and evaluate the performance of the 3D object detector (KM3D (Li, 2020)) based on the output of the box generation. In two additional experiments with the KM3D detector, a cross-validation between our dataset and the KITTI dataset is presented and an unsupervised approach of the generation pipeline is evaluated.

A. Configuration. All experiments are carried out on 4 different processing pipeline configurations. The first three configurations use a segmentation mask as input for the 'segmentation fitting' technique to estimate the 3D box. The orientation estimation in each



(a) Estimated ground-surface points based on vehicle width.

(b) The remaining roof points estimated based on vanishing lines.

(c) Resulting 3D box.

.

Figure 6: Simple Box construction method for corner-points (yellow) of the 3D box with the assumption that a car has an width of 1.75m (blue arrows). The other 3D corner-points are calculated through the estimated vanishing lines (red and cyan) and their intersections.

of these three pipelines is carried out with PCA, PCA + ray-casting or optical flow. The fourth pipeline is based on the Simple box 3D box estimation technique, and uses Optical flow-based orientation estimation, such that it does not depend on an object segmentation map. The Farnebäck (Farnebäck, 2003) Optical-flow algorithm is computed on a pyramid of 3 layers, where each layer is a factor of two smaller than the previous one. The averaging window size is 15 pixels. In all experiments using a segmentation mask, Mask-RCNN (pre-trained on the COCO dataset (Lin et al., 2015)) is used to predict 2D boxes and the corresponding object segmentation. The Mask-RCNN segmentation masks have proven to be accurate for vehicles in surveillance scenarios. The detection threshold is set to 0.75 and a minimum IoU threshold of 0.5 is applied to match between detections and ground-truth boxes. For the Simple box method, the vehicle width is set to 1.75 meters and the average height $H_{\text{avg}} = 1.5$ meters. We have empirically determined a maximum offset of $H_{\text{variation}} = 0.25$ meters with respect to the statistical average height H_{avg}, resulting in $T_{\text{height}} = 1.3$ m. For training the KM3D model, we have used the parameters, data preparation and augmentation process from the original implementation (Li, 2020). We train for 50 epochs using learning rate 5×10^{-5} and a batch size of 16.

B. Traffic Surveillance Dataset. Our experiments are carried out on a proprietary traffic surveillance dataset annotated with 2D boxes. This dataset contains 25 different surveillance scenes of different traffic situations like roundabouts, straight road sections and various crossings. The dataset is split in 20k training images and 5k validation images, with 60k and 15k annotations, respectively. The ground-truth test set used for evaluation consists of 102 images with 444 3D boxes which are manually annotated. This set contains 8 different surveillance scenes, which are outside the training and validation sets. In all experiments except the cross-validation experiment, we combine the train dataset from KITTI (Geiger et al., 2012) with the train part of our dataset to increase the amount of data.

C. Evaluation Metrics. In our experiments, we measure the average precision 3D IoU (AP3D) and average orientation similarity (AOS), as defined by the KITTI dataset (Geiger et al., 2012), with two different IoU thresholds of 50% and 70%. Next to these existing metrics, we introduce the Middle point Mean Error (MME) of the ground surface. This MME is defined as:

$$MME = \frac{1}{N} \sum_{i=0}^{N-1} \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2}, \quad (1)$$

where N is the total number of true positive detections, $\hat{x}$ and $\hat{y}$ are the ground-truth x- and y-locations of the middle-point on the ground surface in real-world coordinates, where x and y are the estimated x- and y-locations. This metric depicts the average location error of the center point on the ground plane in 3D world-coordinates (ignoring the vehicle width/height/length dimension).

4.1 Optimizing 3D Annotation Pipeline

This experiment evaluates the four different annotation pipeline configurations, which create 3D annotations from the existing 2D annotated datasets. To this end, all pipelines are executed on the testing dataset, where the 2D annotated boxes of the testing dataset are used as input. The resulting 3D boxes generated by the different pipelines are compared with the manually annotated 3D boxes in the testing dataset.

The results of the proposed pipelines are shown in the left part of Table 1. The Simple box pipeline clearly results in the best detection score (IoU=0.5) with 36.4% mAP for Hard objects, most accurate center-point estimation (MME=0.66) and best orientation estimation (AOS=93.8). Note that detection performance for the strict IoU=0.7 evaluation is very

Table 1: Comparison of 3D annotation processing configurations without 3D object detector (left) and the same comparison for the 3D annotation configurations with trained 3D detector on the data generated by the configurations (right). Detection performance increases with the addition of the 3D detector (IoU=0.5). The Simple box-based pipeline is the best.

	3D Annotation pipeline					**3D Annotation pipeline + 3D detector**				
	AP3D IoU=0.7 / IoU=0.5			**MME**	**AOS**	**AP3D IoU=0.7 / IoU=0.5**			**MME**	**AOS**
Pipeline	Easy	Moderate	Hard			Easy	Moderate	Hard		
PCA	5.9/42.1	5.7/37.6	3.3/23.3	0.80	84.7	2.8/53.5	2.9/49.8	2.5/35.9	0.66	89.2
RayCast.	6.0/41.9	5.8/31.7	3.3/22.9	0.80	84.9	3.5/53.3	3.6/48.0	2.9/34.0	0.85	89.4
Opt.Flow	7.1/52.0	6.5/39.5	3.8/24.8	0.71	90.0	4.2/62.8	5.6/51.0	3.8/42.9	0.66	89.5
Simple b.	18.9/59.7	13.1/46.7	9.9/36.4	0.66	93.8	26.2/73.4	19.9/62.9	15.6/51.9	0.70	89.6

limited, probably because of the inaccurate 3D boxes generated by the pipeline. The practical performance for IoU=0.5 is representative for the application of video surveillance.

A. Orientation Estimation. PCA and ray-casting both depend on the segmentation map. Whereas PCA is dependent on the mass of the segmentation, ray-casting fits to the contours of the segmentation. The ray-casting technique (84.9%) results in a minor improvement in AOS score with respect to PCA (84.7%). Since the AOS scores of Optical flow and Simple box are higher, which both use Optical flow as methods for orientation estimation, it can be concluded that Optical flow outperforms the segmentation map-based methods. However, Optical flow cannot be computed when previous/next frames are not available, then the ray-casting method can be used.

B. Segmentation Fitting. Visual inspection of the 3D generated box results of the segmentation fitting-based methods show that the main cause of errors is the width and length estimation, as shown in Figure 7a. In this example, the length is estimated 1 meter too short. These errors are caused by segmentation maps that do not perfectly align with the actual object contours. The segmentation maps do not consistently include the headlights, wheels and other parts of the vehicles and sometimes include vehicle shadows. Although we always map the Mask-RCNN detection and segmentation output to our ground-truth

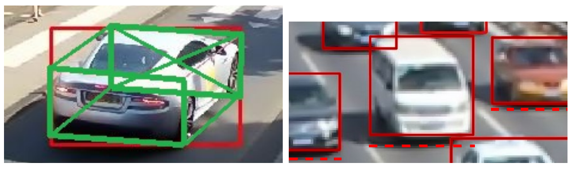

(a) Segmentation fitting. (b) Simple box.

Figure 7: Two examples of cases which result in wrongly estimated 3D boxes. At the left-hand side, a segmentation fitting problem occurs because the front-right wheel point is estimated too far to the right and does not extend beyond the wheel in length. At the right-hand side, the Simple box method will estimate the box too high because the 2D-box bottom line does not touch the ground surface but aligns with the vehicle bumper. The dashed red line depicts the desired bottom line for the 2D-box.

2D bounding-boxes to ensure that no false detections are used as 3D ground truth, we rely on the quality of the generated segmentation maps from Mask-RCNN.

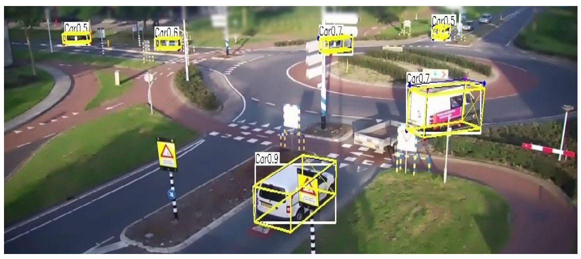
C. Simple Box. The Simple box pipeline results in higher performance than the segmentation fitting-based methods (36.4% vs. 22-24% AP3D). From these results we conclude that a 2D box annotation and orientation information is sufficient to estimate a 3D box. Visual inspection of the results show that errors commonly originate from 2D bounding boxes of which the bottom line is not touching the ground plane. In such a case, the starting point for the Simple box method is too high, so that the constructed 3D box floats in the air (see the examples in Figure 7b).

4.2 3D Annotation Pipeline + Detector

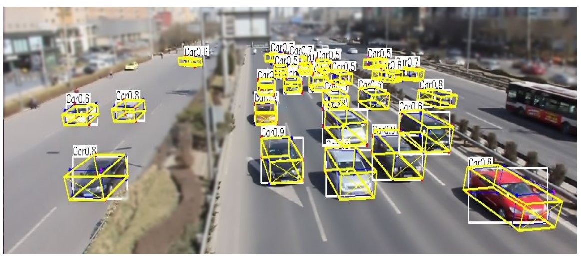
In this experiment, the KM3D detector is trained on the 3D annotations generated by the different annotation configurations. The performance of the KM3D detector is measured on the test dataset.

The scores from the detection results are shown at the right-hand side of Table 1. The addition of the 3D detector results in a significantly increased detection performance (increase of 12-18% AP3D) for all configurations with respect to the direct 3D annotation pipeline results (left-hand side in Table 1). Visual inspection of the detection results for PCA, ray-casting and Optical flow shows that the estimated ground surface is estimated better by the 3D detector than the direct output of the annotation pipeline. This means that the annotation quality of the generated 3D box dataset is sufficient to train a model that generalizes over all the noisy input data.

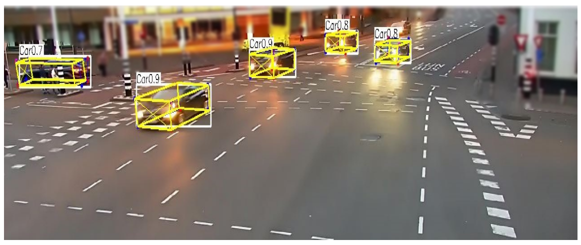
Some detection results of the 3D detector trained with annotations from Simple box are shown in Figure 8. These images visualize three limitations of the method. First, not every vehicle is estimated on the ground surface. Second, the orientation estimation of vehicles that are driving in a horizontal direction are prone to errors. Third, the height of the vehicles often seems to be estimated too low. However, more detailed measurements for the estimated vehicle dimensions show that the height, length and width have



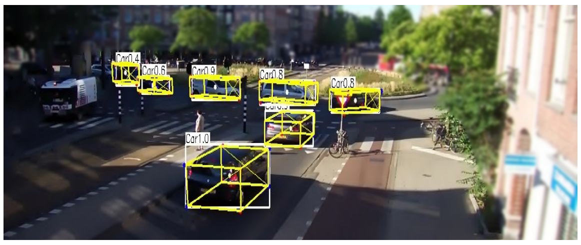
(a) Correct results in a complex scene.



(b) Some faulty orientation estimations in the back.



(c) Horizontally oriented vehicle is estimated incorrectly.



(d) Incorrect height estimation.

Figure 8: Detection results of KM3D trained with the annotations generated by the SimpleBox pipeline. Partially blurred for privacy reasons. Majority of the 3D bounding-boxes are accurately estimated.

a mean offset with respect to the ground-truth annotations of 0.05, 0.02 and 0.01 meters, respectively.

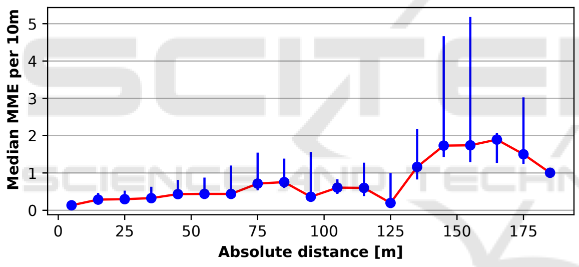


Figure 9: Box plot showing the median (blue dots) of the error for objects at each 10-m distance from the camera. Lines above and under the dots represent the standard deviations for errors estimated further away and nearby, respectively.

In a last experiment with the KM3D detector we investigate the effect of the distance of each vehicle to the camera on the accuracy of the 3D box estimation, trained by using annotations from the Simple box pipeline. We calculate the error in steps of 10-m distance from the camera. The results are shown in Figure 9. The blue dots show the median values for each 10-m interval, the blue lines above/below represent the standard deviations of the errors of detections at larger/smaller distance. From the results, it can be observed that accurate position estimation is possible until about 130-m distance. The MME metric becomes inaccurate from about 140-m distance, but our test set lacks the images to validate this in more detail. Note that the manual generation of 3D ground-truth boxes also becomes more inaccurate at larger distance to the camera (and thus, lower resolution). Moreover, a small offset in image coordinates far away from the camera can lead to large errors in the real-world locations.

Table 2: Cross-validation on the KITTI dataset with the KM3D detector, based on the AP3D metric (IoU=0.5). The 3D annotations for training are generated with the Simple box pipeline.

Train set	Test set	Easy	Mod.	Hard
KITTI	KITTI	51.4	41.0	33.5
KITTI & Ours	KITTI	55.3	44.7	35.2
KITTI	Ours	5.2	3.87	2.3
Ours	Ours	66.0	60.1	43.4
KITTI & Ours	Ours	73.4	62.7	51.9

4.3 Cross-validation on KITTI Dataset

This experiment evaluates the baseline KM3D detector on our test set and performs a cross-validation on the well-known KITTI dataset (Geiger et al., 2012). Although KITTI data focuses on autonomous driving, this will provide insight in the differences with our application. The KM3D detector is trained independently on KITTI, our dataset, and the combination. Our training set annotations are generated using the Simple box pipeline. Evaluation is carried out by inference on the KITTI test set and our test set. The results are measured by the AP3D metric.

Table 2 depicts the obtained results. The original results on the KITTI test set are similar to the original work (Li, 2020). Testing the KITTI-trained model on our dataset results in poor performance because of the

Table 3: Detection performance when using unsupervised Simple box pipeline (replacing 2D annotations with Mask-RCNN output). Unsupervised performance (US) is lower than when using the original 2D annotations.

	AP3D IoU=0.5			**MME**
Method	**Easy**	**Mod.**	**Hard**	
3D annotations	59.7	46.7	36.4	0.66
3D annotations US	37.4	31.5	15.6	0.76
3D ann. + KM3D	73.4	62.9	51.9	0.70
3D ann. + KM3D US	67.3	55.8	40.7	0.65

changed viewpoints. Training on the combination of both datasets increases the AP3D with about 3% on the easy cases when evaluating on the KITTI dataset. The AP3D performance on our dataset increases from 55.6% to 62.5%. Although the two datasets contain different camera viewpoints, the detection model effectively uses this information to generalize better.

4.4 Unsupervised Approach

This experiment investigates the effect of replacing the 2D box annotations with automatic detections from Mask-RCNN, thereby limiting the annotation effort to scene and camera-calibration parameters only. The 2D annotations which are used in the Simple box pipeline are created using the pre-trained Mask-RCNN detection network, instead of using the annotated 2D boxes. We apply this modified Simple box pipeline directly to our train and test set images and train the KM3D detector on the output.

The results are shown in Table 3. The pipeline with the 3D object detector has superior results over the unsupervised configuration without 3D object detector. The AP3D performance gap is $20-25\%$. The AP3D difference between the unsupervised configuration and the supervised configuration is only $6-11\%$. Visual inspection shows that the 2D detection set generated by Mask-RCNN contains many false detections. This results in falsely annotated 3D boxes in our Simple box pipeline and causes the KM3D detector to learn false objects. Nevertheless, since the unsupervised pipeline is fully automatic and requires no prior annotations, it shows opportunities for future work.

5 CONCLUSION

We have investigated the detection of 3D bounding boxes for objects using calibrated static monocular cameras in traffic surveillance scenes. It has been found that it is possible to directly estimate 3D boxes of vehicles in the 2D camera image using a 3D object detector. We have selected the KM3D CNN-based detection model, which has only been applied earlier in autonomous driving with different viewpoints. Although the 3D object detector can be directly applied to traffic surveillance, 3D public datasets are not available to train the 3D detection model.

Therefore, we propose a system to semi-automatically construct 3D box annotations, by exploiting camera-calibration information and simple scene annotations (static vanishing points). To this end, we use existing traffic surveillance datasets with 2D box annotations and propose a processing pipeline to convert the 2D annotations to 3D boxes. The resulting 3D annotated dataset is used to train the KM3D object detector. The trained 3D detector can be directly integrated in surveillance systems, as inference only requires 2D images and camera calibration.

For optimization, we have validated four different annotation processing configurations, each containing orientation estimation, segmentation and 3D construction components. In addition to combinations of existing components, we propose the novel Simple box method. This method does not require segmentation of vehicles and provides a more simple 3D box construction, assuming a fixed predefined vehicle width. When comparing our different 3D annotation configurations, we have found that the Simple box pipeline provides the best object annotation results, albeit all four configurations result in accurate orientation estimation and localization. Using the 3D box annotations from Simple box directly (without the 3D object detector), the AP3D score is 36.4% which improves to 51.9% AP3D when using KM3D trained on this data. Similarly, we have found that the different viewpoints from the KITTI autonomous driving dataset actually increase performance when adding it to our traffic set.

The experimental results show that it is possible to use a 2D dataset to generate a 3D dataset. It has been shown that the existing KM3D object detector trained on the generated dataset generates more accurate 3D vehicle boxes than the vehicle annotations from the proposed automatic 3D annotation pipeline, due to its capacity to generalize. The resulting 3D box detections are accurate (51.9% AP3D), both in locations and size, up to about 125 meters from the camera. The use of unsupervised annotation using existing 2D detectors can potentially increase the 3D detection performance even further.

REFERENCES

Brazil, G. and Liu, X. (2019). M3d-rpn: Monocular 3d region proposal network for object detection.

Brouwers, G., Zwemer, M., Wijnhoven, R., and With, P. (2016). Automatic calibration of stationary surveillance cameras in the wild. In *Proc. of the IEEE CVPR*.

Cai, Y., Li, B., Jiao, Z., Li, H., Zeng, X., and Wang, X. (2020). Monocular 3d object detection with decoupled structured polygon estimation and height-guided depth estimation. In *AAAI*.

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., and Urtasun, R. (2016). Monocular 3d object detection for autonomous driving. In *Proc. of the IEEE CVPR*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

Ding, M., Huo, Y., Yi, H., Wang, Z., Shi, J., Lu, Z., and Luo, P. (2020). Learning depth-guided convolutions for monocular 3d object detection. In *Proc. of the IEEE/CVF Conference on CVPR*.

Dubská, M., Herout, A., and Sochor, J. (2014). Automatic camera calibration for traffic understanding. In *BMVC*, volume 4,6, page 8.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conf. on Image Analysis*. Springer.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conf. on CVPR*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proc. of the IEEE ICCV*.

Li, P. (2020). Monocular 3d detection with geometric constraints embedding and semi-supervised training.

Li, P., Zhao, H., Liu, P., and Cao, F. (2020). Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. *arXiv preprint arXiv:2001.03343*.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft coco: Common objects in context.

Mousavian, A., Anguelov, D., Flynn, J., and Kosecka, J. (2017). 3d bounding box estimation using deep learning and geometry.

Nilsson, M. and Ardö, H. (2014). In search of a car - utilizing a 3d model with context for object detection. In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications - Volume 2: VISAPP, (VISIGRAPP 2014)*, pages 419–424. INSTICC, SciTePress.

Sochor, J., Špaňhel, J., and Herout, A. (2018). Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance. *IEEE transactions on intelligent transportation systems*.

Sullivan, G. D., Baker, K. D., Worrall, A. D., Attwood, C., and Remagnino, P. (1997). Model-based vehicle detection and classification using orthographic approximations. *Image and vision computing*, 15(8):649–654.

Wijnhoven, R. G. and de With, P. H. (2011). Unsupervised sub-categorization for object detection: Finding cars from a driving vehicle. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2077–2083. IEEE.