Lightweight convolutional neural network for real-time 3D object detection in road and railway environments

A. Mauri, Redouane Khemmar, B. Decoux, M. Haddad, Rémi Boutteau

Journal of Real-Time Image Processing (2022) 19:499–516
https://doi.org/10.1007/s11554-022-01202-6

ORIGINAL RESEARCH PAPER

Lightweight convolutional neural network for real-time 3D object detection in road and railway environments

A. Mauri[1] · R. Khemmar[1] · B. Decoux[1] · M. Haddad[2] · R. Boutteau[3]

Received: 29 July 2021 / Accepted: 10 January 2022 / Published online: 11 February 2022

Abstract

For smart mobility, and autonomous vehicles (AV), it is necessary to have a very precise perception of the environment to guarantee reliable decision-making, and to be able to extend the results obtained for the road sector to other areas such as rail. To this end, we introduce a new single-stage monocular real-time 3D object detection convolutional neural network (CNN) based on YOLOv5, dedicated to smart mobility applications for both road and rail environments. To perform the 3D parameter regression, we replace YOLOv5's anchor boxes with our hybrid anchor boxes. Our method is available in different model sizes such as YOLOv5: small, medium, and large. The new model that we propose is optimized for real-time embedded constraints (lightweight, speed, and accuracy) that takes advantage of the improvement brought by split attention (SA) convolutions called small split attention model (Small-SA). To validate our CNN model, we also introduce a new virtual dataset for both road and rail environments by leveraging the video game Grand Theft Auto V (GTAV). We provide extensive results of our different models on both KITTI and our own GTAV datasets. Through our results, we show that our method is the fastest available 3D object detection with accuracy results close to state-of-the-art methods on the KITTI road dataset. We further demonstrate that the pre-training process on our GTAV virtual dataset improves the accuracy on real datasets such as KITTI, thus allowing our method to obtain an even greater accuracy than state-of-the-art approaches with 16.16% 3D average precision on hard car detection with inference time of 11.1 ms/image on an RTX 3080 GPU.

Keywords 3D bounding box estimation · 3D multi-object detection · Multi-modal dataset · Deep learning · Smart mobility

Abbreviations

ADAS	Advanced driver assistance system
GTAV	Grand theft auto V
CNN	Convolutional neural network
YOLOv5	You only look once
KITTI	Karlsruhe Institute of Technology and Toyota technological institute
LiDAR	Light detection and ranging
L-CNN	Lightweight-CNN
CARLA	CAR learning to act
ROAD	ROad event awareness dataset
SYNTHIA	SYNTHetic collection of Imagery and Annotations
R-CNN	Region-based CNN
LSTM	Long short term memory
GAM3D	Ground-aware monocular 3D object
KFPN	Keypoint feature pyramid network
M3D-RPN	Monocular 3D region proposal network
SA	Split attention
API	Application programming interface
IOU	Intersection over union
AP	Average precision
ToF	Time-of-flight
DS	Dimension score
CS	Center score

✉ R. Khemmar
redouane.khemmar@esigelec.fr

A. Mauri
antoine.mauri@esigelec.fr

B. Decoux
benoit.decoux@esigelec.fr

M. Haddad
madjid.haddad@segula.fr

R. Boutteau
remi.boutteau@univ-rouen.fr

[1] Normandie Univ., UNIROUEN, ESIGELEC, IRSEEM, 76000 Rouen, France

[2] SEGULA Technologies, 19 Rue d'Arras, 92000 Nanterre, France

[3] Normandie Univ., UNIROUEN, UNILEHAVRE, INSA Rouen, LITIS, 76000 Rouen, France

OS	Orientation score
GT	Ground truth
RoIs	Region of interests
SRE	Squared relative error
RMSE	Root mean square error
MTL	Multi task learning

1 Introduction

Multiple object detection consists of detecting and identifying objects in scenes. In smart mobility applications, vehicles must have a better perception of their environment. This perception must guarantee not only a good detection of the objects, but also a good interpretation of the scenes. In multimodal road and railway smart mobility, the objects of interest to be detected are vehicles, pedestrians, bus, cyclists, trees, etc. If the road sector is widely covered in the scientific literature, this is less the case for the railway smart mobility. This is in part due to the lack of specific railway datasets with ground truth data dedicated to 3D object detection. This poses a real challenge in that it requires the 3D object detection algorithm to train under dataset with ground truth data.

In the past few decades, specifically since 2012, Deep Learning has become a very powerful tool because of its ability to handle large amounts of data. The interest using more and more hidden and intermediate fully connected layers has surpassed traditional techniques in image processing and computer vision, especially in pattern recognition, object detection, and classification [1, 2]. One of the most popular deep neural networks is CNN. With the rise of Deep Learning approaches for computer vision tasks and the emergence of CNNs, it has become possible for cameras to be used for object detection, depth estimation, tracking, instance, and semantic segmentation, etc. These kinds of methods have many applications, especially for smart mobility to enhance safety and increase vehicle autonomy. This need for environmental perception has led to the development of 3D detection methods, using either a LiDAR-type depth sensor or images from cameras, to improve safety. Although 3D object detection is a much sought-after field, few methods focus on the real-time aspect that is essential for a real-world application. Of course, in road and railway multimodal smart mobility, it is desired that 3D multi-object detection is performed in real time. This represents another challenge. In this paper, we propose to fill these gaps, lack of railway dataset with ground truth data and real-time 3D object detection, by proposing a new virtual dataset (GTAV) dedicated to both road and railway environments for 3D object detection, which includes images taken from the point of view of both cars and trains. With this dataset, we propose a new method dedicated to real-time 3D object detection based on the well-known 2D object detector YOLOv5 [3]. With this new method, we introduce 3D anchor boxes which makes our method the fastest method available for 3D object detection with accuracy comparable to state-of-the-art methods. Using a pre-trained model on our GTAV dataset, our approach outperforms state-of-the-art methods. Through our experiments, we also demonstrated the validity of our method for real-time embedded applications for autonomous vehicles (cars and trains). The main contributions put forward in this paper are the following:

1. A new Lightweight CNN (L-CNN) method for real-time 3D object detection based on the YOLOv5 2D object detector. Our L-CNN allows the following object predictions:
   - 2D detection
   - Distance from the camera
   - 3D centers projected on the image plane
   - 3D dimensions and orientation
2. A new virtual dataset (GTAV) dedicated to both road and railway environments for 3D object detection. The GTAV dataset includes images taken from the point of view of both cars and trains.

The remainder of this paper is organized as follows. Section 1 introduces the paper. In Sect. 2, we review the related work for autonomous navigation datasets, 3D object detection approaches in which we will present methods based on depth sensors as well as methods based on monocular images. In Sect. 3, we describe in more details our multimodal road/railway GTAV dataset. Our new L-CNN model about the 3D multi-class object detection is presented in Sect. 4. Analysis and experimental results are described in Sect. 5. Finally, the conclusion and future directions are outlined in Sect. 6.

2 Related work

2.1 Autonomous driving datasets

Any approach based on Deep Learning requires annotations (also called "ground-truth") to perform a training. One of the main factors in the performance of any of these approaches in terms of precision is the quality and quantity of the data in the dataset. Among the datasets dedicated to autonomous vehicles, the vast majority are intended to autonomous driving in road environments and focus on object detection and scene segmentation. Some of them are mono-modal (in general, with images as a single modality), and others are multi-modal.

Even though the building of mono-modal datasets is less time-consuming and expensive than multimodal datasets. The latter represent a crucial advantage in the use of supervised artificial intelligence approaches because they offer a wider variety of ground truths from different sensors. Although these data are expensive because they require the use of expensive sensors such as LiDAR or radar, they are indispensable for environmental perception applications such as object detection/localization, distance estimation, tracking. We can also find many other road datasets for similar tasks such as CityScape [4], PascalVOC [5], etc. The best-known road dataset is KITTI [6] (Karlsruhe Institute of Technology and Toyota Technological Institute), which includes a rich ground truth from both distance sensors (LiDAR), a GPS/IMU unit, and stereoscopic cameras (color or grayscale). It comprises different information captured and synchronized at $10Hz$. The vehicle is equipped with different sensors: cameras, 3D Velodyne LiDAR (100k points per frame), 3D GPS/IMU data (location, speed, acceleration). The camera is calibrated and synchronized with Velodyne LiDAR and with a GPS/IMU sensor [6]. KITTI includes 3D object track labels for different object classes (cars, trucks, trams, pedestrians, cyclists). It is a reference base for both training and evaluation of methods for stereo, optical flow, visual odometry, 2D object detection, 3D object detection, depth estimation, 2D tracking, etc. Among the road datasets, we can also find the NuScenes [7] dataset which is a public large-scale dataset for autonomous driving. NuScenes is more recent than the KITTI dataset which inspired it. It includes 1000 driving scenes (of 20 s in length each) in Boston and Singapore, cities known for their dense traffic. It includes 23 object classes annotated with accurate 3D bounding boxes at 2 Hz which is a good annotation for common autonomous driving tasks such as object detection, distance estimation, and tracking. NuScenes includes approximately 1.4 M camera images, 390 k LIDAR sweeps, 1.4 M RADAR sweeps, and 1.4 M object bounding boxes. As with KITTI, this dataset incorporates one LiDAR sensor, 5 radar sensors, 6 cameras to provide 360° coverage around the vehicle, and one GPS/IMU unit [7]. This dataset contains $100x$ more images than KITTI and allows training methods for 2D and 3D object tracking, 2D and 3D object detection, depth estimation.

ROad [8] event Awareness Dataset for Autonomous Driving (ROAD), includes 22 videos with 122 K annotated video frames, and 560 K detection bounding boxes with 1.7 M individual labels. ROAD is a dataset that was designed to test a robot-car's situation awareness capabilities. The annotation process follows a multi-label approach in which road agents (vehicles, pedestrians, etc.), their locations as well as the action they [8].

We can find many other road datasets for similar tasks which are also multimodal large-scale autonomous vehicles datasets. Argoverse [9] dataset images are captured under different weather conditions. It includes annotations for RGB images, LiDAR/radar, and 3D boxes. It includes more sensors than both KITTI [6] and NuScenes [7] datasets. Argoverse includes 3D bounding boxes with tracking information for 15 objects of interest [9]. Another multimodal dataset, Lyft [10] uses cameras and LiDAR to provide 2D annotation and 3D bounding boxes for 4 object classes. All these datasets provide contextual knowledge through human-annotation which is very important for scene understanding applications. Waymo open dataset [11] is a new large-scale, high-quality LiDAR and camera data that includes 1150 scenes (each with the $20s$). Sensors produce annotated data with 2D (camera image) and 3D (LiDAR) bounding boxes, with consistent identifiers across frames.

To our knowledge, there are only a few image datasets dedicated to the railway [12, 13]. However, these datasets are not dedicated to 3D detection and do not have ground truth information on depth estimation. Therefore, these datasets cannot be used for the training or evaluation of 3D object detection approaches.

Finally, there are databases of virtual images created with simulators such as CARLA [14] or SYNTHIA [15]. The advantage of these databases is that they are simpler to acquire, the ground truth data are acquired automatically by the API simulator, which allows the acquisition of mass data. It also allows acquiring a database adapted to specific needs such as the understanding of scenes in the railway environment. However, one of the disadvantages of this kind of dataset is that they have dated graphics, which makes it difficult to use trained dataset in real conditions. To overcome this problem, recent work has been done on the acquisition of a database via a video game [16]. Indeed, video games make graphic quality their priority, which makes using of video games particularly interesting. The video game GTAV is used to acquire road databases similar to KITTI or CityScape. Although there are many virtual road datasets, to our knowledge, there is no similar state-of-the-art work for the railway.

2.2 3D object detection from monocular images

In the very recent years, many methods based on deep learning have been proposed for the detection of 3D objects from monocular images. Most of them use single RGB images, and a few use image sequences. Many of these methods are extension of proven 2D detectors, thus leveraging their very good performances. 3D parameters processing is then added to the models. Some other methods infer 3D parameters directly through end-to-end learning.

In [17], the problem is tackled in two steps. First candidate 3D boxes are generated and scored by exploiting several features, namely class semantic, instance semantic,

contour, object shape, context, and location prior. Then the candidates with the best scores are applied to a Fast R-CNN [18] to predict class labels, bounding-box offsets and object orientation.

In [19], 2D detection is done by the multi-scale region-proposal MS-CNN [20], extended to regress the orientation and the dimensions of the 3D bounding boxes. Estimation of orientation uses a MultiBin principle, in which angle is turned to a hybrid classification and regression task, in which values are separated into bins representing the classes.

DeepMANTA [21] is a coarse-to-fine model for simultaneous vehicle detection, localization, visibility characterization, and 3D dimension estimation. It is based on a two-step process, the first one of which outputs bounding boxes associated with vehicle information, and the second one uses these outputs and a 3D vehicle virtual dataset to recover 3D orientations and locations. In [22], a depth map obtained by means of a fully-convolutional network (FCN) is fused with the input RGB image before feeding a region-proposal network (RPN). It is also combined at different level in the flow stream to finally get estimation of class labels, 2D box position, and 3D box orientation, dimensions and location. For the orientation, the same MultiBin principle as in [20] is used.

In [23], a CNN model for joint 3D object detection and tracking is proposed. It is based on a faster RCNN to predict the 2D bounding boxes, which are then associated to 3D information including position, orientation, dimensions, and projected 3D box centers of each object. The tracking across sequences of frames is obtained by two LSTM (Long Short-Term Memory) layers, allowing a further refinement of the 3D bounding box positions.

In SMOKE [24], a CNN is trained end-to-end in a single stage by means of a unified loss function. The parameters of 3D bounding boxes are regressed directly without using 2D detection, thanks to a network made of two branches: a classification branch where an object is encoded by its 3D center (projected on the image plane), and a 3D parameter regression branch to construct 3D bounding box around each center. In MonoGRNet [25], 3D object detection is decomposed into four sub-tasks: 2D object detection, object center depth estimation, projected 3D center estimation and local corner regression. Those four pieces of information are obtained in parallel through a single-pass process, and then combined.

Anchor-based methods have also been proposed in M3D-RPN [26] and GAM3D [27]. M3D-RPN leverages depth-aware convolution for locating the specific features and improve the 3D scene understanding. The approach consists only of a single-stage using anchor boxes for predicting both 2D and 3D object positions. GAM3D also uses 3D/2D anchor boxes to predict the objects' 3D bounding box and introduces ground-aware convolution module to provide additional 3D cues to the network, thus improving the accuracy of the network.

In [28], 3D detection is reformulated as a keypoint detection problem. A keypoint feature pyramid network (KFPN) is defined, providing the center and the perspective points of the 3D bounding-boxes. The fact that the network backbone is based on a lightweight architecture (such as ResNet-18 [29] or DLA-34 [30]) and the anchor-free nature of the detector, allow real-time processing on 1080Ti GPU.

Most of these models are dedicated to improve performance of 3D object detection in terms of accuracy. A few other focus more on processing time, aiming for real-time on powerful GPUs. In our case, we want to focus on lightweight CNN architectures making it possible to achieve real-time on low-power GPUs like the Jetson-TX2. We believe that it is important to go further on this point to facilitate the large-scale development of deep-learning solutions for the safety of autonomous vehicle. Another important aspect of our needs is to be able to perform well in road environment as well as railway environment. The lack of existing railway dataset has led us to develop our own dataset, made of virtual images from GTA-V game. We will use this custom dataset in complement to KITTI. In addition to our virtual railway dataset, we are developing a real railway dataset with ground truth allowing us to go further in the development of railway e-Advanced Driver Assistance Systems (e-ADAS).

3 Multimodal road/rail dataset

3.1 GTAV virtual multi-modal dataset presentation

To compensate for lack of real dataset in the railway environment, we turn to a virtual dataset. The main advantage of a virtual dataset is the simplification of its acquisition and annotation. However, this is done at the cost of the fidelity to reality. Indeed, most of these datasets (such as CARLA) do not have sufficient graphical fidelity to allow the methods trained on them to offer good performances once applied in real world conditions. To overcome this problem, we turn to video games to acquire a railway database for 3D object detection. Indeed, where simulators lack graphical fidelity, video games try to offer the best graphical fidelity to offer users a feeling of authenticity. This is why several works have already been carried out in this field to extract databases. One of the most interesting games is the video game Grand Theft Auto V, which allows the user to drive road vehicles and presents realistic road scenes with, for example, level crossings, crosswalks, heavy traffic, etc. A lot of work has been done in this direction to build databases for the road environment [16, 23]. Our railway database has been acquired with a data acquisition pipeline based on the work of [31]. To do this we used a modified version of the game

[32] code to allow the user to drive a subway or a train, allowing us to build a hybrid database with both road and rail scenes that can then be used for training and evaluation of our 3D detection approach.

3.2 GTAV virtual multi-modal dataset architecture

The ground truth is generated automatically by the game's API and includes among other things the 3D and 2D bounding boxes, the object class, the depth map and the semantic map. All these annotations are taken from the point of view of a stereoscopic camera with an acquisition frequency of 10 Hz. Some statistics for our dataset can be found in Figure 1.

During the acquisition of the dataset, several classes of objects were annotated. However, to combine our virtual dataset with the 3D detection KITTI dataset, we selected only the classes that are also present on KITTI (**Car, Person, Cyclist**) dataset. Our dataset is composed of 140 K samples from the stereoscopic camera. Each of these samples has a depth map and a ground truth for 3D detection (position in the camera coordinates, dimensions, orientation, etc.). Image samples from our dataset can be found in Fig. 2. Our dataset also offers various situations and environmental conditions resulting in a challenging dataset for computer vision tasks. The dataset includes night, day, rainy and clear weather scenes, offering a wide variety of visibility conditions.

4 3D multi-class object detection

Our method is a single-stage 3D object detector based on the YOLOv5 2D detector neural network. Indeed, it is a light convolutional network that offers good accuracy, allowing real-time use even on embedded systems such as Nvidia Jetson *TX*2. YOLOv5 offers a significant performance gain compared to YOLOv5 thanks to an improved network architecture as well as innovations on data augmentation during training. The single-stage

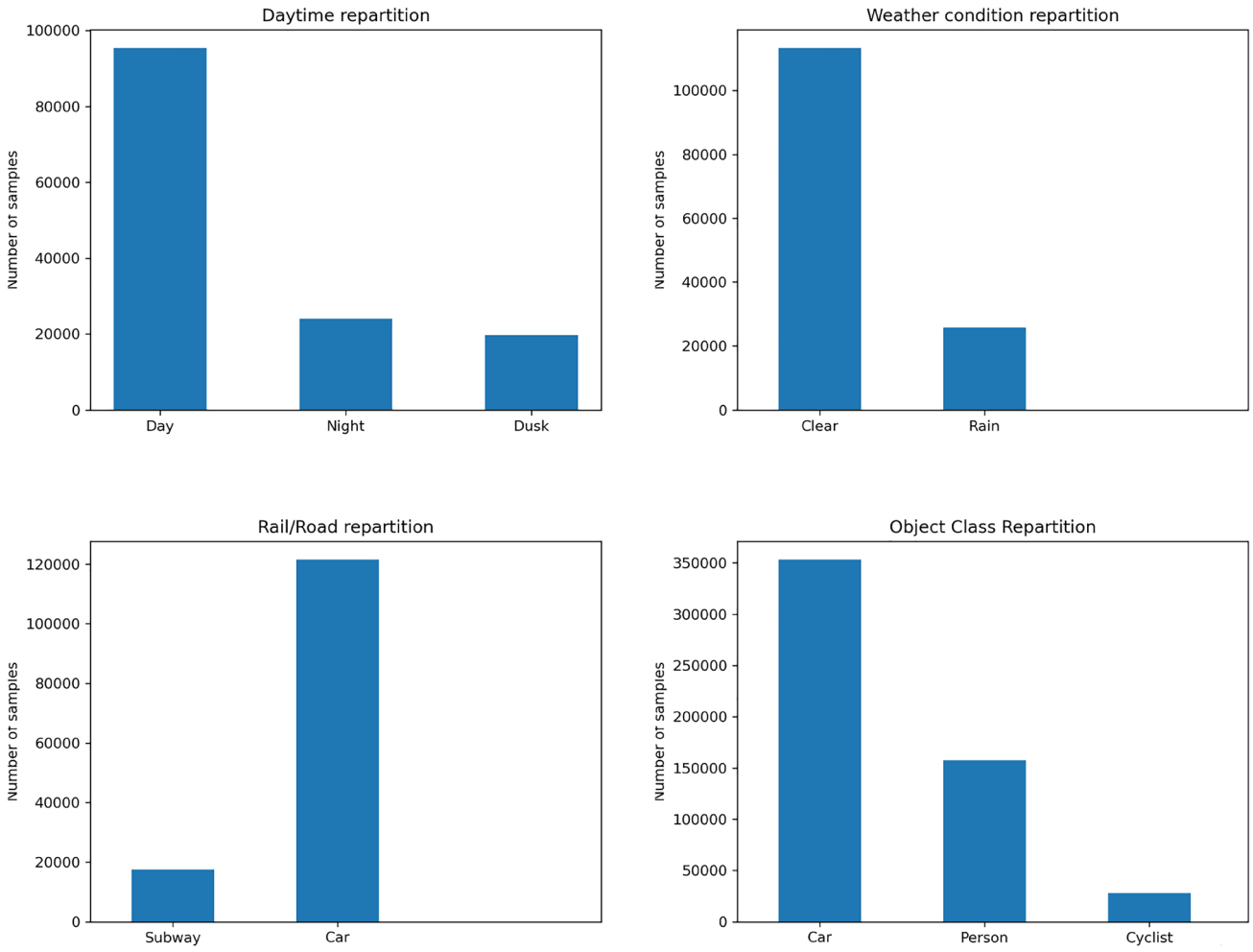

Fig. 1 Statistic details for our new dataset: Top Left is the image daytime distribution, Top Right is the weather condition distribution, Bottom Left is the Road/Railway distribution, Bottom Right is the class distribution

Fig. 2 Images with 3D bounding box ground truths. Our dataset presents a variety of environments and conditions. From top left to bottom right: road with clear weather, Road at night, Road with rainy weather and Rail with clear weather. The ground truths shown are for the same classes as KITTI: Car, Person and Cycle

design of our network allows for better computational times compared to multiple stage methods [19, 21, 23] and it can be trained end-to-end. Another problem posed by multiple-stage methods was the degradation in accuracy for the 3D parameters regression when the predicted RoIs used for feature alignment vary from the ones used in the training [33]. Our single-stage method uses hybrid anchor boxes to directly regress the 2D bounding box as well as the 3D parameters. We thus avoid the problem related to the fluctuation of the accuracy when the ROIs are not fixed. This new approach is the fastest method for 3D prediction from images without sacrificing accuracy. Figure 3 shows an overview of our 3D multi-object detection method.

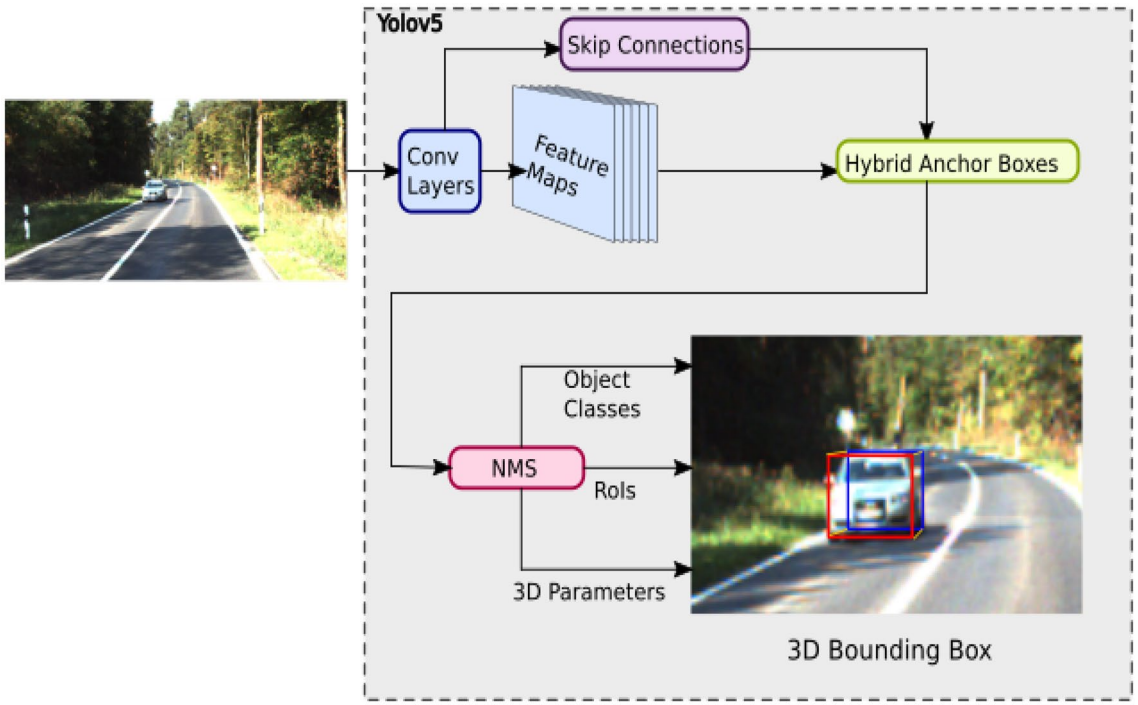

Fig. 3 Overview of our method for 3D multi-object detection. A single RGB image is used as input. We leverage our Hybrid anchor boxes to predict both the 2D and the 3D bounding boxes. Non-Maximum-Suppression is used to filter the predictions. Among the 3D parameters that we predict, we have the projected 3D center on the image plane, the distance of the object, its dimensions, and finally its orientation

4.1 3D bounding box estimation

The 3D bounding box of an object can be described by its center position from the camera $\mathbf{T} = [x\ y\ z]^\top$, its dimensions $\mathbf{D} = [w, h, l]$ (with $\mathbf{w}, \mathbf{h}, \mathbf{l}$ respectively width, height and length) and its orientation $\mathbf{R}(\phi, \theta, \psi)$ (with ϕ, θ, ψ respectively the elevation, azimuth, and roll angles). Given $\mathbf{K}$ the matrix of intrinsic parameters of the camera and $\mathbf{X_o} = [x_o\ y_o\ z_o\ 1]^\top$ a 3D point in the object coordinate system, the re-projection of this point into the image plane $\mathbf{x_{im}} = [u\ v\ 1]^\top$ is given by:

$$\mathbf{x_{im}} = \mathbf{K}.\left[\mathbf{R}\ \mathbf{T}\right].\mathbf{X_o}, \tag{1}$$

where $[RT]$ represents the extrinsic matrix with R (3×3 matrix) the rotation and T (3×1) the translation. It is an extrinsic matrix integrating Translation (T) and Rotation (R). By considering that the center of the 3D bounding box is the origin of the object coordinates, the coordinates of the 3D bounding box are: $\mathbf{X_1} = \left[w/2\ \ h/2\ \ l/2\right]$, $\mathbf{X_2} = \left[w/2\ \ -h/2\ \ l/2\right]$, ..., $\mathbf{X_8} = \left[-w/2\ \ -h/2\ \ -l/2\right]$. The 3D bounding box coordinates in the image can then be obtained using Equation (1).

4.2 Parameters to regress

To better compare the performances of multi-stage approaches with single ones for 3D object detection, we predict similar parameters as it was presented in our previous multi-stage approach [33].

4.2.1 2D object detection

In our work, we use YOLOv5 to perform 2D object detection. YOLOv5 performs the object class classification *cls* as well as the bounding box position regression *b*. The bounding box *b* is then used as prior information for predicting the object center and the object class *cls* is used in the object dimension prediction.

4.2.2 Object center prediction

In this work, we assume that the 3D center of the object is the center of the 3D bounding box. Predicting the center of the 3D bounding boxes of the objects is, therefore, equivalent to predicting their position $\mathbf{X_o} = [x_o\ y_o\ z_o\ 1]^\top$. But instead of directly predicting the coordinates of the 3D object center on the image plane, we use cues from the 2D bounding box estimation. We predict the offset position in pixels of the projected object center from the 2D bounding box center $\tilde{\mathbf{c}} = \left[\tilde{cx}\ \ \tilde{cy}\right]$. The variance of the prediction is thus reduced, making it easier to be learned by the algorithm. The object center $\mathbf{X_o}$ is then computed using the object distance estimation on the Z-axis and the inverted calibration matrix $\mathbf{K^{-1}}$.

4.2.3 Object distance estimation

To obtain the center of the 3D bounding box, it is necessary to get its position on the Z-axis of the camera coordinates. For each ROIs from the object detector, we predict the object's center distance $\tilde{z}$ in meters.

4.2.4 Object dimensions

Instead of directly predicting the object's dimensions, we exploit the fact that those dimensions have a very low variance within the same class (car, truck, etc). Therefore, we choose to use the average dimensions of each object class as a strong prior for the estimation of the dimensions.

4.2.5 Object orientation

In our work, we assume that only the azimuth (noted θ) matters for application on the road environment, and thus we do not predict the orientation characterized by the elevation and roll and we set $\phi = 0$ and $\psi = 0$. Since the same azimuth can lead to multiple observed orientations from the camera point of view (see Fig. 4), we cannot directly predict the angle θ. Instead, we predict the observed angle α and retrieve the global orientation θ using Eq. (2):

$$\theta = \alpha + arctan(\frac{x}{z}). \tag{2}$$

Following the work described in [19], instead of considering the angle prediction as a regression task, we formulate is as a hybrid classification/regression problem in which the possible angles are separated into two bins of which the values are centered on 90 for the first one and −90 for the second one. We then perform a classification to predict in which bin the object angle is located. The parameter which is regressed is the difference between the bin center and α.

4.3 Hybrid anchor boxes for 3D detection

To take advantage of the innovations brought by single-stage approaches, we have modified the YOLOv5 anchor boxes to create hybrid anchor boxes to perform the regression of the 2D bounding boxes and the 3D parameters. We predict three grids of features at different sizes. In each grid, three anchor boxes are predicted. These boxes contain the predictions of the classes of the objects, the relative position of the bounding box from the center of the grid, the probability that an object is present, and finally the 3D bounding box parameters (projected center, distance, dimensions, orientation).

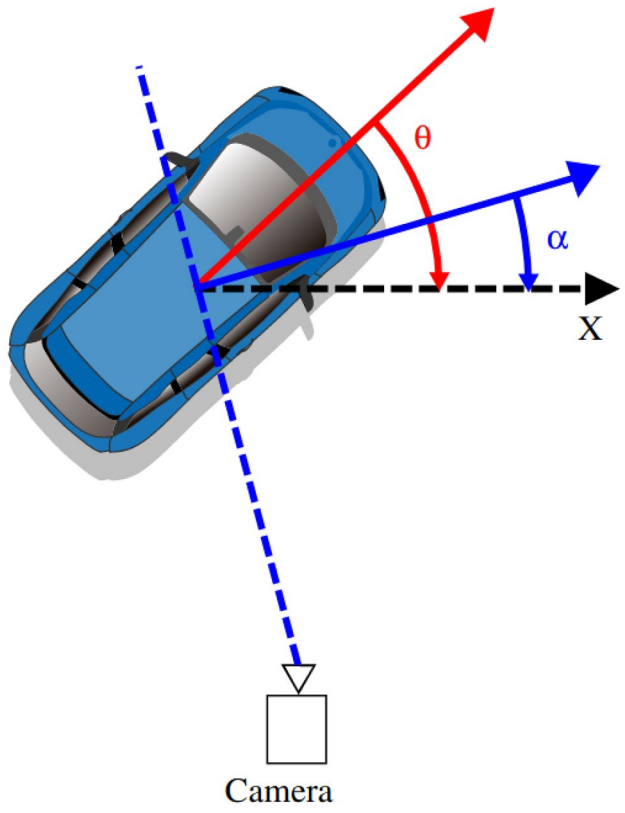

Fig. 4 Illustration of the object azimuth θ and its observed orientation α. The local orientation is retrieved by computing the angle between the normal to the ray between the camera and the object center and the X-axis of the camera. Given that we use left-hand coordinates, the rotation is clockwise. Our method estimates the observed orientation and θ can be obtained using Eq. (2)

The output of our hybrid anchor boxes for our 3D object detector is detailed in Eq. 3. Let y be the output value of one of our new anchor boxes:

$$\mathbf{y} = \left[\mathbf{b_{off}}\ obj\ \mathbf{cls}\ \mathbf{c_{off}}\ Z\ \mathbf{D}\ \mathbf{O}.\right] \tag{3}$$

With $\mathbf{b_{off}} = \left[x\ y\ w\ h\right]$ the 2D bounding box offset prediction, obj is the object prediction, $\mathbf{cls} = \left[cls_1\ \ldots\ cls_N\right]$ the score for the N classes, $\mathbf{c_{off}} = \left[x_c\ y_c\right]$ the prediction of the offset between the projected 3D center and the center of the bounding box in pixels, Z is the position on the Z-axis of the 3D center of the object in meters, $\mathbf{D} = \left[W_1\ H_1\ L_1\ \ldots\ W_N\ H_N\ L_N\right]$ is the predicted dimensions offset in meters for each N class. The predicted offset is between the object's dimension and its class mean dimensions. Finally we have the orientation prediction $\mathbf{O} = \left[bin_1\ \overline{bin_1}\ sin_1\ cos_1\ bin_2\ \overline{bin_2}\ sin_2\ cos_2\right]$, where bin_1 and bin_2 represent the predicted probability that $\alpha \in \left[-195\ 15\right]$ and $\alpha \in \left[-15\ 105\right]$, respectively with α the observed orientation of the object. And $sin_{i\in\{1,2\}}\ cos_{i\in\{1,2\}}$ are the sinus and cosine of the observed angle offset regarding the center of the ith bin. Our bin centers are 90° and −90°, respectively. By adding the prediction of the 3D parameters inside the anchor boxes, our method learns to predict the distance of the objects, the position of their centroid, their dimensions, and their azimuth. With these parameters, we are then able to draw the 3D bounding boxes of the objects. All the anchor boxes of the different grids are then merged and re-projected onto the image and a Non-Maximum-Suppression function is used to filter the anchor boxes. This new approach taking advantage of anchor boxes allows our method to do 3D bounding box prediction in a single stage, thus reducing the size of our network without sacrificing accuracy. Another notable advantage over other multiple stage 3D detection methods is that it avoids the problem of deteriorating accuracy of 3D detection when the predicted ROIs for objects vary, especially when aligning features for 3D regression.

4.4 Losses

In this subsection, we present the loss calculation of our method. The loss is specified in Eq. (4):

$$L = L_{yolo} + k_1.L_{center} + k_2.L_{distance} + k_3.L_{dim} + k_4.L_{orient}, \tag{4}$$

where $\mathbf{L_{yolo}}$ is the loss for 2D detection and class prediction, and $\mathbf{L_{center}}$, $\mathbf{L_{distance}}$, $\mathbf{L_{dim}}$ and $\mathbf{L_{orient}}$ are the losses for the 3D object center, distance, dimensions and orientation, respectively. $k_{i\in[1,\ldots,4]}$ are the weights of the different losses. For $\mathbf{L_{yolo}}$ loss, we use the same as YOLOv5. With $\mathbf{c_k} = \left[cx^k\ cy^k\right]$ the projection on image plane of 3D object k ground-truth center in pixels, $\mathbf{Rc_k} = \left[Rcx_k\ Rcy_k\right]$ the center of the corresponding ROI, N the number of objects and $\tilde{\mathbf{c}}_\mathbf{k}$ the prediction from our network, the center loss is defined by Equation (5):

$$L_{center} = Mean(\frac{1}{N}\sum_{k=1}^{N}\left|\mathbf{c_k} - \mathbf{Rc_k} - \tilde{\mathbf{c}}_\mathbf{k}\right|). \tag{5}$$

Assuming z_k the ground-truth distance of an object k, N the total number of objects and $\tilde{z}_k$ the distance prediction from our method, the distance loss is then calculated using the L1 loss and is detailed in Eq. (6):

$$L_{distance} = \frac{1}{N}\sum_{k=1}^{N}\left|z_k - \tilde{z}_k\right|. \tag{6}$$

With $\mathbf{d_k} = \left[\mathrm{d}x\ \mathrm{d}y\ \mathrm{d}z\right]$ the ground-truth dimensions of an object k in meters, dd_k the mean dimension for the class of object k, N the number of objects and $\tilde{d}_k$ the prediction from our method, the distance loss is detailed in Eq. (7):

$$L_{dim} = Mean(\frac{1}{N}\sum_{k=1}^{N}|\mathbf{d_k} - \tilde{\mathbf{d}}_\mathbf{k} - \mathbf{dd_k}|). \quad (7)$$

Finally, assuming α_k the ground-truth observed angle of object k, N the total number of objects and $\tilde{\alpha}_k$ the prediction, we use the smooth L1 loss as the orientation loss. The loss is written in Eq. (8) with $\beta = 1$:

$$L_{orient} = \frac{1}{N}\sum_{k=1}^{N} e_k, \quad (8)$$

where

$$e_k = \begin{cases} 0.5(\alpha_k - \tilde{\alpha}_k)^2/\beta, & \text{if } |\alpha_k - \tilde{\alpha}_k| < \beta \\ |\alpha_k - \tilde{\alpha}_k| - 0.5 \times \beta, & \text{otherwise} \end{cases}.$$

5 Experimental results

5.1 Training details

5.1.1 Model size

Our method can be presented in different versions with a different size for each, such as in YOLOv5. The depth and width of the network vary with the versions which result in several different learnable parameters. Thus our model L-CNN comes in 3 versions: Large, Medium, and Small. To analyze the performance in terms of 3D detection accuracy and computation time of each version, we have trained and evaluated each of them on the KITTI and GTA datasets.

5.1.2 Split attention model

Among the convolutional blocks composing the network, the bottleneck introduced in [34] is extensively used in YOLOv5. Improvements to these convolutional blocks have recently been introduced in [35] along with Split Attention convolutions. These new Split Attention Bottlenecks increase the performance of networks at the cost of a slight increase in computation time. To create a model with a good compromise between accuracy and computation time, we have modified the small model by replacing the bottlenecks with Split Attention Bottlenecks. This new model, called Small SA (Split Attention), is dedicated to applications on systems with limited computational resources, such as the Jetson *TX*2 embedded boards, without sacrificing prediction accuracy.

5.1.3 KITTI

We trained our method on the KITTI dataset dedicated to 3D detection on the same training and validation split defined in [36]. The training split contains 3712 images and the validation split 3769 images. To speed up the training process and improve accuracy, we trained our models with either pre-trained weights from YOLOv5 or pre-trained weights from our model after training for 15 epochs on our GTAV dataset. We demonstrate that pre-training our model on our dataset significantly improves the accuracy of our method on the KITTI dataset. Furthermore, as in [27] we perform the training with both the left and right images of the training split of KITTI thus doubling the number of images available for the training resulting in improved performances of our method during evaluation. To make our method compatible with embedded boards such as the Nvidia Jetson *TX*2, we have done the training with a lowered resolution of 672×224. We also trained with a resolution of 1312×416, closer to the original resolution, to compare the accuracy of our method with the state-of-the-art method on the KITTI dataset.

5.1.4 GTA

Inspired by previous work on dataset creation [16] using images from the video game Grand Theft Auto V, we created our hybrid dataset of road and rail images. This new dataset allows us to overcome the problem of having a railway dataset with a ground truth rich enough to allow 3D bounding box learning for cars, pedestrians, and motorcycles. The training of our method was conducted on a split containing road and railway images (107 K images). The evaluation was then conducted on the validation split of the dataset containing 11,630 images. The training on this dataset was performed on 50 epochs.

5.1.5 Data augmentation

Data augmentation is very useful to improve performance of CNN models through the construction of new and different examples for training. Even though KITTI and our GTA-V dataset presents a variety of environments and conditions (in GTA-V dataset for example, we have different situations like road with clear weather, road at night, road with rainy weather, and rail with clear weather as it shown in Fig. 1), the risk of overfitting our models is still present. Data augmentation gives multiple advantages: 1. Reduce overfitting, 2. Increases the generalisation capabilities of the trained models thus yielding better results in real-world environment. This is a good way to introduce "non-linearity" into the dataset model making it more similar to the real world.

We use Mosaic data augmentation as defined in YOLOv4 [37] which consists in mixing 4 training images to improve the understanding of the spatial context of objects for our method. We also applied translation and scaling transformations to the images as further augmentation.

5.1.6 Hyperparameters

The hyperparameters of our models determined after many iterations of short training sessions. We use the Adam optimizer with a cyclic learning rate scheduler as described in [38]. The maximum learning rate was set to $9.4e-4$ with a final learning rate of $1.8e-5$. Finally, we use gradient accumulation to perform training with an effective batch size of 64.

5.1.7 Loss weights

Since our method is trained end-to-end, our loss function combines the loss of each task. To obtain optimal results it is necessary to find the weight ($k1$, ..., $k4$) of each loss. Through multiple trials we found $k1 = 0.005$, $k2 = 0.2$, $k3 = 0.0176$, $k4 = 0.01$. To avoid overfitting for 2D detection, we imposed the condition $L_{yolo} < 0.1$ which when satisfied L_{yolo} is ignored in the optimization loop.

5.2 Evaluation

In this subsection, we present the metrics that will be used in the evaluation of our models.

5.2.1 3D average precision

For evaluation on the KITTI dataset, we used the official benchmark metrics such as the 3D Average Precision (AP) with 40 recall positions as presented by [39]. The official KITTI benchmark uses an IOU threshold of 0.7 for the car class and a 0.5 threshold for the person and cycle classes. We also added evaluation results for the car class with a 0.5 threshold.

5.2.2 2D detection

To evaluate the performance of the 2D detection, we used the metrics described by the authors of YOLOv5 on each class of the dataset. Given that the regression of the 3D bounding box parameters relies on accurate RoI and classes for the objects, evaluating the precision of the 2D detector is a necessity. The error metrics are the Average Precision (AP), the Recall (R), and the mean Average Precision (mAP_{50}).

5.2.3 Distance estimation

For our distance estimator evaluation, we used the same evaluation as the ones used for image-level depth estimation methods. The metrics used include Absolute Relative Error (Abs Rel), the Squared Relative Error (SRE), the Root Mean Square Error (RMSE), the logarithmic RMSE (log RMSE), and the percentage of Bad Matching Pixels. Let z_{gt} and z_{pd} be, respectively, the ground truth and predicted distance of the object i, it is calculated using Eq. (9), where $\delta = 1.25^k$:

$$\alpha_{k=[1..3]} = max(\frac{z_{gt}}{z_{pd}}, \frac{z_{pd}}{z_{gt}}) < \delta^k. \tag{9}$$

5.2.4 Dimensions

The dimension prediction evaluation is performed using the Dimension Score (DS) described by the authors of [23]. With V_{pd} and V_{gt} the predicted and ground truth volume of the object, the DS is computed using Eq. (10) averaged across all examples:

$$DS = min(\frac{V_{pd}}{V_{gt}}, \frac{V_{gt}}{V_{pd}}). \tag{10}$$

5.2.5 Object center

The object center predictions were evaluated with the Center Score (CS) as described in [23]. Assuming x and y the projected center coordinates in pixels and w and h the width and the height of the 2D bounding box, CS is computed with Eq. (11):

$$CS = (2 + cos(\frac{x_{gt} - x_{pd}}{w_{pd}}) + cos(\frac{y_{gt} - y_{pd}}{h_{pd}}))/4. \tag{11}$$

5.2.6 Orientation

To evaluate the orientation predictions, we use the Orientation Score (OS) averaged across all examples. OS is calculated using Equation (12):

$$OS = (1 + cos(\alpha_{gt} - \alpha_{pd}))/2. \tag{12}$$

5.2.7 Computation time

To evaluate the computation time of our models, we compute the average time necessary for a forward pass (including Non-Max-Suppression) for one single image during inference. We tested all the methods presented on an Nvidia RTX 3080 GPU. We also included computation time on an embedded Nvidia Jetson *TX*2 board as shown in Table 4.

5.3 Result analysis

We present the quantitative results of our models on the KITTI dataset using the official 3D AP metric in Table 1 for the Car class, Table 2 for the Person class, and Table 3

Table 1 Quantitative results for the Car class from our evaluation on the KITTI dataset. We can see that our method benefits from being pre-trained on our GTA dataset and manages to outperform other state-of-the-art methods, especially for moderate and hard targets. The Car class on the KITTI dataset represents ∼ 82% of all the dataset objects

Method	Pretrained on GTA	Image size	[val] IOU 0.7			[val] IOU 0.5			Time/img (ms)	Memory consumption
			Easy	Mod	Hard	Easy	Mod	Hard		
Mono3D [17]		–	2.53	2.31	2.31	25.19	18.20	15.52	–	–
Multi-fusion [22]		1696x512	10.53	5.69	5.39	47.88	29.48	26.44	–	–
M3D-RPN [26]		1696× 512	20.27	17.06	15.21	48.96	39.57	33.01	78.3	5 GB
GAM3D [27]		1280× 288	**23.63**	16.16	12.06	60.92	42.18	32.02	34.2	2.1 GB
Ours (small)		672× 224	7.29	5.48	5.34	44.35	31.80	29.91	1.5	1.6 GB
		1312× 416	15.52	13.27	13.19	48.24	36.14	31.78	4.4	1.7 GB
	×	1312× 416	15.59	13.50	13.14	49.71	38.02	32.70	4.4	1.7 GB
Ours (small SA)		672× 224	13.79	11.72	11.39	44.51	32.47	27.26	3.1	**1.5 GB**
		1312× 416	10.34	10.07	9.85	23.22	20.22	19.56	6.1	1.6 GB
	×	1312× 416	15.57	13.32	13.39	51.33	38.09	33.31	6.1	1.6 GB
Ours (medium)		672× 224	9.42	7.96	6.52	46.22	34.66	33.01	2.7	1.7 GB
		1312× 416	17.96	14.27	13.73	51.36	38.28	37.41	8.2	1.9 GB
	×	1312× 416	17.63	15.04	14.68	53.95	44.73	39.86	8.2	1.9 GB
Ours (large)		672× 224	12.52	10.52	9.75	53.27	40.58	35.34	4	2 GB
		1312× 416	21.07	16.07	15.68	55.69	41.89	40.46	11.2	2.15 GB
	×	1312× 416	23.26	**18.46**	**16.16**	**60.93**	**48.45**	**42.72**	11.2	2.15 GB

Table 2 Quantitative results for the Person class from our evaluation on the KITTI dataset. We can see that our method benefits from being pre-trained on our GTA dataset. The Person class on the KITTI dataset represents ∼ 13% of all the dataset objects.

Method	Pretrained on GTA	Image size	[val] IOU 0.5			Time/img (ms)	Memory consumption	# Parameters
			Easy	Mod	Hard			
Ours (small)		672× 224	5.30	4.53	4.59	**1.5**	1.6 GB	
		1312× 416	4.66	4.23	3.95	4.4	1.7 GB	7.3 M
	×	1312× 416	6.91	5.45	5.03	4.4	1.7 GB	
Ours (small SA)		672× 224	9.96	9.09	9.09	3.1	**1.5 GB**	
		1312× 416	4.55	4.55	4.55	6.1	1.6 GB	9.7M
	×	1312× 416	11.75	10.85	10.56	6.1	1.6 GB	
Ours (medium)		672× 224	6.51	6.18	5.90	2.7	1.7 GB	
		1312× 416	7.94	5.93	5.44	8.2	1.9 GB	21.6M
	×	1312× 416	8.66	6.19	6.01	8.2	1.9 GB	
Ours (large)		672× 224	**11.76**	**11.36**	**10.62**	4	2 GB	
		1312x416	8.55	7.00	6.77	11.2	2.15 GB	47.5M
	x	1312x416	11.58	8.59	7.99	11.2	2.15 GB	

for the Cycle class. We present our results along with state-of-the-art methods for comparison. These results show that our approach, although less complex than state-of-the-art methods, has accuracy comparable or even superior to them. Our model is also significantly faster than all other tested approaches, with a computation time of 11.2ms per image on our Large model when inferring on an RTX 3080 GPU. Our method is even able to achieve higher accuracy using pre-trained weights on our GTAV dataset, which shows the potential of video game datasets to improve 3D object detection methods on real datasets. Thanks to this, our "Large" model can outperform state-of-the-art methods in moderate and difficult scenarios. Finally, we also conducted experiments on a Jetson *TX*2 embedded board to confirm the applicability of our smallest model for real-time tasks for embedded applications. We present the results in Table 4.

Table 3 Quantitative results for the Cycle class from our evaluation on the KITTI dataset. We can see that our method benefits from being pre-trained on our GTA dataset. The Cycle class on the KITTI dataset represents ~ 5% of all the dataset objects

Method	Pretrained on GTA	Image size	[val]			Time/img (ms)	Memory consumption	# Parameters
			IOU 0.5					
			Easy	Mod	Hard			
Ours (small)		672 ×224	3.68	3.12	2.27	**1.5**	1.6 GB	
		1312 ×416	4.66	4.23	3.95	4.4	1.7 GB	7.3M
	×	1312 ×416	14.56	11.83	11.64	4.4	1.7 GB	
Ours (small SA)		672 ×224	4.39	2.65	2.20	3.1	**1.5 GB**	
		1312 × 416	0.98	0.88	0.82	6.1	1.6 GB	9.7M
	×	1312 × 416	9.39	6.39	6.53	6.1	1.6 GB	
Ours (medium)		672 × 224	13.34	10.37	10.39	2.7	1.7 GB	
		1312 × 416	10.75	7.65	7.53	8.2	1.9 GB	21.6 M
	×	1312 × 416	18.12	**13.58**	**12.02**	8.2	1.9 GB	
Ours (large)		672 × 224	12.88	10.26	10.31	4	2 GB	
		1312 × 416	**14.88**	10.33	8.21	11.2	2.15 GB	47.5 M
	×	1312 × 416	11.08	5.94	5.71	11.2	2.15 GB	

Table 4 Computing time on the Nvidia Jetson *TX*2 embedded board. Our results show that our new Small SA model is the most appropriate for real-time embedded applications with 60*ms*/*img* (17 fps)

Method	Image size	Time/img (ms)	# Parameters
Ours (small)	672 × 224	70	7.3 M
	1312 × 416	130	
Ours (small SA)	672 × 224	**60**	9.7M
	1312 × 416	200	
Ours (medium)	672 × 224	143	21.6M
	1312 × 416	359	
Ours (large)	672 × 224	199	47.5M
	1312 × 416	613	

Our new SA Small model results show that it performs the same, or worse when not using pre-trained weights, than the Small model when training at full resolution (1312 × 416). However, when using a smaller resolution (672 × 224) it significantly outperforms the Small model. It is a particularly interesting model for applications on systems where the computational capabilities are limited since its memory requirements are lower and it runs at 3.1*ms*/*img* while having greater accuracy than the Small and Medium models trained at the same resolution. Our tests with the Jetson *TX*2 also show that it is the fastest of our methods on this platform. The quantitative results of our models on both datasets for all classes along with the results of our baseline based on our previous multi-stage approach based on YOLOv5 are presented in Table 5 and in Fig. 5. We can see that we achieved significant accuracy gains by switching from a multi-stage approach based on YOLOv3 [40] from our previous approach [33] to an anchor-based approach using YOLOv5. We conducted the experiments using either the ROIs from YOLOv3 predictions or the ground truth for the feature alignment necessary for the regression of the 3D bounding box parameters. This to highlight the problem posed by ROIs variation in the 3D parameter regression in multi-stage approaches. We show that our new single-stage approach significantly outperforms our previous method in both cases, especially for depth estimation and orientation.

We also showcase the various models from our current approach (small, SA small, medium, large) and we can see that the precision between the different models for 2D detection, Dimensions, Center, and Orientation does not significantly benefit from the heavier models while Distance estimation improves when using larger models. Qualitative results on both KITTI and GTA datasets are presented in Fig. 6.

6 Conclusion and outlook

In this paper, we presented our new anchor-based 3D object detection method from single-image input using a modified YOLOv5 network. Along with this method, we also presented our new virtual dataset based on the video game GTAV for both road and rail environments, filling the gap in railway datasets for 3D object detection and localization. Through our work, we showed that state-of-the-art precision on the KITTI benchmark could be reached by adopting transfer learning from models trained on our new dataset while being the fastest method available for 3D object detection from images. Our approach is particularly well suited for embedded use, as demonstrated by our tests on the Nvidia Jetson *TX*2 embedded board. We also provided

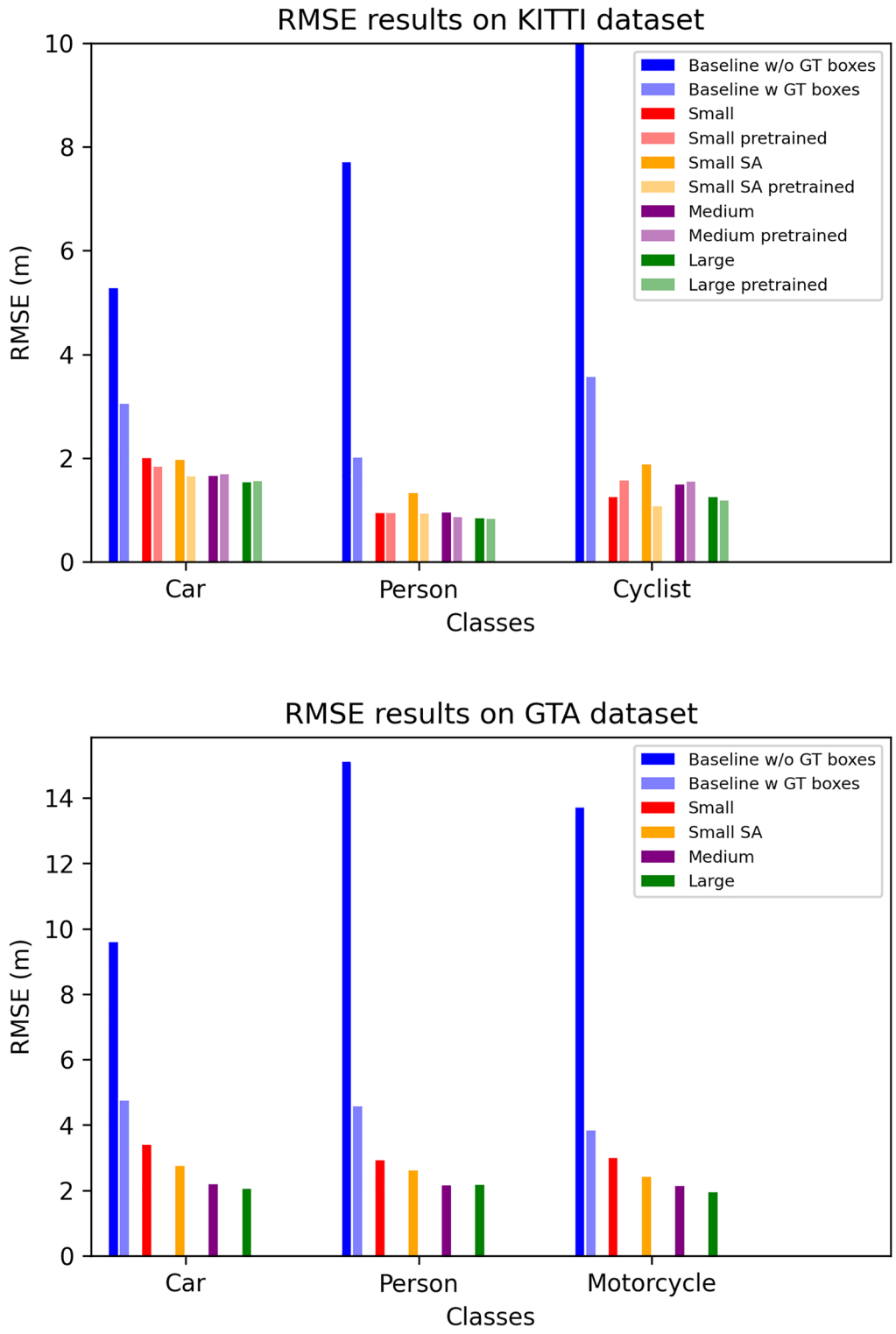

Fig. 5 Depth RMSE error obtained by our different models along the baseline multi-stage [33] method for each class on the KITTI and GTA datasets. Our new method improves accuracy on the KITTI dataset significantly when we use weights pre-trained on our new GTA dataset. With these weights, our new model with Split Attention (SA) convolutions is able to reach the same performances as our medium-sized model

Table 5 Quantitative results from our evaluation on both the KITTI and our GTA datasets. Results from our previous method are displayed along with our new method. Our new method, significantly outperforms our previous approach, especially in terms of depth estimation accuracy

Dataset	Classes	Method	2D detection			Distance							Dimensions	Center	Orientation
			AP	R	mAP@0.5	Abs Rel	SRE	RMSE (m)	log RMSE	α_1	α_2	α_3	DS	CS	OS
KITTI	Car	Baseline (w GT RoIs)	–	–	–	0.096	0.312	3.05	0.1800	0.941	0.980	0.988	0.872	**0.981**	0.781
		Baseline (w/o GT RoIs)	0.558	**0.879**	0.783	0.163	1.290	5.28	0.2710	0.839	0.948	0.971	0.867	0.962	0.783
		Ours (small)	0.886	0.814	0.790	0.051	0.109	2.00	0.0782	0.987	0.997	**0.999**	0.879	0.960	0.907
		Ours (small + GTA)	0.882	0.818	**0.796**	0.050	0.099	1.83	0.0757	0.988	0.997	**0.999**	0.882	0.961	0.935
		Ours (small SA)	0.880	0.764	0.739	0.062	0.133	1.97	0.0901	0.981	0.997	**0.999**	0.874	0.956	0.859
		Ours (small SA + GTA)	0.901	0.777	0.755	0.047	0.086	1.65	0.0733	0.991	**0.998**	**0.999**	0.883	0.960	0.908
		Ours (medium)	0.892	0.800	0.777	0.047	0.085	1.66	0.0755	0.988	0.997	**0.999**	0.883	0.963	0.929
		Ours (medium + GTA)	0.887	0.822	0.798	0.045	0.083	1.69	0.0704	0.989	**0.998**	**0.999**	0.883	0.963	0.938
		Ours (large)	**0.908**	0.787	0.787	0.042	0.072	**1.53**	0.0670	**0.992**	**0.998**	**0.999**	**0.885**	0.964	**0.947**
		Ours (large + GTA)	0.903	0.798	0.777	**0.041**	**0.070**	1.56	**0.0657**	**0.992**	**0.998**	**0.999**	**0.885**	0.964	0.946
	Person	Baseline (w GT RoIs)	–	–	–	0.098	0.254	2.01	0.1540	0.935	0.983	0.992	0.705	**0.993**	0.584
		Baseline (w/o GT RoIs)	0.508	0.518	0.464	0.424	7.920	7.70	0.4850	0.729	0.844	0.892	0.719	0.885	0.616
		Ours (small)	0.817	0.528	0.514	0.057	0.068	0.94	0.0800	0.979	0.998	**1**	0.729	0.949	0.750
		Ours (small + GTA)	0.740	0.554	0.525	0.054	0.062	0.94	0.0762	0.980	0.999	**1**	0.736	0.948	0.745
		Ours (small SA)	0.773	0.435	0.419	0.086	0.136	1.33	0.1050	0.964	0.999	**1**	0.682	0.945	0.581
		Ours (small SA + GTA)	0.821	0.436	0.427	0.059	0.067	0.92	0.0803	0.981	0.999	**1**	0.723	0.944	0.716
		Ours (medium)	0.828	0.527	**0.527**	0.049	0.065	0.95	0.0745	0.981	0.996	**1**	0.709	0.949	0.815
		Ours (medium + GTA)	0.825	0.532	0.517	0.051	0.055	0.86	0.0731	0.982	**1**	**1**	0.733	0.950	**0.818**
		Ours (large)	**0.851**	0.514	0.501	0.048	0.050	0.83	0.0689	0.987	0.999	**1**	0.722	0.951	0.812
		Ours (large + GTA)	0.817	**0.539**	0.519	**0.046**	**0.049**	**0.82**	**0.0667**	**0.993**	**1**	**1**	**0.736**	0.950	0.772
	Cyclist	Baseline (w GT RoIs)	–	–	–	0.098	0.359	3.57	0.1500	0.940	0.979	0.988	0.809	**0.997**	0.738
		Baseline (w/o GT RoIs)	0.342	0.371	0.253	1.040	30.900	17.40	0.7970	0.415	0.622	0.719	0.812	0.678	0.542
		Ours (small)	0.568	0.400	0.342	0.058	0.075	1.25	0.0755	0.989	**1**	**1**	0.787	0.960	0.851
		Ours (small + GTA)	0.589	0.403	0.364	0.056	0.099	1.57	0.0780	0.986	0.997	**1**	**0.827**	0.964	0.857
		Ours (small SA)	0.490	0.274	0.215	0.099	0.205	1.88	0.1140	0.959	**1**	**1**	0.800	0.952	0.868
		Ours (small SA + GTA)	0.655	0.348	0.321	0.053	0.072	1.07	0.0746	0.987	0.997	**1**	0.819	0.961	0.919
		Ours (medium)	0.692	0.476	0.436	0.051	0.083	1.49	0.0664	0.995	**1**	**1**	0.824	0.962	0.865
		Ours (medium + GTA)	0.654	**0.492**	**0.460**	0.049	0.083	1.54	0.0665	0.995	**1**	**1**	0.822	0.961	0.888
		Ours (large)	**0.780**	0.429	0.407	**0.042**	0.059	1.25	**0.0571**	**1**	**1**	**1**	0.825	0.963	0.878
		Ours (large + GTA)	0.629	0.448	0.404	0.046	**0.058**	**1.18**	0.0622	0.998	0.998	**1**	0.808	0.965	**0.913**

Table 5 (continued)

Dataset	Classes	Method	2D detection			Distance							Dimensions	Center	Orientation
			AP	R	mAP@0.5	Abs Rel	SRE	RMSE (m)	log RMSE	α_1	α_2	α_3	DS	CS	OS
GTA	Car	Baseline (w GT RoIs)	–	–	–	0.055	0.385	4.74	0.0761	0.990	0.999	0.999	0.860	**0.999**	0.993
		Baseline (w/o GT RoIs)	0.477	0.915	0.778	0.129	2.280	9.59	0.2170	0.873	0.938	0.965	0.812	0.894	0.905
		Ours (small)	0.921	0.942	0.957	0.062	0.222	3.40	0.0749	0.998	**1**	**1**	**0.876**	0.982	**0.997**
		Ours (small SA)	0.906	0.937	0.947	0.041	0.134	2.75	0.0542	0.998	**1**	**1**	0.873	0.980	0.995
		Ours (medium)	**0.930**	0.951	**0.964**	0.033	0.086	2.18	0.0440	**0.999**	**1**	**1**	0.865	0.984	**0.997**
		Ours (large)	**0.930**	0.905	0.919	**0.029**	**0.075**	**2.05**	**0.0402**	**0.999**	**1**	**1**	0.867	0.984	**0.997**
	Motorcycle	Baseline (w GT RoIs)	–	–	–	0.062	0.266	3.84	0.0753	**1**	**1**	**1**	**0.918**	**1.00**	0.967
		Baseline (w/o GT RoIs)	0.614	0.764	0.725	0.186	3.580	13.70	0.2810	0.691	0.878	0.967	0.901	0.689	0.909
		Ours (small)	0.937	0.925	0.935	0.061	0.190	2.99	0.0755	0.998	**1**	**1**	0.877	0.976	0.993
		Ours (small SA)	0.935	0.911	0.918	0.041	0.116	2.42	0.0548	0.998	**1**	**1**	0.871	0.975	0.993
		Ours (medium)	0.934	**0.938**	**0.944**	0.034	0.086	2.14	0.0463	0.998	**1**	**1**	0.876	0.980	**0.995**
		Ours (large)	**0.944**	0.933	0.943	**0.029**	**0.070**	**1.95**	**0.0414**	**0.999**	**1**	**1**	0.876	0.981	0.994
	Person	Baseline (w GT RoIs)	–	–	–	0.116	0.612	4.56	0.1590	0.877	0.966	1.00	**0.963**	**0.997**	0.856
		Baseline (w/o GT RoIs)	0.263	0.659	0.488	0.372	10.500	15.10	0.4530	0.620	0.834	0.903	0.961	0.812	0.735
		Ours (small)	**0.932**	0.814	0.834	0.049	0.170	2.92	0.0674	0.996	**1**	**1**	0.872	0.970	**0.957**
		Ours (small SA)	0.913	0.790	0.798	0.038	0.127	2.60	0.0538	**0.998**	**1**	**1**	0.868	0.970	0.920
		Ours (medium)	0.931	0.832	0.844	0.030	0.088	**2.16**	0.0452	0.997	**1**	**1**	0.878	0.974	0.955
		Ours (large)	0.929	**0.833**	**0.847**	**0.028**	**0.087**	2.17	**0.0435**	0.997	**1**	**1**	0.872	0.975	0.953

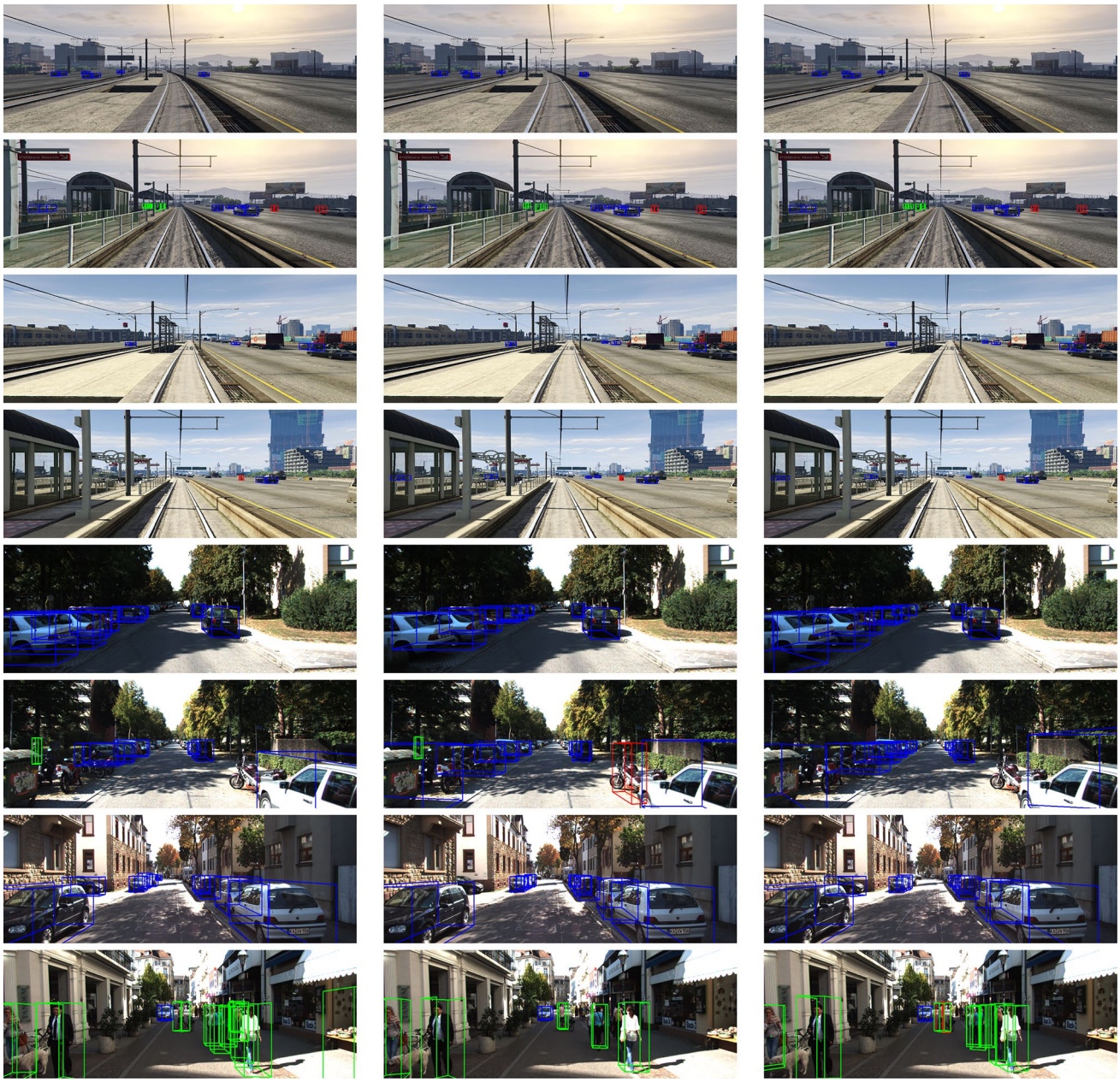

Fig. 6 Qualitative results of our methods on both KITTI and GTA datasets. These images are extracted from the validation split of each dataset. We display the results of both our best model (large pre-trained on GTAV) and our new lightweight model (L-CNN) for embedded platforms (Small SA pre-trained on GTAV). 4 top lines: results obtained under GTAV dataset (left column: ground truth, middle column: Small SA prediction, right column: large model prediction), 4 bottom lines: results obtained under KITTI dataset (left column: ground truth, middle column: Small SA prediction, right column: large model prediction)

an extensive evaluation of our approach, along with our previous 2 stage network on both the KITTI and our own GTAV road/railway virtual dataset. These results showed that our method is suitable for railway applications in a virtual environment, yet we still need to prove the applicability of our light-weight network under real conditions. Our fastest model (Small-SA) has also an important gap in precision with our largest model (Large). To address these issues, we first plan to acquire our railway dataset with real-world images (as a multimodal dataset provided by 2 cameras, 1 LiDAR, 1 Radar, 1 GPS/IMU) to further validate our approach. To improve the precision of our smallest model (Small-SA) we aim to use knowledge distillation to improve its precision and thus reduce the gap in precision without augmenting the computation time. We also aim to develop a lightweight 3D tracking algorithm and integrate it into our

method to provide a complete method for road and rail safety and further improve detection accuracy using Multi-Task Learning (MTL).

Acknowledgements This research is supported by SEGULA Technologies and the M2SINUM project (This project is co-financed by the European Union with the European Regional Development Fund (ERDF, 18*P*03390/18*E*01750/18*P*02733) and by the Haute-Normandie Regional Council via the M2SINUM project). We would like to thank SEGULA Technologies for their collaboration and the research engineers and technicians of the IRSEEM LNA (Autonomous Navigation Laboratory) for their support. Finally, this work was carried out in part on computing resources provided by CRIANN (Centre Régional Informatique et d'Applications Numériques de Normandie, Normandy, France).

References

1. Hong, D., Gao, L., Yokoya, N., Yao, J., Chanussot, J., Du, Q., Zhang, B.: More diverse means better: multimodal deep learning meets remote-sensing imagery classification. IEEE Trans. Geosci. Remote Sens. **59**(5), 4340–4354 (2020)
2. Hong, D., Han, Z., Yao, J., Gao, L., Zhang, B., Plaza, A., Chanussot, J.: Spectralformer: Rethinking hyperspectral image classification with transformers. IEEE Trans. Geosci. Remote Sens. (2021)
3. Jocher, G., Stoken, A., J. B. et al., ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration (2021). [Online]. Available: https://doi.org/10.5281/zenodo.4418161
4. Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset. In: CVPR Workshop on the Future of Datasets in Vision, vol. 2 (2015)
5. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: a retrospective. Int. J. Comput. Vis. **111**(1), 98–136 (2015)
6. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the kitti dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013)
7. Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: "nuscenes: A multimodal dataset for autonomous driving," In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 11 621–11 631, (2020)
8. Singh, G., Akrigg, S., Di Maio, M., Fontana, V., Alitappeh, R. J., Saha, S., Jeddisaravi, K., Yousefi, F., Culley, J., Nicholson, T., et al.: Road: The road event awareness dataset for autonomous driving. arXiv preprint arXiv:2102.11585, (2021)
9. Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al.: Argoverse: 3d tracking and forecasting with rich maps. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8748–8757 (2019)
10. Kesten, R., Usman, M., Houston, J., Pandya, T., Nadhamuni, K., Ferreira, A., Yuan, M., Low, B., Jain, A., Ondruska, P., et al.: Lyft level 5 av dataset 2019. https://level-5.global/data/ (2019)
11. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2446–2454 (2020)
12. Zendel, O., Murschitz, M., Zeilinger, M., Steininger, D., Abbasi, S., Beleznai, C.: Railsem19: A dataset for semantic rail scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 0 (2019)
13. Harb, J., Rébéna, N., Chosidow, R., Roblin, G., Potarusov, R., Hajri, H.: "Frsign: A large-scale traffic light dataset for autonomous trains," CoRR, vol. abs/2002.05665, (2020). [Online]. Available: arXiv:2002.05665
14. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16, (2017)
15. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A. M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3234–3243 (2016)
16. Richter, S. R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: European Conference on Computer Vision (ECCV), ser. LNCS, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, pp. 102–118, (2016)
17. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2147–2156 (2016)
18. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp. 1440–1448 (2015)
19. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 7074–7082 (2017)
20. Cai, Z., Fan, Q., Feris, R. S., Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection (2016)
21. Chabot, F., Chaouch, M., Rabarisoa, J., Teuliere, C., Chateau, T.: Deep manta: a coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2040–2049 (2017)
22. Xu, B., Chen, Z.: Multi-level fusion based 3d object detection from monocular images. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2345–2353 (2018)
23. Hu, H.-N., Cai, Q.-Z., Wang, D., Lin, J., Sun, M., Krahenbuhl, P., Darrell, T., Yu, F.: Joint monocular 3d vehicle detection and tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5390–5399 (2019)
24. Liu, Z., Wu, Z., Tóth, R.: Smoke: single-stage monocular 3d object detection via keypoint estimation (2020)
25. Qin, Z., Wang, J., Lu, Y.: Monogrnet: a general framework for monocular 3d object detection (2021)
26. Brazil, G., Liu, X.: M3d-rpn: Monocular 3d region proposal network for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9287–9296 (2019)

27. Liu, Y., Yixuan, Y., Liu, M.: Ground-aware monocular 3d object detection for autonomous driving. IEEE Robot Autom. Lett. **6**(2), 919–926 (2021)
28. Li, P., Zhao, H., Liu, P., Cao, F.: Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving (2020)
29. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
30. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation (2019)
31. UM and F. C. for Autonomous Vehicles (2021) umautobots/gta-visionexport. https://github.com/umautobots/GTAVisionExport, Online; Accessed July (2021)
32. Jotrius. (2015) Railroad engineer. https://www.gta5-mods.com/scripts/railroad-engineer, Online; Accessed July (2021)
33. Mauri, A., Khemmar, R., Decoux, B., Haddad, M., Boutteau, R.: Real-time 3d multi-object detection and localization based on deep learning for road and railway smart mobility. J Imaging **7**(8), 145 (2021)
34. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
35. Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Sun, Y., He, T., Mueller, J., Manmatha, R. Li, M., Smola, A. J.: Resnest: Split-attention networks. arXiv:2004.08955
36. Chen, X., Kundu, K.,Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals for accurate object class detection. In: Advances in Neural Information Processing Systems. Citeseer, pp. 424–432 (2015)
37. Bochkovskiy, A., Wang, C., Liao, H. M.: Yolov4: Optimal speed and accuracy of object detection. arXiv:2004.10934
38. Smith, L. N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, vol. 11006. International Society for Optics and Photonics, p. 1100612 (2019)
39. Simonelli, A., Bulo, S. R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1991–1999 (2019)
40. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv:1804.02767 (2018)

Antoine Mauri is currently a CIFRE (Convention Industrielle de Formation par la REcherche) PhD Student with the IRSEEM, Normandy University of Rouen, France and SEGULA Technologies. He obtained an Engineer degree in Robotics and Computer Vision, from Telecom Physique Strasbourg in 2019 along with a Master IRIV (Imagerie, Robotique, Ingenierie pour le Vivant). His research interest includes Artificial Intelligence, deep learning, and computer vision for Road and Rail Smart Mobility.

Redouane Khemmar is currently associate professor with the ESIGELEC, Normandy University of Rouen, France. He received his European Master of Science in Computer Vision from University of Poitiers (France) and his PhD in Computer Vision from the University of Strasbourg (France). For two years, he was with the University of Strasbourg as an associate teacher researcher, before moving to the industry, where he worked as engineer and project manager in Jouve, Alten, and Thales companies. After, he joined IRSEEM/ ESIGELEC High Engineer School (Normandy University) as an associate professor in embedded systems and computer vision, and the "Instrumentation, Computer Sciences and Systems" research team in the IRSEEM laboratory. His research interests include mobile robotics, environment perception, and computer vision dedicated to autonomous vehicles and smart mobility.

Benoit Decoux is currently with the ESIGELEC, Normandy University of Rouen, France. He obtained his PhD from the University of Rouen in 1995. From 2008 to 2012, he has worked as a research engineer in the Kelenn Technology Company, on industrial applications of optical character recognition. Since 2012, he works as a teacher and researcher, and his research interests include embedded systems, computer vision and artificial intelligence.

Madjid Haddad is a Research and Innovation Manager at SEGULA Technologies. After completing a Master's degree in Mechanical Engineering at the University of Toulouse III, he has carried out his PhD in Mechanical Engineering at the University of Toulouse III. Afterwards, he did a post doctorate in Mechanical Engineering at the University of Technology of CompiÃƒÂ¨gne. He then joined the SEGULA Technologies as Research and Innovation Manager.

Rémi Boutteau received his engineering degree from the IMT Lille Douai and his MSc degree in Computer Science from the University of Lille in 2006. In 2010, he received his PhD degree from the University of Rouen Normandy for works related to Computer Vision (catadioptric sensors, 3D reconstruction, Structure-from-Motion). From 2009 to 2020, he was an Associate Professor at the ESIGELEC engineering school and a researcher in the IRSEEM research institute. In 2018, he obtained the HDR (French Habilitation to supervise research) from the University of Rouen Normandy for his research on autonomous vehicles localization. Since 2020, he is a Full Professor at University of Rouen Normandy within the STI team (Intelligent Transportation System) at the LITIS Lab (IT Laboratory, Information Processing and Systems). His research interests are perception, localization and computer vision dedicated to autonomous vehicles.