Exploring Domain Adaptation with Depth-based 3D Object Detection in CARLA Simulator

Miguel Antunes, Luis M. Bergasa, Santiago Montiel-Marín, Fabio Sánchez-García, Pablo Pardo-Decimavilla, and Pedro Revenga

RobeSafe research group, Electronics Department, Universidad de Alcalá, Spain
{miguel.antunes, luism.bergasa, santiago.montiel, pedro.revenga}@uah.es
{fabio.sanchezg, pablo.pardod}@edu.uah.es

Abstract. Data collection and scene understanding have become crucial tasks in the development of intelligent vehicles, particularly in the context of autonomous driving. Deep Learning (DL) and transformer-based architectures have emerged as the preferred methods for object detection and segmentation tasks. However, DL-based methods often require extensive training with diverse data, posing challenges in terms of data availability and labeling. To address this problem, techniques such as transfer learning and data augmentation have been adopted. Simulators like CARLA have gained popularity in the autonomous driving domain, enabling the evaluation of architectures in realistic environments before real-world deployment. Synthetic data generated by simulators offers several advantages, including cost-effectiveness, access to diverse scenarios, and the ability to generate accurate ground truth annotations. In this paper, we focus our investigation on evaluating the performance and domain adaptation capabilities of a 3D object detection pipeline based on depth estimation using a stereo camera in the CARLA simulator. Our main objective is to analyze the results of the depth estimation stage using two different approaches: CoEx and SDN. The different experiments will be performed on real and synthetic scenarios from the KITTI and SHIFT datasets.

Keywords: Deep Learning, Transfer Learning, 3D object detection, KITTI, CARLA.

1 INTRODUCTION

Data collection and scene understanding have become one of the main tasks in intelligent vehicles developed in recent years. Through multiple sensors such as cameras, LiDAR, or radar, they obtain the necessary information to make decisions depending on the surrounding scene. Specifically, for autonomous driving, knowing and monitoring other vehicles, pedestrians, or traffic signals is essential to ensure safety and respect traffic rules. Almost all current methods are based on Deep Learning (DL) because it has shown impressive results in object detection, semantic segmentation, and instance segmentation tasks. Convolutional

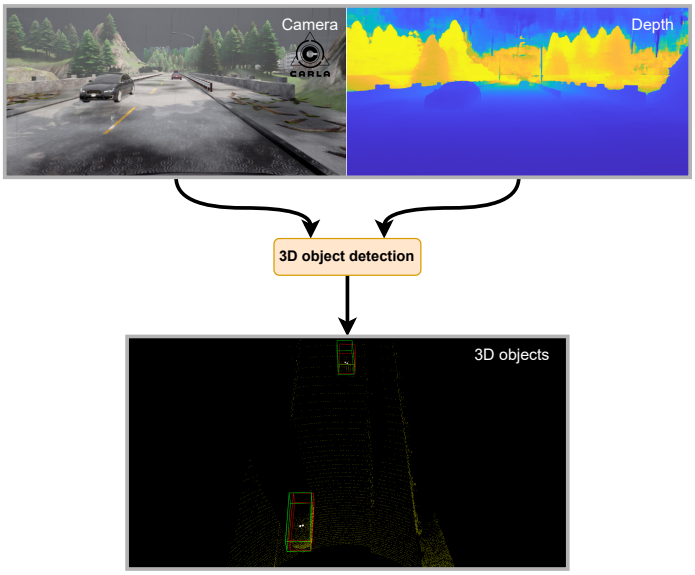

Fig. 1: Representations of three different stages in the pipeline. The first image is an example from CARLA simulator, the second is the corresponding estimated depth and the last is the projected poincloud and detected objects.

Neural Networks (CNN) have become the preferred DL method due to their ability to learn complex representations of images and detect objects with high accuracy. The main inconvenience of this methods is the necessity of an extensive training process that usually requires a large amount of diverse data. To address this problem, techniques such as transfer learning or data augmentation have emerged, which help to accelerate the different training processes. In the case of autonomous driving, the use of simulators such as CARLA [1] has increased in order to evaluate the robustness of architectures in realistic environments before being deployed in real life, allowing simulated data to be used as synthetic sources.

The use of this type of data offers several advantages over using real-world data alone. First, collecting and labeling real-world data can be time-consuming, expensive, and sometimes dangerous. Training an object detector for autonomous vehicles requires a large amount of labeled data covering a wide range of scenarios, such as different weather conditions, lighting, and traffic situations. Collecting such data through real-world testing would be time-consuming and expensive. Second, synthetic data and simulators can provide access to a larger and more diverse set of scenarios, which is difficult to achieve in real-world data collection. This allows for more efficient training of DL detectors, leading to faster and more accurate results. Third, simulators and synthetic data can be used to generate ground truth annotations directly, which are necessary for training object detectors. This can be done with a high degree of accuracy, unlike manual labeling of real-world data, which can be prone to human error.

In this paper, we introduce a modular architecture designed to generate depth information from a stereo image and perform 3D object detection. We aim to quantify the adaptability of the different stages within this multi-stage 3D detection architecture (Figure 1) to both real and simulated environments. In addition, we explore the potential of using synthetic data as a baseline in real-world scenarios. We emphasize the importance of evaluating how effectively the detection architecture adapts to synthetic data, with a specific focus on the depth estimation stage, which lacks a published comparative analysis in CARLA. To facilitate this assessment, we employ the KITTI and SHIFT datasets for the training and evaluations. The results, which include both CoEx and SDN in terms of depth estimation, are presented, along with an evaluation of the detection performance of the complete architecture. These findings aim to provide comprehensive insights into the system's capabilities and its potential applications in real-world scenarios.

2 RELATED WORK

In this section, we review previous research and approaches related to our proposed object detection pipeline and performance in CARLA. We examine various methods and techniques for depth estimation and stereo camera-based detection, highlighting their strengths and limitations.

2.1 2D object detection

The goal of 2D object detection is to accurately locate objects of interest within an image, typically by drawing bounding boxes around them, and classify them into different categories. It is one of the most explored tasks in perception systems. Detectors such as Single Shot MultiBox Detector (SSD) [2] or YOLO (v5 and v8) [3] are preferred over other two-stage detectors such as Faster-RCNN[4] for applications that require real-time processing and low computational resources. These detectors use a single-stage architecture that allows them to perform detection and classification in a single feedforward pass, as opposed to two-stage detectors, which first propose regions of interest and then perform classification within these regions. This makes lightweight detectors faster and more efficient, ideal traits for autonomous driving tasks. Studies in 2D detection such as [5] show that the use of synthetic data can help in the training and performance of these models in real scenes.

2.2 3D object detection

Several deep learning-based 3D monocular camera object detection methods have been proposed in recent years. DeepMANTA [6] attempts to associate the vehicle with a series of templates based on 2D object detections and keypoints. SMOKE [7] uses a hierarchical layer fusion network (DLA-34) as a backbone and estimates object keypoints and 3D bounding box parameters. ImVoxelNet [8]

accepts an arbitrary number of input images, extracts features from the images using CNN, and projects them into 3D (voxels) for feature fusion. MonoDTR [9] proposes to use Depth Aware Feature Enhancement (DFE) via auxiliary supervision to feed a transformer encoder-decoder architecture with positional information generated by Depth Positional Encoding (DPE). The use of stereo cameras allows to maintain depth information that can be used to generate more accurate detections in the 3D world. Most techniques employ an approach in which depth information is estimated from the two stereo images and utilized for performing detection in a 3D space. For instance, DSGN [10] or PLUMENet [11] create a 3D depth information volume and carry out detections within a single network. When using three-dimensional point clouds instead of images, there are specialized object detectors such as SECOND [12] or CenterPoint [13] that can detect objects with a higher spatial context. Typically, these techniques are associated with sensors such as LiDAR or radar. However, it is also possible to use point clouds generated by cameras.

For CARLA applications, works such as [14] or [15] use simulated LiDAR data to detect objects in the environment, proving that the performance achieved is lower than real data due to the difference in the quality of the point clouds. On the other hand, [16] proposes the use of synthetic data as an additional source for improving performance in real-world scenarios of a LiDAR 3D detector.

2.3 Camera depth estimation

Depth estimation systems can provide a significant amount of information about the environment around a system. For this reason, approaches have emerged that focus on estimating the depth of pixels in an image and then performing a 3D reconstruction of the environment. Similar to object detection techniques, depth estimation methods can use both monocular and stereo cameras. Stereo cameras offer an advantage over monocular cameras by providing more 3D information, allowing for more accurate depth estimation in long range scenarios. Methods, such as SDN [17], CoEx [18] or Google HITNet [19] focus on executing stereo matching for obtaining depth or disparity information from stereo images. For monocular camera there are techniques such as DepthFormer [20] that employ an encoder-decoder architecture using convolutions to extract local information and transformers for handling long-range dependencies or NeW CRF [21] that uses windows with fully-connected CRFs. In our experiments, we use CoEx due to its recognized speed and efficiency in the field. In addition, we use SDN to compare with the modular architecture proposed by the authors to ensure that the results are consistent. It's important to note that the authors did not provide depth metrics, while our approach includes this evaluation.

3 ARCHITECTURE

To evaluate the impact of the simulation environment on each stage of the pipeline, we conducted a multi-stage analysis of our proposed architecture (Fig. 2), which is explained in the different parts of this section.

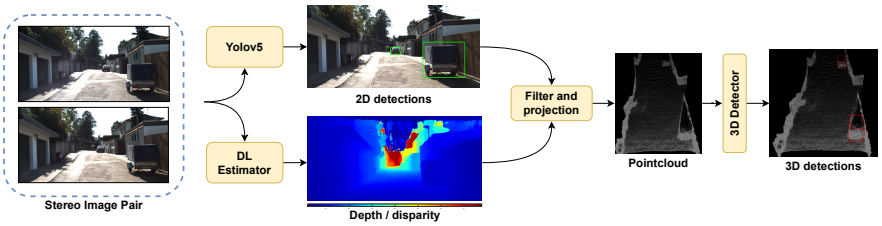

Fig. 2: Proposed 3D detection pipeline. The different images correspond to the output of the different stages of a KITTI scene.

3.1 Image 2D object detection

The first step in the pipeline involves performing the 2D object detection of all objects of interest within the scene. This is a critical step in many computer vision applications, as it serves as the starting point for the next stages in the pipeline. In our specific case, we have decided to use a detector from the YOLOv5 family [3], which has proven to be effective in detecting a wide variety of object classes in real-time scenarios. By taking advantage of the capabilities of this detector, we are able to efficiently and accurately identify objects of interest within the scene belonging to three different classes: car, pedestrian and cyclist. Depending on the required specifications, one of the five YOLOv5 models can be selected, which vary in terms of performance and inference time. For each object we define a 2D bounding box $B_{2D} \in \mathbb{R}^{1\times 4}$ with the maximum and minimum values in each image axis.

3.2 Depth estimation and point cloud projection

The two RGB images from the stereo camera $I_{stereo} \in \mathbb{R}^{2\times H\times W\times 3}$ are fed into a neural network to determine the disparity of the pixels $D \in \mathbb{R}^{H\times W}$. In our experiments, we choose to evaluate two state-of-the-art (SOTA) models for depth/disparity estimation: Stereo Depth Network (SDN) [17] and CoEx [18]. The SDN model directly optimizes the depth loss and has a higher computational cost compared to CoEx.

To perform the necessary transformations, knowledge of parameters from the camera's intrinsic matrix (equation 1) is required, such as the focal lengths in each axis of the image (f_x, f_y) or the optical center (c_x, c_y).

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

To work in an equivalent space to the real scenario, depth values $Z \in \mathbb{R}^{H\times W}$ should be obtained from the predicted disparity image D. The transformation from disparity to depth values is performed using Equation 2a, where the baseline distance B and the horizontal focal length f_x of the camera must be known.

To reduce the probability of getting detections in the wrong positions, the depth image is applied only over to bounding boxes detected by Yolo (Fig. 3a).

With this approach, the point cloud represents only the frustums containing the objects to be detected, eliminating undesired elements and a large amount of noise introduced by depth estimation at objects edges and the sky.

Once filtered, the depth map is projected using the Equations in 2b into the 3D environment to generate a point cloud $PCL \in \mathbb{R}^{n\times 3}$ similar to a LiDAR sensor point cloud (Fig. 3b). However, unlike LiDAR or radar, the camera-generated point cloud is much denser as it concentrates $H * W$ points only in the camera's field of view (FoV). The disadvantage compared to LiDAR or radar point clouds is that depth information is obtained through estimation rather than measurement with the corresponding sensor and there is no additional information such as intensity or doppler.

$$a)\ \ Z(u,v) = \frac{f_x \cdot B}{D(u,v)} \quad b)\ \ x = \frac{(u-c_x)\cdot z}{f_x},\ \ y = \frac{(v-c_y)\cdot z}{f_y},\ \ z = Z(u,v) \quad (2)$$

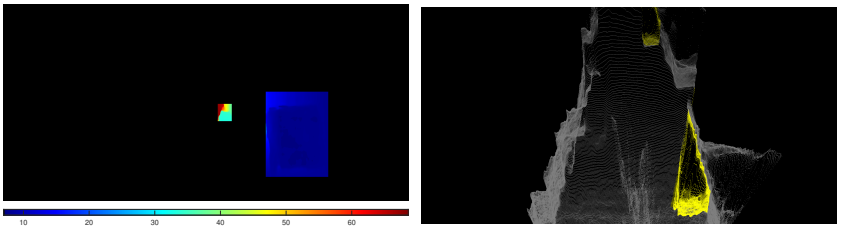

Fig. 3: a) Filtered depth image. b) Generated point cloud from predicted depth. Full data in gray and filtered data in yellow. Both from the same scene as Figure 2.

3.3 3D object detection

Once the point cloud has been generated and filtered from the camera information, we perform 3D object detection. Since data is a point clouds from other sensors, LiDAR detection techniques can be applied to obtain the present objects in the scene. As discussed in the experiments section, it is important to note that the intensity channel normally present in LiDAR is not present in the camera. Therefore, when training and inference are performed, it is important to either eliminate it or fix it to a known value to avoid introducing noise into the detector. Specifically, we decided to use PointPillars [22], working with the input data as if it were a LiDAR point cloud, even though their distribution is different.

4 EXPERIMENTS

In this section we present the results of our experiments, conducted using real world data from autonomous driving datasets such as KITTI and data from CARLA simulator using SHIFT dataset [23]. We evaluate the performance of our pipeline under various conditions and different types of objects and obstacles.

4.1 Datasets

KITTI dataset [24] is a popular benchmark in computer vision and robotics for tasks such as object detection, tracking and depth estimation. The dataset consists of 7481 high-resolution (1242x375) stereo images with a baseline of 0.54 meters, 3D Velodyne HDL-64E point clouds and sensor data captured from a moving platform mounted on a vehicle. The dataset includes a variety of real-world scenarios, such as urban and rural driving, and is often used to train and evaluate algorithms for autonomous driving and KITTI has played a significant role in advancing research in computer vision and robotics, and continues to be an important resource for the development of new algorithms and techniques. **SHIFT** dataset [23] provides data from the CARLA simulator for our experiments. The amount of labelled information and the variety of scenarios is much greater than open-source available real world datasets, since it collects data generated entirely in CARLA. Specifically, this dataset collects data from a 128 channel LiDAR and a multi-camera system that includes a stereo pair with a 0.2 meter baseline and a 1280x800 resolution. It provides labeled data for both 2D and 3D objects, as well as depth information for all pixels in the images. In the experiments in this paper, a subset of sequences with a similar number of frames (7500) and split to KITTI is selected. It should be noted that the intensity provided by the simulated LiDAR point clouds differs from a real sensor. Therefore, when performing domain adaptation, we only use the coordinate information to train the 3D object detector.

4.2 Metrics

Two different sets of metrics for depth estimation and object detection are used in the experiments. Firstly, standard metrics were used to measure the performance of the depth estimation networks following the KITTI benchmark. All depth values are in meters except in iRMSE:

– Root Mean Squared Error (**RMSE**): squared difference between the predicted depth value and the ground-truth value.

$$\mathbf{RMSE} = \sqrt{\frac{\sum_{i,j}(\widehat{Z}^{i,j} - Z^{i,j})^2}{N}} \tag{3}$$

– Inverse Root Mean Squared Error (**iRMSE**): squared difference between the inverted predicted depth value and the ground-truth value, both in Km.

$$\textbf{iRMSE} = \sqrt{\frac{\sum_{i,j}(\frac{1}{\widehat{Z}^{i,j}} - \frac{1}{Z^{i,j}})^2}{N}} \tag{4}$$

– Relative Absolute Error (**absErrorRel**): percentage of error relative to the ground-truth value.

$$\textbf{absErrorRel} = \frac{1}{N}\sum_{i,j}\frac{|\widehat{Z}^{i,j} - Z^{i,j}|}{Z^{i,j}} \tag{5}$$

– Logarithmic Root Mean Squared Error (**RMSElog**): squared difference between the natural log of the predicted depth value and the natural log of the ground-truth value.

$$\textbf{RMSElog} = \sqrt{\frac{\sum_{i,j}(log(\widehat{Z}^{i,j}) - log(Z^{i,j}))^2}{N}} \tag{6}$$

– Threshold n (δ_n):

$$max(\frac{\widehat{Z}^{i,j}}{Z^{i,j}}, \frac{Z^{i,j}}{\widehat{Z}^{i,j}}) < 1.25^n \tag{7}$$

For object detections we decided to use AP or Average Precision, one of the most common metrics in detection tasks, to measure how well the model is able to detect and locate objects in an image or 3D environment. The corresponding AP11 (Average Precision) is calculated for each class and represents the area under the Precision-Recall curve, which shows the relationship between the number of true positives, false positives, and false negatives generated by the model. The Precision-Recall curve is generated by varying the detection threshold of the model.

We also use average orientation similarity (AOS) to measure each object orientation. Following the KITTI 3D object detection benchmark, we calculate three different mAP:

– **2D detections on each image**: a true positive is considered if the overlap between the ground-truth and the detected 2D object is more than a 70% for cars class objects or 50% if they are pedestrians.
– **Bird Eye View (BEV)**: the overlap of the projected BEV between the ground-truth and the predicted object must be greater than 50% for cars and 25% for pedestrians.
– **3D detections**: a true positive is considered if the tridimensional intersection of the 3D bounding boxes of the ground-truth and detected objects is greater than 50% for cars and 25% for pedestrians.

4.3 Procedure and results

Due to the modular nature of our architecture, we measure the performance on two main tasks in real and simulated scenarios: depth estimation and object

detection. All experiments are performed on a desktop computer with a GPU NVIDIA GTX 1080 Ti.

Depth evaluation: we train and evaluate two state-of-the-art stereo depth estimation methods: Stereo Depth Network (SDN) [17] and CoEx [18]. In both cases, the initial weights for training are obtained from SceneFlow, followed by transfer learning on KITTI or SHIFT datasets. This approach allows us to evaluate the domain adaptation capabilities from a similar starting point. All training processes consist of 50 epochs with an initial learning rate of 0.001. It is worth pointing out that in the case of SDN, the obtained metrics quantify the performance of depth estimation complementing the study conducted by the authors of the paper, who only provide detection metrics.

Table 1: Depth error metrics on different training and validation subsets with the two methods.

Method	Train Set	Eval. Set	RMSE ↓	iRMSE ↓	RMSElog ↓	Rel. abs. error (%) ↓
SDN	KITTI	KITTI	3.3366	13.6359	0.1498	9.6186
	SHIFT	SHIFT	4.6104	16.3160	0.1855	8.2246
	SHIFT	KITTI	3.9165	21.2901	0.1914	11.9913
COEX	KITTI	KITTI	3.5426	15.6604	0.1607	10.3993
	SHIFT	SHIFT	4.3414	17.4171	0.1769	7.2447
	SHIFT	KITTI	6.2974	18.8357	0.2073	15.0839

Table 1 and Table 2 show the different metrics obtained by training and evaluating on both datasets. In both cases, the transition from a real to a virtual environment when training and evaluating in the same dataset has a noticeable effect, resulting in a degradation of performance in SHIFT for most metrics, with the exception of Relative Absolute Error, which can be attributed to the fact that the ground truth in SHIFT is a complete image with depth values, unlike KITTI, which consists of the LiDAR projection, leaving some pixels unlabelled. This increase in error may be magnified due to a smaller baseline in the stereo

Table 2: Delta thresholds values on different training and validation subsets with the two methods.

Method	Train Set	Eval. Set	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
SDN	KITTI	KITTI	0.9581	0.9753	0.9856
	SHIFT	SHIFT	0.9396	0.9667	0.9785
	SHIFT	KITTI	0.9294	0.9624	0.9766
COEX	KITTI	KITTI	0.9567	0.9741	0.9843
	SHIFT	SHIFT	0.9383	0.9690	0.9805
	SHIFT	KITTI	0.9275	0.9612	0.9743

camera setup, which leads to greater errors at long distances from the vehicle.

When training on SHIFT and evaluating on KITTI, both networks show only a slight degradation in performance, confirming that training on CARLA can achieve acceptable performance in real-world scenarios.

Object detection: the final stage of the system is responsible for performing object detection using the projected and filtered point cloud generated from the depth estimation at each pixel of the image. It is important to point out that SHIFT does not provide difficulty categories in its labeling, as KITTI does. To ensure a fair comparison, the results on KITTI are averaged across all three difficulties (Easy, Medium, and Hard). We conduct multiple training runs of PointPillars, consisting of 15 epochs and an initial learning rate of 0.001. In the first stage, we train the detector using LiDAR point clouds from both datasets to establish a baseline for comparison. Subsequently, we train PointPillars using the depth estimations from SDN and CoEx on both datasets, enabling us to quantify the performance difference between real and simulated environments. It is important to note that due to the multi-stage nature of the system, the error of the different stages is propagated to the next component of the architecture. All results are summarised in Table 3. Training directly on LiDAR gives the best

Table 3: Training results on both datasets (AP11 with Filtered PseudoLidar) of PointPillars only on LiDAR and our modular architecture.

		Car				Pedestrian			
		2D	BEV	3D	AOS	2D	BEV	3D	AOS
KITTI	LiDAR	**89.674**	**91.658**	**91.589**	**89.240**	**61.942**	**68.725**	**68.456**	**44.463**
	SDN	85.917	85.369	83.865	83.553	39.563	37.098	36.675	22.950
	CoEx	77.733	76.912	73.975	75.930	32.053	33.455	33.255	17.493
SHIFT	LiDAR	**62.346**	**96.733**	**96.679**	30.420	**57.075**	**60.209**	**60.210**	**22.050**
	SDN	61.294	89.567	85.545	28.540	15.711	11.091	11.091	7.980
	CoEx	60.889	85.707	80.829	**31.530**	8.128	9.071	9.070	3.389

results in both datasets and classes for all objects in the camera FoV. In the case of KITTI, the use of the heavier depth estimation network achieves results that are quite close to LiDAR for the car class, but there is a significant drop in performance for pedestrians. Similar effects are observed in the SHIFT dataset, specifically for cars, where LiDAR achieves the best results and both SDN and CoEx fall slightly behind. However, the drop in performance for pedestrians is even more pronounced, with AP values as low as 10, which is significantly lower than the drop observed in KITTI.

Figure 4 shows three qualitative examples in KITTI and SHIFT. As observed in the generated depth metrics, SDN produces more accurate depth images, particularly for smaller objects like traffic lights. On the other hand, CoEx struggles with such objects, resulting in larger errors and pointclouds with more noise. Despite this limitation, our system demonstrates reliable detections at short distances, typically within the range of 10-20 meters, with both depth estimation architectures.

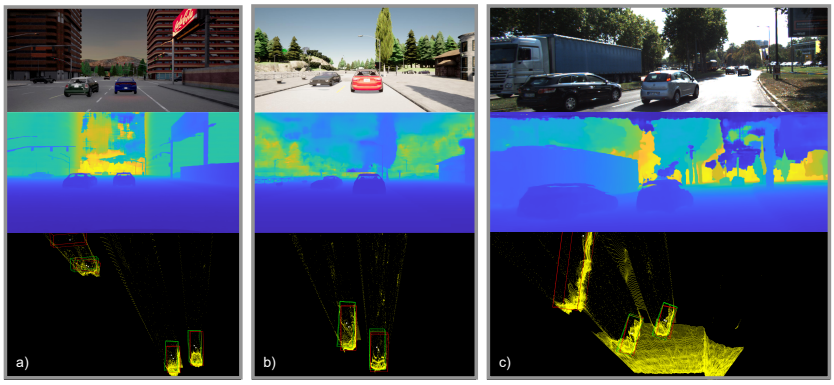

Fig. 4: Cualitative results with PointPillars as the 3D object detector. a) SHIFT using SDN. b) SHIFT using CoEx. c) KITTI using SDN.

5 CONCLUSIONS

A modular 3D detection system was designed and implemented to perform comprehensive evaluations. The depth estimation evaluation aimed to quantify how well the system could estimate the distances of objects in a real and simulated environment. On the other hand, the object detection evaluation focused on analyzing the system's ability to accurately detect and localize objects using data from the different depth estimation systems. By conducting these evaluations, a thorough review of the system's performance in both the depth estimation and object detection domains was achieved, providing valuable insight into the system's ability to operate in different environments.

From the experiments conducted, it can be concluded that depth estimation is more accurate in real data compared to a rendered simulator and that the synthetic data can be used as a baseline in real environments in which labeled data is more limited. The use of pseudo-LiDAR techniques without rectification gives results close to the use of LiDAR for large objects in the image, such as cars. However, there is a noticeable drop in performance for less visible objects such as pedestrians, and this drop is amplified in synthetic images.

Acknowledgements. This work has been supported from the Spanish PID2021-126623OB-I00 project, funded by MICIN/AEI and FEDER, and TED2021-130131A-I00, PDC2022-133470-I00 projects, funded by MICIN/AEI and the European Union NextGenerationEU/PRTR.

References

1. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in Proceedings of the 1st Annual Conference on Robot Learning, 2017, pp. 1–16.

2. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," pp. 21–37, 2016.
3. G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics
4. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.
5. M. Antunes, L. M. Bergasa, et al., "Including transfer learning and synthetic data in a training process of a 2d object detector for autonomous driving," in ROBOT2022: Fifth Iberian Robotics Conference. Springer International Publish- ing, 2023, pp. 465–478.
6. F. Chabot et al., "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," 2017.
7. Z. Liu, Z. Wu, and R. Tóth, "Smoke: Single-stage monocular 3d object detection via keypoint estimation," 2020.
8. D. Rukhovich, A. Vorontsova, and A. Konushin, "Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection," 2021.
9. K.-C. Huang, T.-H. Wu, H.-T. Su, and W. H. Hsu, "Monodtr: Monocular 3d object detection with depth-aware transformer," 2022.
10. Y. Chen, S. Liu, X. Shen, and J. Jia, "Dsgn: Deep stereo geometry network for 3d object detection," 2020.
11. Y. Wang, B. Yang, R. Hu, M. Liang, and R. Urtasun, "Plumenet: Efficient 3d object detection from stereo images," 2021.
12. Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," Sensors, 2018.
13. T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," CVPR, 2021.
14. J. Del Egido et al., "360 real-time 3d multi-object detection and tracking for autonomous vehicle navigation," in Advances in Physical Agents II. Springer International Publishing, 2021, pp. 241–255.
15. W. Zimmer, M. Grabler, and A. Knoll, "Real-time and robust 3d object detection within road-side lidars using domain adaptation," 2023.
16. D. Dworak and others., "Performance of lidar object detection deep learning architectures based on artificially generated point cloud data from carla simulator," in MMAR 2019, 2019, pp. 600–605.
17. Y. You, Y. Wang, et al., "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," 2020.
18. A. Bangunharcana, J. W. Cho, et al., "Correlate-and-excite: Real-time stereo matching via guided cost volume excitation," 2021.
19. V. Tankovich, C. Häne, Y. Zhang, A. Kowdle, S. Fanello, and S. Bouaziz, "Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching," 2023.
20. Z. Li, Z. Chen, X. Liu, and J. Jiang, "Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation," 2022.
21. W. Yuan, X. Gu, Z. Dai, S. Zhu, and P. Tan, "New crfs: Neural window fully-connected crfs for monocular depth estimation," 2022.
22. A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," 2019.
23. T. Sun, M. Segu, et al., "SHIFT: a synthetic driving dataset for continuous multi-task domain adaptation," in CVPR, June 2022, pp. 21 371–21 382.
24. 24. A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in CVPR, 2012.