DRIVETRUTH: Automated Autonomous Driving Dataset Generation for Security Applications

Raymond Muller
Purdue University
mullerr@purdue.edu

Yanmao Man
The University of Arizona
yman@email.arizona.edu

Z. Berkay Celik
Purdue University
zcelik@purdue.edu

Ming Li
The University of Arizona
lim@arizona.edu

Ryan Gerdes
Virginia Tech
rgerdes@vt.edu

***Abstract*—With emerging vision-based autonomous driving (AD) systems, it becomes increasingly important to have datasets to evaluate their correct operation and identify potential security flaws. However, when collecting a large amount of data, either human experts manually label potentially hundreds of thousands of image frames or systems use machine learning algorithms to label the data, with the hope that the accuracy is good enough for the application. This can become especially problematic when tracking the context information, such as the location and velocity of surrounding objects, useful to evaluate the correctness and improve stability and robustness of the AD systems.**

In this paper, we introduce DRIVETRUTH, a data collection framework built on CARLA, an open-source simulator for AD research, which constructs datasets with automatically generated accurate object labels, bounding boxes of objects and their contextual information through accessing simulation state and using semantic LiDAR raycasts. By leveraging the actual state of the simulation and the agents within it, we guarantee complete accuracy in all labels and gathered contextual information. Further, the use of the simulator provides easily collecting data in diverse environmental conditions and agent behaviors, with lighting, weather, and traffic behavior being configurable within the simulation. Through this effort, we provide users a means to extracting actionable simulated data from CARLA to test and explore attacks and defenses for AD systems.

I. INTRODUCTION

Autonomous driving (AD) systems are now pervasive, leading to a greater need for rapid and widespread development of datasets to evaluate their correctness, safety, and security. Applications require different types of datasets. For example, a simple anomaly detection system might be built on a 2D video dataset, with 2D labeled object bounding boxes as ground truth. More complex systems require 3D bounding boxes, combined with LiDAR data detailing the velocities of other objects for detailed scene perception.

However, it is difficult to collect datasets in real-world settings with correct annotations under varying environments, such as different weather, geographic locations, aggressiveness of other cars, and traffic conditions. One approach to obtain annotated data is to use machine learning (ML) algorithms to classify raw data. For example, the popular 2D video dataset BDD100K [21], known as the Berkeley Deep Drive Dataset, classifies objects using the DLA-34 algorithm with a 50-60% accuracy. The 3D bounding-box datasets such as nuScenes [3] and WayMo [19] rely on manually labelling of the recorded data from their sensor readings to ensure accuracy, before extrapolating the readings to tangible values, such as 3D bounding boxes or object velocities. However, manually labeling a large number of data points is time-consuming and requires human expertise. Additionally, collecting data with different weather parameters, initial object positions, vehicle velocities, other cars, and pedestrian behaviors requires prohibitive amounts of time. This limits researchers from exploring a large amount of data patterns and defining benchmarks to fairly compare attacks and defenses against AD systems.

In this paper, we introduce DRIVETRUTH, a data collection framework for CARLA[1] with automated dataset annotations. CARLA is a popular open-source simulator for autonomous driving research, built on Unreal Engine 4. It uses the state-of-the-art rendering engine, physics simulation, and NPC logic components to simulate a dynamic world with multiple interacting agents. The simulator enables us to create a realistic 3D environment from which we collect data similar to the real-world scenarios [6]. DRIVETRUTH enables users to configure environmental states and agent behaviors at data collection, e.g., we obtain 2D or 3D bounding boxes of vehicles from the simulation stored as part of the car's 3D model. The weather conditions such as the precipitation on the ground, sun altitude angle, and the behavior of other cars and pedestrian are customizable for each simulation to obtain an arbitrary amount of patterns under different scenarios.

DRIVETRUTH collects a 2D video dataset, splits it into frames with the raw video, bounding-box-labeled video, LiDAR projection video, and class-segmented video, and separately stores each. Additionally, for each frame, it stores the ego vehicle's (collector) context, in the form of speed, position, heading, and control information (throttle, steering wheel angle, etc.) Similarly, for the non-ego object context, the internal ID, bounding box coordinates, position, class, relative velocity, and distance of every vehicle, pedestrian, traffic light, and traffic sign (by default) in-frame are stored. Lastly, DRIVETRUTH tracks any type of the desired object in an environment, whether or not it has a set bounding box within the CARLA simulation through a semantic raycast system used for the traffic lights and traffic signs. During data collection, the parameters such as the number of vehicles/pedestrians, weather distribution, and collection time can be customized. As DRIVETRUTH is built on

[1] https://carla.org/

Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2022
24 April 2022, San Diego, CA, USA
ISBN 1-891562-75-4
https://dx.doi.org/10.14722/autosec.2022.23032
www.ndss-symposium.org

TABLE I: Comparison of DRIVETRUTH against commonly used Autonomous Driving datasets

Dataset Properties	KITTI [8]	Berkeley Deep Drive [21]	WayMo [19]	NuScenes [3]	ApolloScape [10]	Cityscapes [5]	DriveTruth
2D Bounding Box Support	✓	✓	✓	✓	✓	✓	✓
3D Bounding Box Support	✓	✗	✓	✓	✓	✗	✓
LiDAR Support	✓	✗	✓	✓	✓	✗	✓
Ego Vehicle Context†	P, T	P, T	P, T	P, T	P, T	P, T	P, T, C
Non-Ego Vehicle Context†	✗	✗	P, T	✗	P, T	✗	P, T
Accurate Ground-truth	✓	✗	✓	✓	✓	✓	✓
Automated Labels	✗	✓	✗	✗	✗	✗	✓

† P refers to tracking the **position** of the object(s), T refers to tracking the **trajectory** (velocity and/or acceleration), and C refers to tracking the **control information** such as throttle, brake, and steering angle of a vehicle.

top of the CARLA simulator, it works in tandem with other scripts or simulator modifications, allowing further tailoring of the system to suit the needs of users.

Related Work. We provide a brief survey of public real-world datasets and methods that use CARLA for dataset creation relevant to the development of AD applications. We focus on commonly used datasets leveraging multimodal sensor data.

We first compare the DRIVETRUTH with the datasets collected from a real environment. Table I presents the capabilities and drawbacks of datasets generated by DRIVETRUTH with popular AD datasets. There exist multiple datasets with 3D bounding boxes and varying amounts of context in the form of ego vehicle and non-ego vehicle trajectories. However, these datasets require manually annotating either the sensor data or raw image pixels, which are then extrapolated into usable data. This process involves multiple human experts and several passes to verify their correctness. Other datasets use ML to automatically annotate, however, these approaches sacrifice correctness. For instance, the BDD100K dataset automatically labels the classes of individual pixels, yet it does not provide completely accurate ground truth. Lastly, in addition to having support for all listed types of data and accurate ground-truth and automatic labels, DRIVETRUTH also includes an extra piece of context in the form of control information. This information is useful for determining the intent of the ego vehicle. For example, by knowing the angle of the steering wheel, we can tell if the vehicle attempts to turn left, right, or moving straight.

Other works have leveraged CARLA to collect realistic and accurate datasets by reasoning directly from the simulation state. The Automatic Vehicle 2D Bounding Box Annotation Module for CARLA Simulator [1] collects a pure image dataset with the 2D bounding boxes of vehicles. This approach implements several functions to transform 3D bounding boxes represented by CARLA to a 2D bounding box that the camera sees, which we have integrated into DRIVETRUTH. The Carla Dataset Runner [4], collects both pedestrian and vehicle bounding boxes. The Scenic Language can be used to generate different autonomous driving scenarios in several simulators, including CARLA [7]. Finally, the Carla Scenic Data collector leverages CARLA to collect data on anomalous driving scenarios [18]. However, none of these approaches track contextual information, e.g., object speed and positions and the bounding boxes of static objects, e.g., traffic lights and traffic signs, fences, poles, lane markers, sidewalks, and guard rails. This is because static objects in CARLA by default do not have simulation-stored bounding boxes. In addition, CARLA supports user-defined and added objects. Unless the object's 3D model has a bounding box object attached, the object is static by default and can not be tracked by these systems. Rather than using ML algorithms to label these static objects, which have a margin of error, we introduce a novel way to track the 3D bounding box of each static object.

Another line of work uses CARLA as a simulation environment to develop reinforcement learning models for driving [13], build multi-sensor fusion for end-to-end autonomous driving [20] , and uses conditional affordance learning in urban environments [16]. However, these approaches mainly focus on addressing specific problems rather than flexible frameworks to generate datasets for AD systems.

Contributions. We seek to overcome the shortcoming of previous works to automate generating AD datasets from the CARLA simulator with configurable environmental states and agent behaviors. DRIVETRUTH provides the following additional properties over existing approaches to collect datasets at scale:

- DRIVETRUTH stores **contextual data** about the ego vehicle and the relevant objects in the scene, which supplements the bounding boxes and sensor data. This provides a more rigorous verification of the performance of an AD system, particularly for safety and security applications.
- DRIVETRUTH extends CARLA's semantic LiDAR (raycast) system to provide **3D bounding boxes and locations of static objects**, including traffic lights and traffic signs. The above contextual data is also applied to them.
- DRIVETRUTH code is available at https://github.com/purseclab/DriveTruth, for public use and validation.

We note that the semantic LiDAR system allows DRIVETRUTH to track a wider variety of objects than normally allowed in CARLA, extending to every object in the simulated environment. The ability to accurately track all objects and their context, including position and velocity, is crucial for rigorous AD experimentation and testing. For example, the WayMo dataset explicitly uses LiDAR, radar, and annotated maps to track the context of surrounding objects while driving and uses external cameras to track traffic lights [9]. Tesla Autopilot integrates a multitude of sensor arrays and GPS data to track

TABLE II: Default Semantic Tags provided in CARLA (0.9.11), all of which can be tracked by DRIVETRUTH

Value	Tag	Description
0	Unlabelled	This category is unused by default, but is assigned to objects with no collision boxes (i.e., intangible objects)
1	Building	Buildings like houses, skyscrapers, and the elements attached to them (air conditioners, scaffolding, awnings, ladders, etc.)
2	Fence	Wood or wire structures that enclose an area of ground.
3	Other	Objects that do not fit into any other category.
4	Pedestrians	Humans that walk or ride/drive any vehicle or mobility system.
5	Pole	Small, mainly vertically oriented poles, but horizontal poles like those carrying traffic lights are also included.
6	Road Line	Markings along the road
7	Road	Parts of the street on which cars usually drive.
8	Sidewalk	Parts of the ground reserved for pedestrians and cyclists. Delimited by some obstacle like a curb, pole, traffic island, or markings.
9	Vegetation	Trees, hedges, and any other vertical vegetation. Ground vegetation is considered part of the terrain category.
10	Vehicles	Cars, vans, trucks, motorcycles, bikes, busses, and trains.
11	Walls	Individual standing walls that are NOT part of buildings.
12	Traffic Sign	Signs installed by a state authority, usually to regulate traffic. This includes only the sign itself, not the pole on which the sign stands.
13	Sky	The open sky, including the clouds and sun.
14	Ground	Any horizontal ground structure that doesn't fit into any other category, such as areas shared by both vehicles and pedestrians.
15	Bridge	The structure of the bridge, not including fences, people, vehicles, and other elements.
16	Rail Track	Rail tracks not drivable by cars, such as subway or train tracks.
17	Guard Rail	All types of guard rails and crash barriers.
18	Traffic Light	The traffic light box, not including the poles.
19	Static	Unmovable props in the scene, like fire hydrants, fixed benches, fountains, bus stops, etc.
20	Dynamic	Elements that can move over time, like movable trash bins, wheelchairs, animals, etc.
21	Water	Horizontal water surfaces like lakes, seas, and rivers.
22	Terrain	Grass, ground level vegetation, soil, and sand.

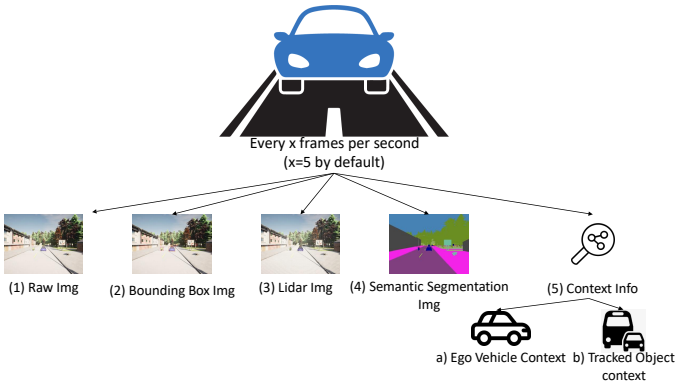

Fig. 1: Data types collected by DRIVETRUTH

different unmoving objects, such as lane markers and traffic lights, and tracks different vehicles in 3D [11].

While DRIVETRUTH generates simulated data rather than real-world data, specialized simulated data has been used for the continued development of existing AD systems. For example, Baidu's Apollo AD architecture deployed to the robotaxis comes with a special game-engine-based simulator [2], similar to CARLA. It is built to help engineers test and validate new algorithms for Apollo's perception module. Although DRIVETRUTH is intended for more general autonomous vehicle and vision-based systems, it provides similar functionality to create large datasets to test and validate AD systems.

II. DRIVETRUTH

DRIVETRUTH is built on CARLA 0.9.11, the latest version of CARLA at the time of writing. It automatically labels 2D bounding boxes (with 3D bounding boxes support) for vehicles, pedestrians, traffic signs, and traffic lights. It also supports tracking and labelling of 23 objects listed in Table II, as well as user-defined objects set up with the proper bounding boxes and tags. Control information such as collection rate (frames per second), simulation runtime, number of simulations to run, number of vehicles, and number of pedestrians can be set through provided parameters. For environmental conditions, e.g., rain, clouds, wind, and sun angle, we provide in-code variables to specify a weather distribution, and provide a list of presets that are randomly selected.

DRIVETRUTH, with these parameters set, provides a list of maps classified as either "residential" or "highway", representing either a high-density area with many pedestrians for the former or a lower-density, higher-speed road with little pedestrians for the latter. Users are able to update the DRIVETRUTH's map list to include different maps available in the simulator, and add their own maps to CARLA, which can be easily integrated into DRIVETRUTH's default maps.

The user then specifies a frame rate at which the system starts collecting data. Figure 1 shows the five data types outputted by DRIVETRUTH. First, the simulated sensors capture the information about the surrounding world, outputting the raw image, LiDAR image transposed onto the camera, and a segmented image. DRIVETRUTH then accesses information about the simulated world to find vehicles, pedestrians, and objects that the user has specified to track within the code, such as traffic lights and signs. Depending on whether the object has a built-in bounding box for its 3D model, DRIVETRUTH uses different techniques to extrapolate the 3D bounds of the object to generate the bounding box and location. This data is drawn onto another image and saved, before polling the attributes of the objects to retrieve object contexts, such as position and velocity. Lastly, it polls the ego vehicle for its position, heading, velocity, and its control data including throttle, steer, and brake, useful to determine the vehicle's intent.

A. Bounding Dynamic Objects

For moving objects in CARLA, extracting the bounding box of pedestrians and vehicles is not challenging because they are by default provided along with the 3D model. For instance,

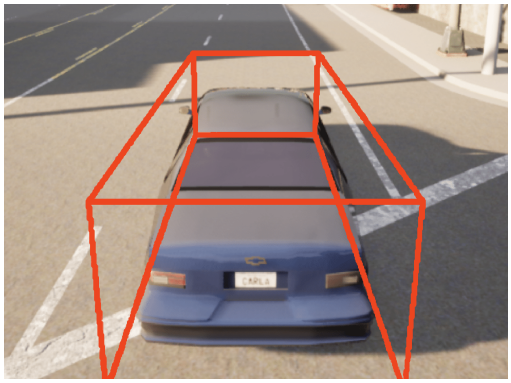

Fig. 2: An Illustration of 3D vehicle with its bounding box.

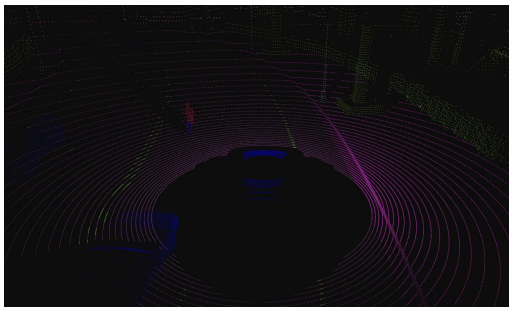

Fig. 3: An illustration of semantic point cloud [17].

Figure 2 shows a vehicle's 3D model with its bounding box obtained by DRIVETRUTH.

To compute the bounding box of dynamic objects, we first set up the ego vehicle with several sensors, including a LiDAR and a forward-facing camera, used to filter out the relevant data for 2D bounding box collection. When collecting a frame as dictated by the set FPS rate parameter, DRIVETRUTH gathers all vehicles and pedestrians in the scene at the time of capture. It then filters out those not visible to the camera by taking the LiDAR data. The filtering process is based on the camera's Field of view (FOV) and a set max distance. It then removes any objects, not in the point cloud. Thereafter, DRIVETRUTH extracts the bounding boxes from each remaining model and translates them to the camera's 2D space, which is then stored along with the rest of the object information.

DRIVETRUTH additionally pulls a vehicle's information from the simulation, such as its internal ID, relative velocity, location, and distance from the ego vehicle. Every static or dynamic object in CARLA has a location, rotation, and velocity, which allows DRIVETRUTH to access and compare with the ego vehicle to obtain measurements relative to the other objects.

B. Bounding Static Objects

Static objects in CARLA do not have bounding box data associated with them; thus, they cannot be directly extracted as opposed to vehicles or pedestrians. DRIVETRUTH uses the *semantic LiDAR sensor*, positioned in the same place as the radar, to extract information from the scene for computing the bounding box of any static object.

Algorithm 1 Point Cloud to 3D Bounding Box

Input: List of points in point cloud p_c, list of sought semantic tags s
Output: Dictionary of Bounding Boxes as {"id": (center, extent)}

function POINT_CLOUD_TO_3D_BBOX(p_c, s)
  $bounded\_objects = []$
  for $p \in p_c$ **do**
    if $p.id$ not in $bounded\_objects$ and $p.tag$ in s **then**
      $bounded\_objects \leftarrow p.id$
    end if
  end for
  $3d\_extent = \{\}$
  for $o \in bounded\_objects$ **do**
    Initialize $3d\_extent[o]$ with min/max points in each axis x_min, x_max, y_min, y_max, z_min, z_max all null
  end for
  for $p \in p_c$ **do**
    if $p.id \in bounded\_objects$ and $p.tag \in s$ **then**
      Update $3d\_extent[p.id]$ with new min/max points in each axis as necessary
    end if
  end for
  $3d\_bboxes = \{\}$
  for $o \in bounded\_objects$ **do**
    Compute center/extent from coordinates in $3d\_extent[o]$
    $3d\_bboxes[0] = (center, extent)$
  end for
  return $3d\_bboxes$
end function

The semantic LiDAR sensor works similarly to the LiDAR simulator but it uses *raycasts* that each hit a point in the world and expose information about the point. A single raycast is a single ray that shoots out in a given direction and stops as soon as it hits a point. Unlike the LiDAR, the raycast is not a "true" point but an internal abstraction; thus, it is not subject to intensity, drop-off, or noise of a normal light beam.

DRIVETRUTH fires millions of raycasts per second around the vehicle and obtains a point cloud that blankets its surroundings. To illustrate, we show an example of point cloud data in a top-down view of the ego vehicle (in blue) in Figure 3. Each point in the point cloud contains the following four information:

1) **Point coordinates** in (x, y, z)
2) **Cosine angle** between the angle of incidence of the raycast and the normal of the surface the raycast hit
3) **Internal ID** of the object hit by raycast
4) **Semantic tag** of the object hit by raycast

The last two items are of tantamount importance as they enable DRIVETRUTH to distinguish individual objects within the point cloud (from their internal ID) and their classes.

We manually add the list of default semantic tags provided in CARLA (Table II) to the mesh of each unique object within the simulator to identify the class of each object For example, for a traffic sign on a pole, the pole has the "pole" semantic tag while the actual sign has the "traffic sign" semantic tag. Because tags are specified per-object, a user is also able to add or redefine their own semantic tags from the CARLA source code, which are then usable by DRIVETRUTH.

With these tags and the point cloud, DRIVETRUTH obtains an

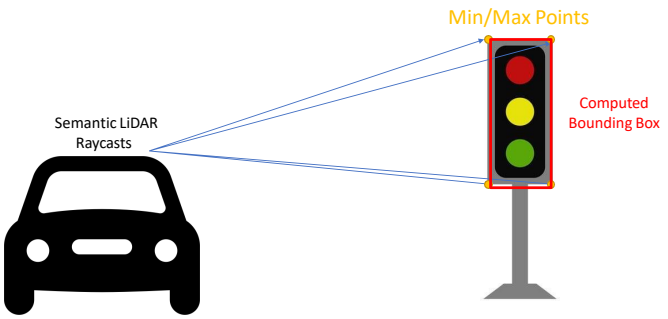

Fig. 4: Illustration of using point cloud data for extracting the bounding box of a traffic light.

Fig. 5: An example frame collected by DRIVETRUTH (best viewed when printed in color)

object and class for every point in the point cloud. Algorithm 1 details the process of converting the point cloud to a set of 3D bounding boxes. We first scan the point cloud for the desired tags, and find all object IDs with points associated with tag (Lines 2-7). We then set up a dictionary that stores the min and max points of the object in each axis, and return the final bounding box (Line 8). Lastly, we go through the point cloud again, scan the portions of the objects with the relevant tags, and update the min and max points (Lines 9-16). Once the entire point cloud is scanned, we obtain the min and max points on each axis for each of our sought objects, which we turn into a bounding box center and the extent to return as a final dictionary (Lines 17-22), as shown in Figure 4.

We use the computed center of the 3D bounding box as the object's location to store the contextual data. For the rest of the contextual data (e.g., velocity), we use the Object ID to pull the simulated state of the object and get the measurements. This is because, for objects like traffic signs, the internal object includes more than just the desired object. For example, it may include the sign and pole rather than just the sign. This means that the most accurate position measurement is the one we measure via raycast, but other measurements such as velocity can be obtained directly from the object. Once the bounding boxes are obtained, each object is assigned to its computed bounding box before being processed similarly to the dynamic objects. We use LiDAR to filter out objects that are not in the camera's view, and extract the bounding boxes and translate them into 2D space.

III. IMPLEMENTATION

DRIVETRUTH is written in Python 3.7, with a combined total of 1252 LoC. In addition to CARLA, it relies on the NumPy, Pillow (PIL), and OpenCV libraries to render the bounding boxes. DRIVETRUTH runs on the latest version of CARLA (version 0.9.11) at the time of writing. To collect dataset through DRIVETRUTH, after CARLA is installed, parameter can be configured using the collect_dt_data.py script, and then data collection process can be started. We provide details of parameters on the project GitHub page.

Figure 5 shows a sample DRIVETRUTH frame with two vehicles and a traffic sign, with the 2D bounding boxes transposed onto the image. DRIVETRUTH takes on average 5-10 secs to initialize a run and a further 45-50 mins to capture 1000 frames (200 secs) of data on a laptop equipped with an NVIDIA GTX 1060 and 16 GB of RAM, set with the system default of a 5 fps capture rate.

IV. DATASET USE CASES

Datasets generated by DRIVETRUTH contain a wealth of data, which can be leveraged for diverse AD security and safety applications, including model pre-training, multi-object tracking, and multi-sensor fusion. For instance, neural network-based object perception (e.g., object detection, image segmentation) models for AD systems are typically deep, with many layers; thus, they require a large amount of data for training. Due to the DRIVETRUTH's realistic simulated world, users can pre-train a deep neural network (DNN) model with a large-scale dataset with precise labels generated by DRIVETRUTH, transfer the knowledge to the DNN model to the real-world small-scale datasets [19], [21] for fine-tuning.

Evaluating Dataset Usability. We show that datasets generated by DRIVETRUTH are close enough to real-world images and can be used for pre-training.

We first tested the official model of YOLOv3 [15], previously trained on the COCO object detection dataset [14], on a sample DRIVETRUTH dataset containing 500 objects. We obtained an average precision of 47.2% at 50% Intersection over Union (IoU) with the ground-truth bounding box. This means that 47.2% of the bounding boxes computed by YOLO overlap at least 50% of the ground truth. These results are similar to YOLOv3's performance on the COCO dataset.

Second, to test object tracking performance on our dataset, we feed the RGB images to a pre-trained DaSiamRPN [22] network, the state-of-the-art Siamese network for object tracking. We give the ground-truth bounding box of the first frame to DaSiamRPN and then let it track the object from all subsequent frames. For each frame, DaSiamRPN returns a bounding box for the object, with which we compare the ground-truth bounding box using IoU. Results on a randomly-sampled dataset with 500 objects show an average precision of 77% at 50% IoU is achieved. This means DRIVETRUTH can be used to train and test object tracking algorithms.

Similarly, DRIVETRUTH can be used to collect data to either pre-train an end-to-end model or train a Kalman filter-based

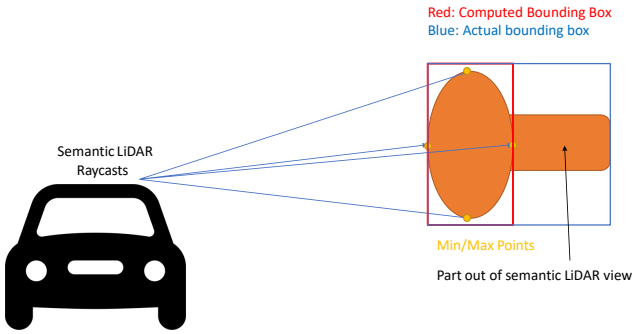

Fig. 6: Illustration of an object with unique geometry that could throw off the 3D bounding box extents

object tracking model [12]. Since DRIVETRUTH generates data from both the camera and LiDAR sensors, it facilitates studying multi-heterogeneous-sensor fusion algorithms.

Usage of Contextual Data. The contextual information provided by DRIVETRUTH can be used to enrich the robustness of the above applications. In most video datasets, context information is not provided, and therefore all information is relative. For example, if a bounding box moves to the left of the frame, there is no way for a system to know if the object is moving left or the camera is turning right. With context information, it is possible to know the *absolute* state of the environment under which the data is collected. For security and safety applications relying on precise, absolute measurements of the environment in order to make decisions, this context information is vital. To illustrate, with the ego velocities provided by context information, a Kalman filter algorithm can predict future bounding boxes for tracking purposes more precisely than without using context information.

V. LIMITATIONS AND DISCUSSION

3D Bounding Boxes. DRIVETRUTH obtains the 3D bounding boxes of dynamic objects, cars, and pedestrians. For static objects, it includes functions to detect traffic lights and traffic signs without any issues. However, when encountering a special type of elongated geometry in static objects, it may find a precise 2D bounding box, yet it may fail to find a precise 3D bounding box because of the occlusion from the semantic LiDAR. We currently use a single semantic LiDAR positioned on top of the vehicle to generate a point cloud. If the LiDAR sees the whole object (or at least the minimum and maximum extents of the object), the 3D bounding box is precise. If more parts are occluded, the less precise 3D bounding boxes are obtained, although the 2D bounding box remains precise as the camera only sees the non-occluded parts. For flat objects such as traffic signs and traffic lights, because they are placed perpendicularly to the direction of travel on roads, they are always viewed head-on or from an angle where their extents can clearly be seen; thus, occlusion is not an issue.

To illustrate, consider an object with unique geometry in Figure 6. If an object is viewed head-on and has geometry built to block the view of the camera or LiDAR, the point cloud could potentially miss the maximum extent of the 3D bounding box by certain angles. For objects with this sort of strange geometry, DRIVETRUTH must use multiple semantic LiDAR sensors placed around the map and merge them into a single point cloud for analysis. In this way, there will be no occlusion, and the 3D bounding box can be extracted correctly. Future work will expand the capabilities of the semantic LiDAR system to provide more accuracy for these edge cases.

Simulator Fidelity. There might be issues in the CARLA simulator that may cause imprecise data collection. For instance, we found that, for a certain type of traffic light model at intersections, all types of LiDAR data pass through these traffic lights. This means that the detection of traffic lights with DRIVETRUTH may yield errors. We have reported this issue to the CARLA developers[2], who have acknowledged this bug and are working on a fix. As the CARLA simulator continues to steadily improve, we plan to integrate new features into DRIVETRUTH to take advantage of those extra additions to the simulation. For example, CARLA developers plan to add snow and thermal cameras in a future update. This will allow DRIVETRUTH to work with snowy conditions that would enhance the data collection in varying weather conditions.

Computation Overhead. The CARLA simulator requires a powerful GPU and at least 8 GB of RAM to run DRIVETRUTH. As we have illustrated in Figure 5, a frame collected by DRIVETRUTH on a mid-tier laptop shows a fairly high-quality and realistic image with bounding boxes. However, higher-end systems are able to run CARLA's Town 10 map fitted with highly realistic textures and meshes. In exchange for the high computational overhead, DRIVETRUTH is able to generate more photorealistic datasets.

VI. CONCLUSIONS

We design and develop DRIVETRUTH, a configurable data collection framework that leverages the CARLA simulator to collect AD data and automatically labels it. DRIVETRUTH extends the semantic LiDAR raycasts to compute the bounding boxes of static, immobile objects that do not have an assigned bounding box within the simulator. It also pulls the 3D model-assigned bounding boxes of dynamic objects, and stores the context of each object it tracks. We hope that DRIVETRUTH catalyzes the research to improve stability and robustness of the AD systems, and fulfills the desire to quickly generate high-quality data for AD applications.

REFERENCES

[1] "Automatic vehicle 2D bounding box annotation module for carla simulator," 2020. [Online]. Available: https://mukhlasadib.github.io/CARLA-2DBBox/

[2] "Apollo game engine based simulator," https://apollo.auto/gamesim.html, 2021.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[4] "Carla dataset runner," 2019. [Online]. Available: https://github.com/AlanNaoto/carla-dataset-runner

[2]https://github.com/carla-simulator/carla/issues/4099

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *Conference on Robot Learning*, 2017.

[7] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and scene generation," *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Jun 2019. [Online]. Available: http://dx.doi.org/10.1145/3314221.3314633

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, p. 1231–1237, 2013.

[9] Google, "Learn how WayMo drives - WayMo help," 2021. [Online]. Available: https://support.google.com/waymo/answer/9190838?hl=en

[10] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape open dataset for autonomous driving and its application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, p. 2702–2719, Jul 2019.

[11] S. Ingle and M. Phute, "Semi autonomous driving, an uptick for future autonomy," *International Research Journal of Engineering and Technology*, vol. 3, no. 9, Sep 2016.

[12] Y. J. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. W. Wei, "Fooling detection alone is not enough: Adversarial attack against multiple object tracking," in *International Conference on Learning Representations (ICLR)*, 2020.

[13] X. Liang, T. Wang, L. Yang, and E. Xing, "Cirl: Controllable imitative reinforcement learning for vision-based self-driving," *Computer Vision – ECCV 2018 Lecture Notes in Computer Science*, p. 604–620, 2018.

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

[15] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[16] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," *Robot Learning*, 2018.

[17] "Carla sensors reference," 2021. [Online]. Available: https://carla.readthedocs.io/en/latest/ref_sensors/#semantic-lidar-sensor

[18] T. Shah, J. R. Lepird, A. T. Hartnett, and J. M. Dolan, "A simulation-based benchmark for behavioral anomaly detection in autonomous vehicles," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 2074–2081.

[19] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, and et al., "Scalability in perception for autonomous driving: Waymo open dataset," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. Lopez, "Multimodal end-to-end autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, p. 1–11, 2020.

[21] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[22] Z. Zhu, Q. Wang, L. Bo, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *European Conference on Computer Vision*, 2018.