Instance Segmentation in CARLA: Methodology and Analysis for Pedestrian-oriented Synthetic Data Generation in Crowded Scenes

Maria Lyssenko*†, Christoph Gladisch*, Christian Heinzemann*, Matthias Woehrle*, Rudolph Triebel†‡
*Robert Bosch GmbH, Corporate Research, Robert Bosch Campus 1, 71272 Renningen
firstname.lastname@de.bosch.com
†Technical University of Munich, Boltzmannstrasse 3, 85748 Garching
firstname.lastname@in.tum.de
‡German Aerospace Center (DLR), Münchener Strasse 20, 82234 Wessling
rudolph.triebel@dlr.de

Abstract

The evaluation of camera-based perception functions in automated driving (AD) is a significant challenge and requires large-scale high-quality datasets. Recently proposed metrics for safety evaluation additionally require detailed per-instance annotations of dynamic properties such as distance and velocities that may not be available in openly accessible AD datasets. Synthetic data from 3D simulators like CARLA may provide a solution to this problem as labeled data can be produced in a structured manner. However, CARLA currently lacks instance segmentation ground truth. In this paper, we present a back projection pipeline that allows us to obtain accurate instance segmentation maps for CARLA, which is necessary for precise per-instance ground truth information. Our evaluation results show that per-pedestrian depth aggregation obtained from our instance segmentation is more precise than previously available approximations based on bounding boxes especially in the context of crowded scenes in urban automated driving.

1. Introduction

In automated driving (AD), camera-based perception functions still pose a major challenge for reliable decision making especially in unstructured and crowded urban scenes. While there is a high amount of available image data, it is often not suitable for safety evaluation where high-quality data is of utmost significance and where specific ground truth annotations may be required. In contrast to standard performance measures, the evaluation in safety-critical domains is often performed on a per-instance basis,

Figure 1: Result of the implemented instance segmentation see Section 3. The Figure depicts separated pedestrian instances within the semantic segmentation from Subfigure 2b, each pedestrian is assigned a specific ID, represented as different coloring in the image.

e.g. evaluation is carried out for each individual pedestrian in an image separately [3, 24, 17]. This requires that all information for evaluation of safety-related metrics like distance and maybe further dynamic properties such as orientation and velocity are annotated for each single pedestrian.

Recent examples for safety evaluations based on distance are given by Bolte et al. [3] and Lyssenko et al. [17], who evaluate detection performance for individual pedestrians in relation to their distance, as well as by Volk et al. [24], who incorporate trajectories and velocities of pedestrians and cars into the computation of a safety metric.

Synthetic data from 3D simulators like CARLA [11] may provide a solution to the dataset engineering problem for such evaluations as labeled data and dynamic properties of subjects can be derived automatically. In addition, simulators provide tight control over the generated data which is beneficial for structured dataset engineering. Thus, synthetic data from a simulator (possibly combined with do-

main transfer techniques *e.g*. Ros et al. [20]) is a viable option for generating such high-quality data sets in a structured way along with the necessary instance-wise ground truth information. However, CARLA, a 3D simulation environment with particular focus on driving functions, does currently not provide an instance segmentation ground truth sensor[1] that allows to distinguish different pedestrians in the segmentation maps, which is a significant impediment for using such data for metric evaluation.

The main contribution of this work is a back projection pipeline that allows us to obtain accurate instance segmentation maps for CARLA. Our approach is based on a chain of coordinate transformations that back project pixels from the RGB images into the CARLA world and actors like pedestrians therein. A resulting instance segmentation map [2] is shown in Figure 1. The figure also shows a main motivation of our work: Accurate distance estimation of pedestrians in crowded scenes. Here, the previous state-of-the-art [17] used bounding boxes for determining the depth of a pedestrian. As we see on the right of the figure (bright green), this is not an issue for individual pedestrians since the bounding box will be well separated from others. However, let us consider the pedestrian on the left (pale green). In its bounding box, there are four additional pedestrians present in the background with widely different depths. Situations like these may frequently happen in urban scenarios, especially when they are crowded. In this case, using a bounding box approximation, as described for example in [17], will lead to inconsistent results. With an instance segmentation-based depth estimation, we can remedy such issues and can therefore provide more accurate results for depth-based evaluation metrics when using CARLA.

In our evaluation, we systematically construct a dataset using CARLA that is labelled with instance segmentation ground truth and per-pedestrian distance information. For comparison, we reimplemented the bounding box-based distance estimation by Lyssenko et al. [17], that is based on semantic segmentation, for evaluating the increase in precision. Our results show that even in the best case of accurately fitted bounding boxes, instance segmentation provides a significantly improved precision on our dataset. Using an optimal bounding box fit we could measure deviations in distance of more than $20m$ for several pedestrians between the approaches. We consider this as significant for safety-related evaluations, particularly in urban environments.

[1] `https://github.com/carla-simulator/carla/issues/76`

[2] For visualization purposes we show a crop of the full resolution 2048×1024 CARLA image

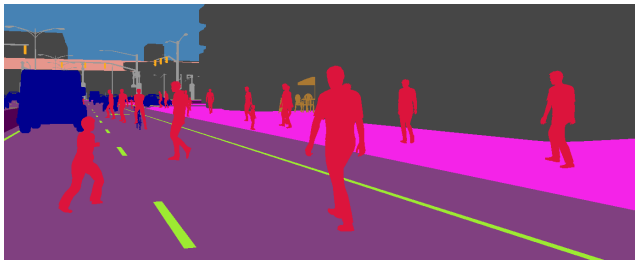
2. Related Work

Datasets: Until recently, it was challenging to find datasets containing instance-wise annotations for AD. Datasets like Pascal-Voc [12] and MS COCO [16] offer instance segmentation annotations but the diversity of objects is rather focused on indoor scenes (like *e.g*. in robotics). These may be suitable as additional—but not main—datasets for AD. KITTI [13, 5], is referred to as the pioneering multimodal dataset in AD providing 200k 2D and 3D bounding boxes for over 22 scenes, as well as additional depth information acquired from front-facing stereo images. The BDD100K dataset [25] is a large AD dataset with 100K video and 10 different tasks including rich set of corresponding annotations: object bounding box drivable area, full-frame semantic and instance segmentation. Additionally, the dataset covers various weather conditions, time, and scene types. The nuScenes dataset [5] comprises 1000 scenes with fully annotated 3D bounding boxes covering 23 semantic classes. Although it provides a variety of scenarios including dense traffic and challenging driving situations, neither instance nor semantic segmentation is included. The SYNTHIA dataset [20] is a collection of 213,400 synthetic images with semantic segmentation for 11 classes in urban scenes using a variation of seasons. The simulator used for generating the dataset is based on the Unity 3D engine.

However, with exception of *e.g*. A2D2 [14], available benchmark AD datasets display a lack of instance segmentation ground truth, as in *e.g*. nuScenes, KITTI, and even the synthetic SYNTHIA dataset, or do not offer any important physical cues such as distance in *e.g*. BDD100K or Cityscapes [10]. Due to the low costs of acquiring instance-level, pixel-wise annotations, (open-source) simulators with a flexible API may support in a controlled scenario generation with the requested ground truth.

Open-source Simulators: CARLA [11] is a popular open-source 3D simulator for the development of autonomous vehicles. Like Airsim [23], it is based on the Unreal Engine 4 (UE4) and implements various sensors including RGB camera, depth, semantic segmentation, and more. Unfortunately, AirSim focuses on various types of autonomous vehicles but does not refer to other subjects like pedestrians in the paper. Conversely, CARLA includes a controlled spawning function of *e.g*. pedestrians and contains eight 3D worlds (Town01-Town07, and Town10HD) with objects labelled using 23 classes. It provides a Python API and a client-server architecture making it flexible and easy to use out-of-the-box. SUMMIT [6] is a framework that builds on top of CARLA. Hence, it inherits the features from CARLA such as physics and rendering capabilities but extends it for simulating complex traffic behaviors in crowded urban scenarios. Similarly to SUMMIT, CADET [4] introduces a CARLA-based tool for generating

(a) RGB traffic scene scenario in the CARLA simulator.



(b) Semantic segmentation ground truth with 23 object classes.

Figure 2: Figure a shows the rendered CARLA output on the basis of the Unreal Engine, while Figure b and Figure 1 illustrate the corresponding semantic map and its refined instance segmentation, respectively.

training data from the simulator. It enables the export of high-quality synchronized LiDAR and camera data. They generated a dataset offering 10,000 samples with 2D and 3D annotations for the pedestrian and vehicle class, complemented by a depth map-based occlusion detection. Neither of these simulators provide instance segmentation.

Commercial Simulators Commercial 3D simulation is provided, *e.g.*, by Cognata [21]. They also provide datasets with pedestrians for vulnerable road user (VRU) detection applications [9] including scenarios such as roadworks and emergency situations. Additional commercial options for simulation-based synthetic image generation include LGVSL [19], [1] and dSPACE [22] simulators, where, *e.g.* dSPACE allows scenario generation from raw image data equipping the scenery with a variety of 3D assets.

Further work: Furthermore, there are publicly funded projects such as KI Absicherung – Safe AI for automated driving [2], which works on the cross-concerns of DNN-based computer vision algorithms, synthetic image generation for verification and validation. A particular focus is on pedestrian detection in urban scenarios. To evaluate such scenarios 3D synthetic data is developed including semantic and instance segmentation, respectively.

Summary: The goal of this work is to generate data with distance and instance segmentation information based on simulation. We want to generate controlled variations of urban scenes. To this end, CARLA meets the prerequisites by providing a flexible API enabling a controlled subject placement, and by offering instance segmentation ground truth or further functionalities that may support its implementation. With available depth maps, pixel-wise annotations, and precise knowledge of CARLA's 3D world, the present work proposes a novel function in CARLA, tackling the task of instance segmentation for the pedestrian class.

3. Pedestrian Instance Mask Estimation in CARLA

This section describes the combination of CARLA's actor information along with a chain of coordinate transformations to implement an accurate pedestrian instance distinction in crowded scenes. However, for our approach we assume that in CARLA collision of subjects (*e.g.* hugging or standing close together) is avoided [3], *i.e.* 3D actor's bounding boxes do not overlap. We leave potential collision cases as future work. Please not that the current setting still includes crowded scenarios (as it can be seen in Figure 1) which lead to a high degree of overlap in 2D bounding boxes.

We leverage PyTorch 1.7 [18] with CUDA support in order to boost performance on our rendered full-resolution images using CARLA's 0.9.11 Release.

3.1. Instance Segmentation

Unlike semantic segmentation, which only provides per-pixel class labels (a.k.a. semantic segmentation) regardless of object instances, instance segmentation aims to obtain the per-pixel class annotations, as well as instance-aware labels simultaneously.

Let us consider an RGB input image $I \in \mathbb{R}^{H \times W \times 3}$ where H and W denote the image's high and width in pixels, respectively. We define the instance ground truth as a set $\{(G_i, c_j)\}$ with $G_i \in \{0, 1, 2, ...m\}^{H \times W}$ denoting the integer-based instance segmentation mask for the i-th instance (subject), and m as the maximal instance number per image. $c_j \in \{1, 2, 3, ...M\}$ defines the j-th class category with M describing the total number of classes.

Although, for our rendered dataset M is equal to 23 considering all semantic classes in CARLA, our implemented back projection pipeline in Section 3.2 refers so far only to the pedestrian class. However, our instance segmentation approach can be trivially extended to any actor class in the CARLA simulator, *e.g.* vehicles, as all actors offer unique 3D bounding boxes, which is the prerequisite for our method.

Please note that due to CARLA's labelling policy, pedestrian and biker class members carry the same label ID.

[3] UE4's collision avoidance referenced in `https://github.com/carla-simulator/carla/issues/258`

However, our implemented instance approach filters non-pedestrian pixels within the instance mask providing pedestrian pixels only, *i.e.* in Figure 1 the biker is filtered out. Please note, in our dense setup in Figure 1 we may notice very small clusters of pixels belonging to heavily occluded pedestrians. However, these are not artifacts of our approach but actual pedestrian pixels.

Considering the constraints, our implementation reduces M to 1 (only pedestrian class is relevant), m to 21 (in Figure 5a maximal pedestrian count in the test split) and the ground truth set to $\{(G_i, c_1)\}$, respectively.

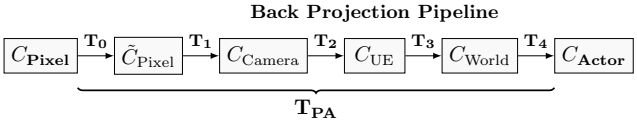


Figure 3: Overview of the applied coordinate system C transformations ($\mathbf{T_0}$-$\mathbf{T_4}$) to back project pedestrian pixels $C_{\mathbf{Pixel}}$ into the CARLA world C_{World}, and subsequently to actor coordinates $C_{\mathbf{Actor}}$ (using Equation 2-8 from Section 3.2).

3.2. Identifying Pedestrians in the Group

We leverage the pinhole camera model that mimics the geometrical mapping of a 3D world point $\mathbf{N} = [X, Y, Z, 1]^T \in \mathbb{R}^4$ to its corresponding 2D projection $\mathbf{n} = [u, v, 1]^T \in \mathbb{R}^3$ onto the pixel grid using Equation 1 (see [26]). We use homogenous coordinates as an augmented representation of points in $\mathbb{R}^n$ to $\mathbb{R}^{n+1}$, hence we use $n + 1$ dimensions. This representation is frequently used with perspective and projective transformations in *e.g.* robotics and computer vision [26].

$$\lambda \cdot \mathbf{n} = \mathbf{K} \cdot [\,\mathbf{R} \mid \mathbf{t}\,] \cdot \mathbf{N}, \text{ with } \mathbf{K} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$\lambda \in \mathbb{R}^+$ denotes an arbitrary scale factor, while the extrinsic parameters ($\mathbf{R}$, $\mathbf{t}$) describe the relationship between the world- and camera coordinate systems. Thereby $\mathbf{R} \in \mathbb{R}^{4\times4}$ and $\mathbf{t} \in \mathbb{R}^4$ define the position of the camera center and the camera's orientation in world coordinates, respectively. The intrinsic properties are summarized by matrix $\mathbf{K}$ with the principal point (u_0, v_0) $\in \mathbb{R}^2$ and the focal lengths α and $\beta \in \mathbb{R}^+$, scaling the image along u and v axes. All intrinsic parameters are provided by CARLA's sensor's class.

We split the back projection in the following steps as shown in Figure 3. We require the inverse transformation of the forward projection, that is given by Equation 2

$$\mathbf{T_{PA}} = \prod_{i=4}^{0} \mathbf{T_i}. \tag{2}$$

$\mathbf{T_{PA}}$ describes the chain of performed transformations in order to achieve the mapping from pedestrian pixels $[u, v, 1]^T_{C_{\mathbf{Pixel}}}$ to 3D points $[X, Y, Z, 1]^T_{C_{\mathbf{Actor}}}$.

Figure 4: Output of CARLA's depth sensor (8-Bit) that stores pixel-wise depth in meters (z_{buffer}).

Starting with a pixel-wise multiplication of the z-buffer (z_{buffer}) acquired from CARLA's depth sensor (see Figure 4), we apply $\mathbf{T_0}$ with

$$\mathbf{T_0} : z_{\text{buffer}}(u, v) \cdot [u, v, 1]^T_{C_{\mathbf{Pixel}}} \tag{3}$$

to obtain depth-weighted pixels when transforming $C_{\mathbf{Pixel}}$ to $\tilde{C}_{\text{Pixel}}$. This prerequisite is necessary to express $\mathbf{T_{PA}}$ as a cascaded transformation matrix enabling tensor initialization on the GPU using CUDA.

Equation 4 applies the inverse intrinsic matrix $\mathbf{K}^{-1}$ to transform the depth-aware pedestrian pixel coordinates in $\tilde{C}_{\text{Pixel}}$ to 3D camera coordinates C_{Camera} in $\mathbf{T_1}$ using

$$\mathbf{T_1} : \tilde{C}_{\text{Pixel}} \overset{\mathbf{K}^{-1}}{\rightarrow} C_{\text{Camera}}. \tag{4}$$

Subsequently, to map camera coordinates to the coordinate system C_{UE} used by Unreal Engine, we need to apply $\mathbf{T_2}$ with

$$\mathbf{T_2} : [x, y, z]_{\text{Camera}} \rightarrow [z, x, -y]_{\text{UE}}. \tag{5}$$

The *unreal-to-world* transformation $\mathbf{T_3}$ from Equation 6 denotes the coordinates mapping from C_{UE} to CARLA's world reference defined through C_{World}.

$$\mathbf{T_3} : C_{\text{UE}} \overset{[\,\mathrm{R_3} \mid \mathrm{t_3}\,]}{\rightarrow} C_{\text{World}} \tag{6}$$

To compute the 3D Point $[X, Y, Z, 1]^T_{C_{\text{UE}}}$ in world reference, we obtain $[\,\mathrm{R_3} \mid \mathrm{t_3}\,]$ from CARLA's API, accessing the matrix from a transformation class that defines a combination of location and rotation of CARLA's actors.

Finally, $\mathbf{T_4}$ from Equation 7 completes the back projection pipeline leveraging the actor's location and orientation.

$$\mathbf{T_4} : C_{\text{World}} \overset{[\,\mathrm{R_4} \mid \mathrm{t_4}\,]^{-1}}{\rightarrow} C_{\mathbf{Actor}} \tag{7}$$

Since CARLA's transformation class provides the *actor-to-world* conversion $[\,\mathrm{R_4} \mid \mathrm{t_4}\,]$ (analogous to $[\,\mathrm{R_3} \mid \mathrm{t_3}\,]$), the final transformation requires the inverse matrix $[\,\mathrm{R_4} \mid \mathrm{t_4}\,]^{-1}$

for the *world-to-actor* mapping. Resulting points $[X, Y, Z, 1]^T_{C_{\mathbf{Actor}}}$ are axis-aligned with the actor's 3D bounding box enabling a check of its extent along the three dimensions.

The last step includes a correspondence verification, whether the back projected 3D point $[X, Y, Z, 1]^T_{C_{\mathbf{Actor}}}$ matches a provided pedestrian actor bounding box from CARLA. After a successful search, the corresponding pedestrian pixel is assigned a unique object identification in the integer-based instance map $\{(G_i, c_1)\}$ (see Section 3.1).

We express the cascaded transformation from $C_{\mathbf{Pixel}}$ to $C_{\mathbf{Actor}}$ as a single matrix $\mathbf{T_{PA}}$ from Equation 2 to transform pedestrian pixels $[u, v, 1]^T_{C_{\mathbf{Pixel}}}$ to 3D points $[X, Y, Z, 1]^T_{C_{\mathbf{Actor}}}$ in CARLA's actor coordinate system using

$$\mathbf{N}_{C_{\mathbf{Actor}}} = \mathbf{T_{PA}} \cdot \mathbf{n}_{C_{\mathbf{Pixel}}}. \tag{8}$$

For an efficient implementation, we leverage CUDA and we convert the transformation matrix $\mathbf{T_{PA}}$ as well as our pixel array $\mathbf{n}_{C_{\mathbf{Pixel}}}$, to tensors on the GPU. As a result we can accelerate the instance mask calculation by a factor of $\approx 10^2$ (in seconds).

4. Data Collection of Urban Scenes in CARLA

CARLA uses Unreal Engine to render a virtual environment containing a virtual city, equipped with important elements for urban automated driving simulation *e.g.* ground markings, traffic signs, vegetation, and construction. The city can be populated with a variety of traffic participants like vehicles, bikes, and pedestrians with different appearances all derived from CARLA's blueprint library, similar to [20].

4.1. Crowded Scene Population

We implemented a Python API on top of CARLA's spawning functionality to control pedestrian placement. Leveraging different ground types (road, shoulder, and sidewalk) we aim for collision-aware pedestrian placement within the sensor's field of view, ideally achieving a wide coverage. In contrast to a previously described methodology in [17], we strive for a roughly equal pedestrian placement across all ground types, rather than contributing occasional waypoints from the road.

In contrast to SUMMIT's CARLA extension [6] we do not rely on topological graph constructs to define behaviors of pedestrian actors, but rather a desired spawning distance from our current sensor's location using CARLA's waypoint reference API[4]. During distance-aware placement, we request a specific pedestrian count at each simulation step. However, placing a high pedestrian count may not always be possible due to unstructured surroundings and its obstacles.

We select different elements from the blueprint library to generate a high variety of appearances [5]. This includes pedestrians with different gender and age (adult, kid) dressed in random clothing samples from a pre-specified wardrobe.

To create realistic and diverse scene settings and to differ from an overall static impression, we modify each pedestrian's dynamic behavior such as initial random speed and heading. Finally, once the actor has been successfully placed inside the simulator, a unique instance identification is assigned which is fundamental to Section 3.2.

4.2. Synthetic CARLA Images

To generate our scenes, we focus on urban environments similar to SYNTHIA [20]. Therefore, our dataset solely contains scenes from CARLA's virtual city *Town03*. Pixel-level semantic annotations include 23 different classes (*e.g.* the pedestrian class depicted in Figure 2b), which are divided into seven categories: sky, construction, human, flat, nature, vehicle, and object.

Table 1: On the basis of the scenario settings from Section 4.1 (controlled pedestrian placement), we created our synthetic dataset that may be used for further DNN evaluation using instance annotations. The table depicts the total number of images to the subset of samples with pedestrians (in parentheses).

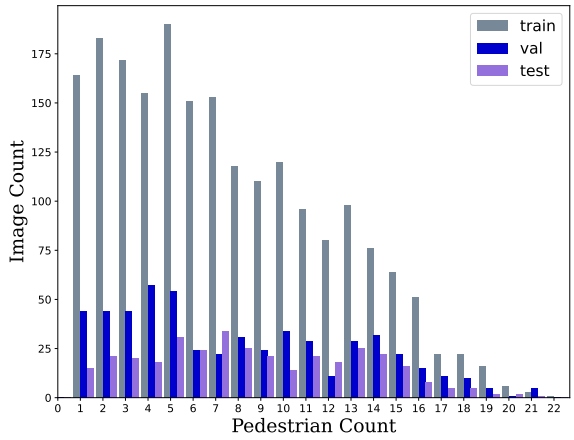
Train	Validation	Test
4559 (2111)	1227 (566)	746 (355)

We include images rendered from CARLA in our dataset with a resolution of 2048×1024 pixels. The training dataset split includes 4559 samples (with 2111 images offering pedestrian instances). Accordingly, the validation and test splits contain 1227 (566) and 745 (354) samples, respectively. For each selected scene we generate the following data: a RGB image, semantic ground truth, a depth map, as well as the novel instance segmentation map, plus additional information such as number of pedestrians along with their distances acquired from CARLA.

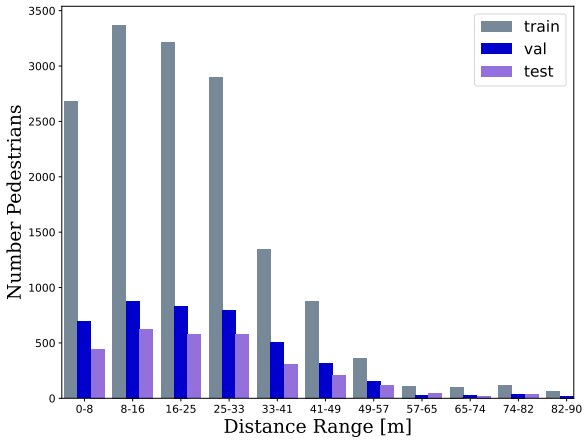
Benchmark datasets in AD, *e.g.* Cityscapes and KITTI [10, 13], provide a statistical representation using the frequency of scenes with a certain number of traffic participants (pedestrians and vehicles). However, those dataset statistics describe an overall scene complexity rather than being class specific. In Figure 5a, we show explicitly the distribution of pedestrian instance occurrences in our

[4] See CARLA's documentation on *carla.Waypoint* API Reference

[5] Actor's appearance described in CARLA 0.9.6 Release `http://carla.org/2019/07/12/release-0.9.6/`



(a) Occurrence distribution of the pedestrian class.



(b) Distribution of camera-instance distances for the pedestrian object class.

Figure 5: Among the dataset splits from Table 1, Figure b and Figure a depict the distance distribution and pedestrian instance occurrence, respectively. The figures outline that our pedestrian distribution is fairly similar across the validation and test split.

dataset. Please note that fluctuations in the requested subject count are caused by CARLA's internal collision conflicts if *e.g.* the aimed spawning location is already occupied by another object, or by successfully spawned pedestrians that are fully occluded. Nevertheless, our dataset exhibits challenging crowded scenarios with a pedestrian count up to 21 [6] instances across all splits.

Figure 5b shows that our rendered dataset covers a wide range of pedestrian distances (up to $90m$) across all dataset splits, highly similar to the nuScenes dataset [5]. In comparison, the AD dataset A2D2 offers a maximal pedestrian distance of $\approx 62m$. Other datasets like *e.g.* Cityscapes and KITTI [10, 13] provide a histogram of object distances over a wider range (up to $249m$ and $121m$, respectively) but unfortunately, the statistics only consider the vehicle class.

[6] Maximal pedestrian count in the test split.

5. Experimental Comparison

In this section we conduct dedicated experiments to show that even optimally fitted bounding boxes decrease the depth precision on our test dataset. Please note, that in our case per-pixel depth information (see Figure 4) is available and only per-instance depth aggregation needs to be performed and evaluated.

5.1. Baseline: Bounding Box Retrieval

Mimicking the approach towards instance segmentation and object detection in CARLA from [4, 17], we use the instance mask resulting from Section 3.1 as a basis for 2D bounding box (BBox) retrieval. To investigate the impact of a synthetic BBox expansion on pedestrian's depth precision we conduct our experiments using an optimal BBox, complemented by 10% and 20% extension. We use the diagonal of the BBox as a basis for a proportional expansion from mass center. We conduct those extensions to investigate how the BBox's size impacts the depth precision of the baseline approach in contrast to our presented methodology.

Please note that the pixel-wise distance from the depth sensor (8 Bit) is implemented using CARLA's logarithmic scale (*carla.ColorConverter* class), accordingly a conversion to linear depth following [8] is conducted. Subsequently, we used 8 Bit granularity, as well as logarithmic depth in our evaluation. However, to leverage CARLA's 24 Bit depth map [7] settings (raw depth data) further studies would be interesting.

5.2. Results

In the following we report on the comparison between depth determined on the basis of a BBox in comparison to the instance segmentation. This will show which systematic issues and idiosyncrasies result when using a BBox for distance annotation for a pedestrian rather than an instance segmentation in crowded scenes. As can be seen in Figure 6, we base our comparison on heatmap plots that show how many pedestrians have a certain distance from instance segmentation (x-axis) as well as for the BBox (y-axis). For the BBox, we aggregate the associated distance of all pixels inside the BBox belonging to the pedestrian class using the median in Figure 6. Note that the plot is a logarithmic heatmap, such that empty cells feature no pedestrians, dark cells feature a few pedestrians and brightly colored cells a large number of pedestrians. For the instance segmentation GT, we generally use the median. Please note that the

[7] https://carla.readthedocs.io/en/0.9.12/ref_sensors/#depth-camera

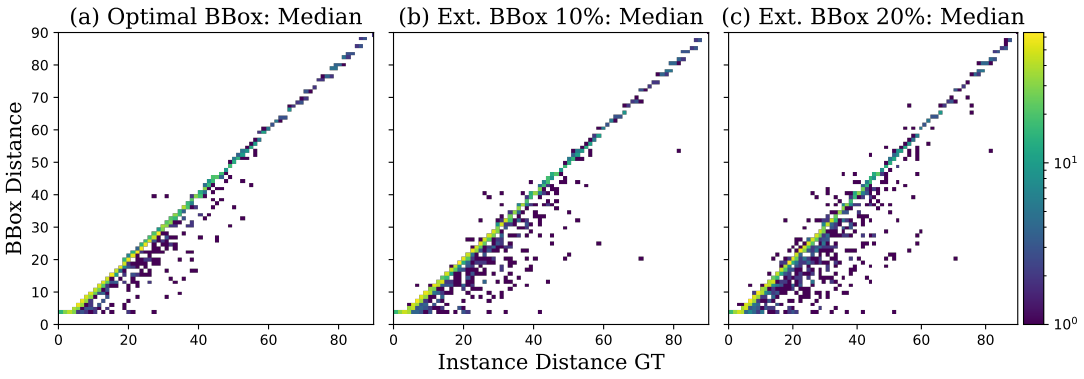


Figure 6: Logsscale heatmaps show the relationship between instance GT and BBox distances for each pedestrian. *(a)* shows an accurate BBox fit to our instance mask (baseline approach, see Section 5.1), while *(b)* and *(c)* show an applied extension on the diagonal of *(a)* using 10 and 20%, respectively.

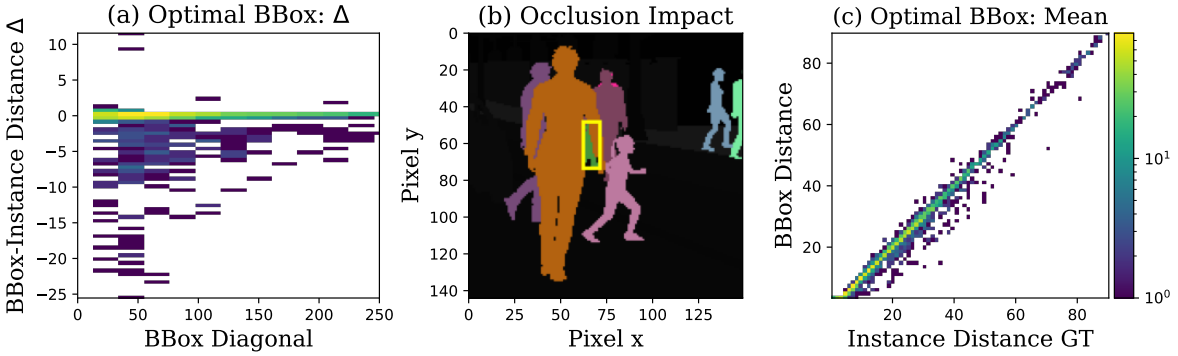


Figure 7: *(a)* shows the error of the BBox methodology in respect to its diagonal. *(b)* shows a concrete example to visualize occlusion effects on BBox distance. (c) reproduces the results from Figure 6 *(a)* using a *mean* aggregation.

maximal difference between mean and median is $\approx 0.18m$ across all test instances. For the BBox, we investigate this difference between mean and median further below. If the BBox would return exactly the same distances as the instance segmentation – which would be the case for fully separated pedestrians (no grouping) – the plot would show just a diagonal. As we can see from the plots, this main diagonal is still clearly visible, but there is some "noise" surrounding it. This shows that for some pedestrians the BBox method does not return accurate distance results.

Figure 6 shows three different variants of the same plot. We increase the size of the BBox from an optimal (minimal) size *(a)* and increase the size in steps of 10% for *(b)* and *(c)*. As a result, we can see the expected effect that results get noisier when the BBox size increases and there is more potential for other pedestrians getting mixed into a BBox. More interesting, we see that there is a bias in differences between the BBox and instance segmentation: Distances determined by the BBox get mostly smaller rather than larger. Note that this difference is quite strongly pulling certain pedestrians to a close-by distance of a few meters.

We analyze this effect further in Figure 7*(a)* by plotting the difference between the two distances (y-axis) conditioned on the size of the bounding box (x-axis). As we can see in the plot, many smaller BBoxes are affected. This is not surprising: When we have smaller BBoxes, a change of the distribution of distances in the BBox and therefore the median is easier to achieve. In order to highlight such as case, we plot one of these examples from the bottom of Fig. 7*(a)* in Fig. 7*(b)* with the corresponding BBox . We can see the occluded pedestrian (green) where only the visible parts are surrounded with a BBox. The occluding pedestrian (orange) however takes up most of the bounding box and as a result the BBox distance will be dominated by the orange rather than the green pedestrian. Concretely, while the instance segmentation results in a distance of $36m$, the orange pedestrian in $\approx 15m$ distance pulls the green pedes-

trian $21m$ forward. Please note that changing the BBox size has not much impact in these cases: the occluding close-by orange pedestrian will always dominate the distance in the BBox. This effect is rather caused by dense placement of pedestrians in crowded scenes and shows that when we want to study such a context, a distance measurement-based on instance segmentation will result in considerably better measurements.

As a final study, let us look at Fig. 7*(c)* where we study the same setup as Fig. 6*(a)* but now with a mean averaging of distance in the bounding box. We can see two effects: First, the pedestrians are not as much pulled to the front as for the median due to averaging effects. For instance, for the example in Fig. 7*(b)*, a median distance on the bounding box results in $\approx 15m$ while the mean distance results in $\approx 25m$. While the averaging helps in reducing this "pulling-effect", it results in substantially more noise: we can see that there are considerably more samples that report a larger distance. Additionally, examples farther away ($> 60m$) are increasingly affected.

6. Conclusions

In this paper, we presented a back projection pipeline that allows us to obtain accurate instance segmentation maps for CARLA. Our CUDA-enabled implementation applies a chain of coordinate transformations to efficiently back project pedestrian pixels into CARLA and to the corresponding actors. For evaluating our instance segmentation, we included a module for controlled pedestrian placement within CARLA, enabling the creation of urban scenes with pedestrians in a wide range of distances and for controlling pedestrian density, *i.e*. including crowded scenes. On this basis, we systematically constructed a synthetic dataset containing accurate instance annotations, as well as per-pedestrian distance information derived from the instance masks We performed a detailed comparison with a bounding box-based approach which identifies that in crowded scenes only the instance-based approach can resolve occlusions among pedestrians and return accurate distance information. Since our instance segmentation methodology delivers precise per-pedestrian annotations and distances for dense scenes with a high degree of overlap, this approach can be used to generate large-scale high-quality datasets with accurate per-pedestrian statistics. Furthermore, subsequent work targets the evaluation of perception functions considering physical cues for safety-argumentation, where estimated per-instance distances must be accurate.

References

[1] NVIDIA DRIVE Sim. `https://developer.nvidia.com/drive/drive-sim`.

[2] KI Absicherung – Safe AI for automated driving. `https://www.ki-absicherung-projekt.de/`, 2019.

[3] Jan-Aike Bolte, Andreas Bar, Daniel Lipinski, and Tim Fingscheidt. Towards corner case detection for autonomous driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 438–445. IEEE, 2019.

[4] Åsmund Brekke, Fredrik Vatsendvik, and Frank Lindseth. Multimodal 3d object detection from simulated pretraining. *CoRR*, abs/1905.07754, 2019.

[5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[6] Panpan Cai, Yiyuan Lee, Yuanfu Luo, and David Hsu. SUMMIT: A simulator for urban driving in massive mixed traffic. *CoRR*, abs/1911.04074, 2019.

[7] Yun Chen, Frieda Rong, Shivam Duggal, Shenlong Wang, Xinchen Yan, Sivabalan Manivasagam, and Raquel Urtasun Shangjie Xue, Ersin Yumer. Geosim: Realistic video simulation via geometry-aware composition for self-driving. In *CVPR*, 2021.

[8] Felipe Codevilla. carla-simulator. `https://github.com/carla-simulator/data-collector/blob/master/carla/image_converter.py`, 2018.

[9] Congata. Pedestrian dataset. `https://www.cognata.com/pedestrian-datasets/`, 2021.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[11] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proc. Annual Conf. on Robot Learning*, pages 1–16, 2017.

[12] M. Everingham, L. Gool, Christopher K. I. Williams, J. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2009.

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[14] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset. 2020.

[15] Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 3db: A framework for debugging computer vision models. In *Arxiv preprint arXiv:2106.03805*, 2021.

[16] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft

COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[17] Maria Lyssenko, Christoph Gladisch, Christian Heinzemann, Matthias Woehrle, and Rudolph Triebel. From evaluation to verification: Towards task-oriented relevance metrics for pedestrian detection in safety-critical domains. In *Workshop on Safe Artificial Intelligence for Automated Driving*, 2021.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[19] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mrti Moeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.

[20] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, 2016.

[21] Assaf Sela. Congata. `https://youtu.be/HFZHcREnlkw`, June 2021.

[22] Assaf Sela. dspace. `https://www.dspace.com/en/pub/home/applicationfields/stories/autonomous-driving.cfm`, 2021.

[23] Shital Shah, Ashish Kapoor, Debadeepta Dey, and Chris Lovett. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *Field and Service Robotics*, pages 621–635, November 2017.

[24] G. Volk, J. Gamerdinger, A. v. Bernuth, and O. Bringmann. A comprehensive safety metric to evaluate perception in autonomous systems. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020.

[25] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018.

[26] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.