UNIVERSITY OF TARTU

Institute of Computer Science

Computer Science Curriculum

Kertu Toompea

Simulations for Training Machine Learning Models for Autonomous Vehicles

Bachelor's Thesis (9 ECTS)

Supervisors:

Raimond-Hendrik Tunnel, MSc

Tambet Matiisen, MSc

Tartu 2019

Simulations for Training Machine Learning Models for Autonomous Vehicles

Abstract:

Training machine learning models for autonomous vehicles requires a lot of data which is time consuming and tedious to label manually. Simulated virtual environments help to automate this process. In this work these virtual environments are called simulations. The goal of this thesis is to survey the most suitable simulations for off-road vehicles (while not discarding the urban option). Only the simulations which provide labeled output data, are included in this work. The chosen 12 simulations are surveyed based on the information found online. The simulations are then analyzed based on the predefined features and categorized according to their suitability for training machine learning models for off-road vehicles. The results are shown in a table for comparison. The main purpose of this work is to map the seemingly large landscape of simulations and give a compact picture of the situation.

Keywords:

Simulations, environments, off-road, autonomous vehicles, sensors, machine learning, automated labeling, survey, overview

CERCS: P170 Computer science

Masinõppe Mudelite Treenimise Simulatsioonid Autonoomsetele Sõidukitele

Lühikokkuvõte:

Masinõppe mudelite treenimine autonoomsete sõidukite jaoks nõuab palju andmeid, mille käsitsi märgendamine on aeganõudev. Simulatsioonid aitavad seda protsessi automatiseerida. Käesolev töö koostab ülevaate 12-st internetiotsingu abil leitud simulatsioonist ja analüüsib neid lähtuvalt nende sobivusest maastikul liikuvatele sõidukitele (säilitades võimaluse liikuda ka linnakeskkonnas).

Võtmesõnad:

Simulatsioonid, keskkonnad, maastik, autonoomsed sõidukid, sensorid, masinõpe, automaatne märgendamine, ülevaade

CERCS: P170 Informaatika

Table of Contents

1 Introduction

Development of autonomous vehicles is an active field, binding together the automotive industry and computer science. Machine learning (ML) is one of the main computer science fields that contributes to the development and testing of autonomous vehicles. However, a lot of labeled data is needed to train ML models to achieve the desired driving behavior and perception of the environment surrounding the vehicle.

The amount of data needed for training and testing ML models for autonomous vehicles is comparable to hundreds of millions (or even billions) physically driven miles [1]. It is time-consuming to create and label such an amount of data with a large variety of scenarios in the real world. Therefore virtual environments are used to simulate the real world and collect automatically labeled data faster (Waymo: 25K virtual cars running 24/7 drive 10M simulated miles per day[1]). In this thesis the virtual environments are called *simulations*. By running a simulation (or several in parallel) non-stop on a modern computer hardware, a lot more data can be generated efficiently, than a real vehicle with a driver could during the same time. Also dangerous situations in simulations do not result in any actual damage.

Based on the different vehicles' simulations found online, it is necessary to clarify that the simulations described in this work are for training ML models for autonomous ground vehicles' obstacle detection. The simulations which are for training people to drive, testing specific traffic scenarios, controlling actual steering/throttle/brake of a vehicle or simulating vehicle dynamics, are not in the scope of this work.

Besides the amount of data and a 'safe' environment, the main benefit of using the simulations in the scope of this work, is their ability to generate and provide **labeled** data for ML. Without labeling, the simulation can produce a large amount of data (e.g. batches of hundreds of thousands images) but which then needs to be labeled manually. Manual labeling is a tedious and time consuming work which is possible to avoid. Therefore automatic labeling is one of the key value points in the simulations.

There are many simulations on the market and several interesting details about them (e.g. how exactly they are made, which are most suitable for off-road use, to what extent they are configurable, what simulation specific details affect the training of ML models for the most, etc.).

[1] https://youtu.be/Q0nGo2-y0xY?t=1880

Before diving into those details about every simulation, it is useful to have an overview of simulations which are potentially suitable for further investigation. No such overview was found which would give answers to the questions introduced in the following paragraph. Therefore the survey made in this work maps the seemingly large landscape of simulations and gives a compact picture of the situation. The focus is on the availability of **sensors**, **off-road vs urban environments** and the **output training labels** needed for ML models. The results (chapter 5) aim to serve as a guide for anyone choosing a simulation for a use case that corresponds to the surveyed parameters.

This work finds answers to the following questions:

- How many of the available simulations come with built-in urban and off-road environments?
- How many have dynamic objects (e.g. humans, vehicles) in their environments?
- How many provide specific sensors (e.g. cameras, lidar, radar)?
- How many output specific training labels (e.g. semantic segmentation, bounding boxes) for ML?
- How many have a free-to-use licence?
- How many offer online documentation (i.e. installation and user guides)?
- How many are built on widely known game engines (i.e. Unreal Engine and Unity)?
- What are their constraints considering the above questions?

Related work found about existing simulations in the scope of this work was limited. A few articles were found which describe a certain simulation product (e.g. CARLA [2], AirSim [5,6]). Other found articles related to simulations and training ML models with synthetic data, mostly described a method [4,7], approach [16,22] or a use case [11,10]. One publicly available overview of simulations was found by Transport Systems Catapult from 2018 [12], but it was too broad and focused on *automated driving systems* (ADS), not training ML models for fully autonomous vehicles. One more overview report was found but it was not freely available[2]. Another seemingly promising article about different simulation platforms was found [13] but it did not include any details about actual simulations and states that many simulations were not yet available at the time of its writing. Finally, one overview of simulations was found as part of a wider review about perception systems and simulators

[2] https://www.strategyanalytics.com/access-services/automotive/autonomous-vehicles/reports/report-detail/simulation-for-autonomous-vehicles

for autonomous vehicles [14]. In chapter 5 of this work, different types of simulation platforms are introduced, partly including the simulations interesting for this work here.

As a result, a fresh overview of simulations in the focus of this work was needed. This need is mainly driven by a personal interest in creating a new simulation which is suitable for both urban and off-road vehicles, with the main focus on the **off-road**.

This work is divided into 6 chapters. The first one here is the Introduction that guided the reader into the world of targeted simulations and described the purpose of this work. The 2nd chapter, Aspects of Simulations, introduces what are the main interest points of the surveyed simulations. The 3rd chapter, Methodology and Features, describes what exactly is surveyed in the simulations and how the information was collected. The 4th chapter, Simulations, holds brief descriptions of each surveyed simulation and presents its collected information about the features. The 5th chapter, Results, shows the gathered information as a whole and draws an interpretation from the results. The 6th and the final chapter, Conclusion, briefly visits the key results of this work.

The reader of this work is expected to be familiar with ML at an introductory level and understand the terminology which is specified in the Glossary.

2 Aspects of Simulations

Before describing the methodology and features of this survey, the main aspects of simulations in the scope of this work are introduced. This chapter gives a better understanding why certain features were selected for surveying and what is their role in the simulation. The aspects cover the environments, sensors and output training labels in a simulation.

2.1 Environments

The word 'environment' in a simulation usually refers to either a driving environment (e.g. urban, off-road) or a weather conditions (e.g. sunny, cloudy, rain). While it is important to have different weather conditions in a simulation, in this work the main focus is on the driving environments.

The driving environment in a simulation defines for which kind of physical environments and vehicles the simulation can be used. For example, if an ML model is trained on the data retrieved from an urban city, it will not perform well if tested in a forest. The training and testing data are too different.

Many simulations are built using game engines. The ones most often used are Unreal Engine[3] (UE) and Unity[4]. This enables game environments to be used also in simulations. Still, attention to physics-based fidelity is required which is important for a realistic driving environment but not necessarily for a game. While it is possible to create new game environments (as game assets[5]), it is not trivial and requires dedicated skills. When using the existing high-fidelity environments available in asset stores, it can be time consuming to find the right ones, they can cost extra or there may be a technical barrier to overcome before they can be used. Regardless of the reasons behind mentioned constraints, in this work the surveyed driving environments are the ones that are already included with the simulation and can be used out of the box for both urban and off-road driving.

The most commonly provided simulation environments are urban (see chapter 5 – Results). They can vary from a short fixed size road track to large cities generated from real world maps with a variety of roads, buildings and traffic intersections. Less common are the off-

[3] https://www.unrealengine.com
[4] https://unity.com/
[5] Game asset - anything that can go into a game (environment, 3D objects, etc.). https://gamedevelopment.tutsplus.com/articles/how-to-fund-your-games-by-creating-and-selling-game-assets--cms-24380

road environments that provide driving options in forests, fields, mountains, underground mines and other places interesting for an off-road vehicle.

Actors

A simulation can have dynamic objects (actors) like humans and vehicles that have specific behaviors in the environment. Besides the main driving actor vehicle (aka the ego-vehicle), which is typically provided, some simulations allow adding new secondary actors. Similarly with the environments, it is not always trivial to add new actor assets into some existing simulation. So in this work, the availability of already included actors (humans, vehicles) is surveyed.

For all the provided environments it is important that the different objects (e.g. ground, vegetation, road, buildings, people, other vehicles, etc.) in the simulation can be automatically labeled for ML. This is described more closely in chapter 2.3 – Output Training Labels.

In addition to the environments, the actors and the output that the simulation produces, one more important part is introduced: the sensors which take input from the environment and provide data for the output. The next chapter introduces the main interesting sensors in the scope of this work.

2.2 Sensors

The main sensors which make up an autonomous vehicle's perception system, are cameras, lidars and radars [17]. In this work, additionally a thermal infrared (IR) camera is added into the scope for a better human detection in off-road environments. While there are other important sensors in autonomous vehicles, the selection is made to limit the list for the scope of this work.

Before getting to the actual simulations where the presence of those sensors is surveyed, an introduction to each sensor is made. It will give an understanding how to think of those sensors in the virtual world and briefly describes how they work.

Cameras

Cameras in simulations are described as sensors. There can be different ones. According to the image type they can provide, cameras in simulations can be either RGB, depth, thermal IR, semantic segmentation or some other type not limited to this list. It is important to note that unlike other cameras here, there is no equivalent for semantic segmentation camera in the real world. The images from those cameras provide unique training data for ML.

The RGB camera generates regular camera images from the surroundings. The depth camera generates ground-truth depth data where each pixel is marked with the distance from the camera. The thermal IR camera generates images where each pixel estimates the temperature of surrounding objects. The semantic segmentation camera generates images with each class (e.g. road, car, human) denoted by a different color (e.g. road=purple, car=blue, human=red). Example images from RGB, depth and semantic segmentation cameras are in Figure 1. As the semantic segmentation is also an important output label, it will be described more closely under a dedicated chapter (2.3 – Output Training Labels).

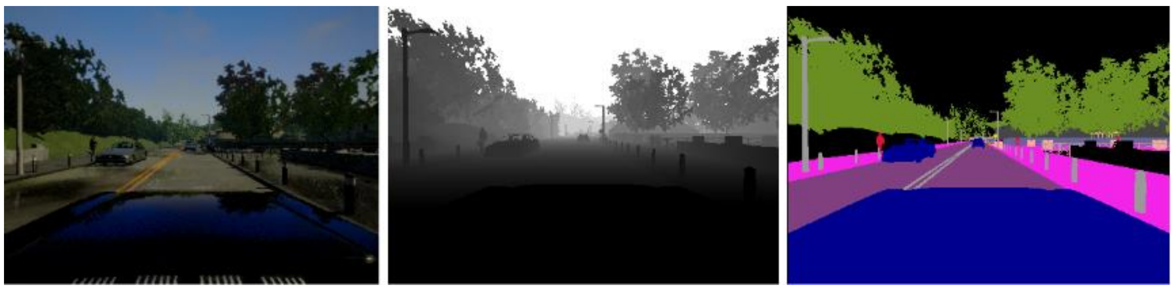
Figure 1. From the left: RGB, depth, semantic segmentation.[6]

Cameras in simulations usually have the following parameters which can be configured:

- image size (width, height) in pixels
- field of view (FOV) in degrees
- position and orientation relative to the vehicle

In this work, the surveyed cameras in the simulations are the RGB, depth and thermal IR camera.

Lidar

The lidar sensor in the real world measures distance to its target objects using laser beams. In a simulation it is implemented using ray casting[7] where each ray represents a real world laser beam (Figure 2). The sensor generates data which is a measurement of all the lidar sample point coordinates in 3D space at a given time. From lidar sample points it is possible to detect objects in 3D space. The more sample points, the better accuracy in detection as the shape of an object becomes more clear.

[6] https://carla.readthedocs.io/en/latest/cameras_and_sensors/
[7] https://en.wikipedia.org/wiki/Ray_casting

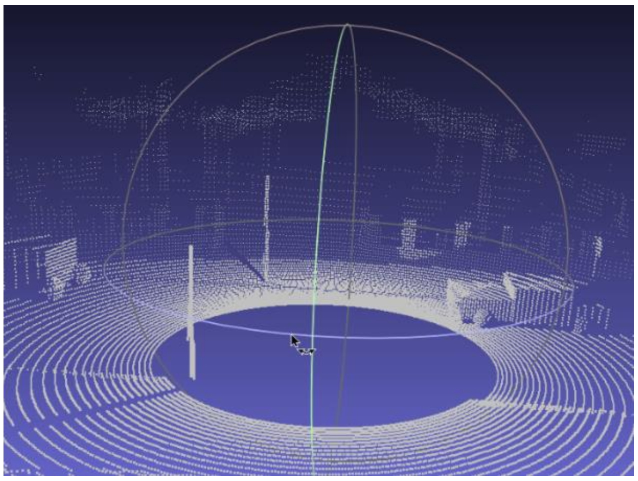
Figure 2. Example image of a lidar point cloud.[8]

The configurable attributes of a typical lidar sensor are:

- number of lasers
- range
- number of points generated per second
- rotation frequency
- FOV (horizontal and vertical) in degrees
- position and orientation relative to the vehicle

In the real world, lidar sensor cannot ’see’ through objects. It is an optical sensor and obstacles like grass, tree leaves, fog and rain limit its view. If the sensor is implemented in a realistic way in the simulation, it will not penetrate objects. For this, an electromagnetic sensor type is needed, such as radar.

Radar

The radar sensor in the real world measures usually the speed and distance of target objects in its field of view. Differently from the lidar, which is an optical sensor, radar uses electromagnetic waves to sense the surroundings. In a simulation (Figure 3), there are several ways to implement radar [20]. A simplified overview of two ways are given. One is that the radio wave is created as a cone-shaped object (starting from the sensor) and the measurement points are recorded in 3D space where the cone intersects with other objects. The second is ray tracing[9], or ray casting as it is for lidar. Objects which can or cannot be penetrated by the wave, are defined in the engine [21]. If the simulated radar sensor is used

[8] https://carla.readthedocs.io/en/latest/cameras_and_sensors/
[9] https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

to measure also the speed of an object, it can be calculated from several measurements in a row, using the closest points [20].

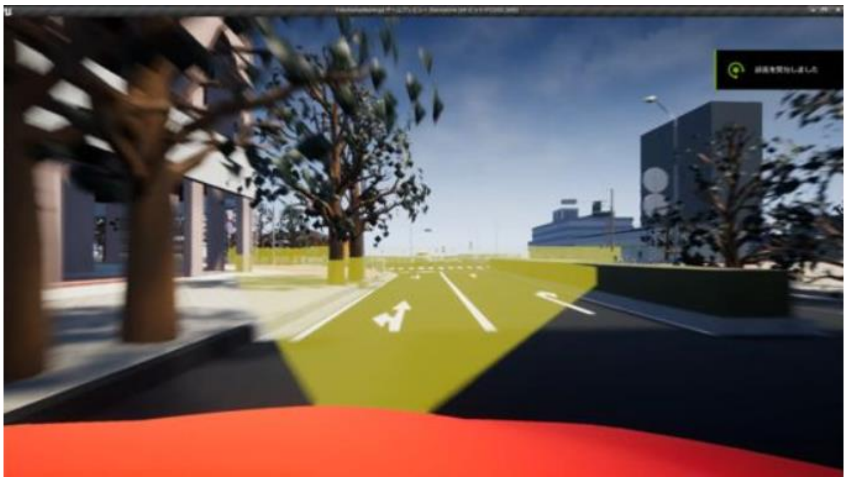
Figure 3. Simulated radar view visualization (in yellow color).[10]

The configurable attributes of a typical radar sensor are based on its implementation example in Unity game engine [21]. These are:

- range
- FOV (horizontal and vertical) in degrees
- position and orientation relative to the vehicle

Both, the lidar and radar are used in autonomous vehicles and the simulations. Without going into further details if or which sensor should be chosen from those two, they both are important sensors in this survey.

2.3 Output Training Labels

The final important aspect is the output labels. The output training labels (or the *ground truth*) are what the simulation outputs as the training data for ML. The main purpose of a simulation is not to simply provide a large amount of data but to provide data which is already labeled. Labeling, when applied automatically, saves time and is more accurate than if done by a human. Easily extracted labeled data is one of the main value points why the simulations in the scope of this work are used. There are different types of labels. Here semantic segmentation and bounding boxes (2D and 3D) are described which all are included in the surveyed features.

[10] https://m.eet.com/content/images/eetimes/Fig2-View_from_simulator_1511161122.jpg

Semantic Segmentation

Semantic segmentation is a labeling type in ML where every pixel in an image is marked with information about to which object class it belongs (Figure 4).

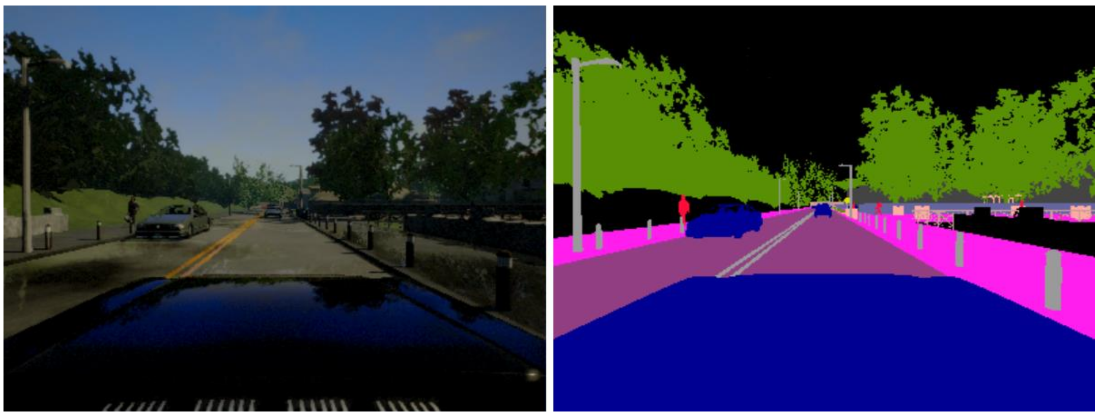
Figure 4. Semantic segmentation example.[11]

It is a useful and detailed method to classify data for ML because every label meaningfully (i.e. semantically) describes the corresponding object [8,9]. In a simulation, for example if a model needs to learn obstacles on the vehicle's path, the exact boundaries as well as the type of every obstacle is available with semantic segmentation. Also, considering the semantic class of an obstacle, different actions can be taken, e.g detecting a human may require a driving path modification but detecting a small rock may not.

Semantic segmentation of a 3D point cloud, e.g. from a lidar, is similar but the pixels are now points in a 3D space and each point is assigned a label as the type of the object it belongs to. In this work, only 2D semantic segmentation is considered in the simulations.

[11] https://carla.readthedocs.io/en/latest/cameras_and_sensors

Bounding Boxes

Bounding boxes in ML are labels that are used for marking the smallest 2D or 3D box surrounding an object (e.g. a human, vehicle, etc.). Examples of 2D and 3D bounding boxes are in Figure 5.

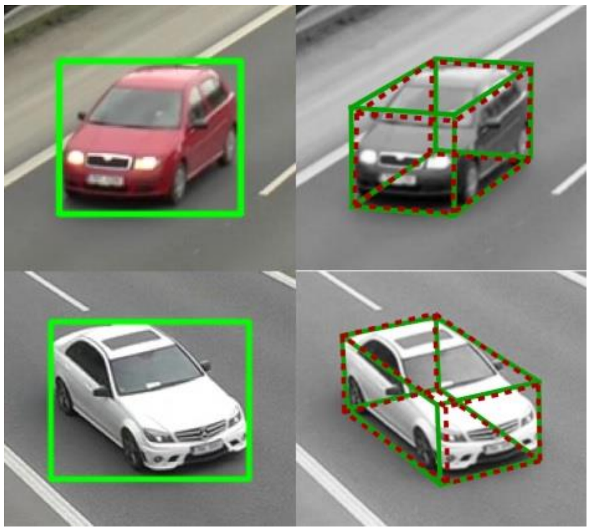

Figure 5. Bounding boxes 2D (left) and 3D (right).[12]

The box (axis-aligned) shows the maximum bounds of an object along each world axis. Bounding boxes are used for moving objects (e.g. humans, vehicles). This is mainly because it is a faster method than semantic segmentation but also because the bounding boxes can be used for tracking. Tracking the moving objects is useful for predicting their path. For example a vehicle can adjust its driving direction to avoid humans or other vehicles who are on a collision course with it. In combination, semantic segmentation and bounding boxes provide more information for learning about obstacles and to avoid them.

[12] https://medusa.fit.vutbr.cz/traffic/research-topics/fine-grained-vehicle-recognition/boxcars-improving-vehicle-fine-grained-recognition-using-3d-bounding-boxes-in-traffic-surveillance/

3 Methodology and Features

The target group of the survey consisted of simulations which are used for generating labeled data for training ML models for autonomous vehicles. The simulations were found online, via Google search and among GitHub repositories, using mainly (but not limited to) the following keywords in varying combinations: simulation, simulator, autonomous, vehicles, cars, machines, driving, machine learning, training, models, synthetic, artificial, data, labels, bounding box, semantic segmentation, off-road, environments.

From the results with videos, images, home pages, code repositories or dedicated documentation sites, the first sample was made from what seemed to be relevant to the targeted simulations, or looked more popular. Then the list was narrowed down, by a more thorough search about each simulation by using mainly its homepage, source code in GitHub and results from Google and YouTube.

Finally only those simulations were selected for this work which:

1. had documented or video reference about at least one of the surveyed sensors,
2. had documented or video reference about at least one of the surveyed environments,
3. had documented or video reference about at least one of the surveyed output training labels,
4. in case of GitHub repository, had activity within a few months from the time of the search,
5. were found to have news coverage or significant mentions since 2018 (especially from GTC[13] 2018),
6. and whose homepage information seemed not older than 2018.

The search was done in Q1 2019 and finally 12 simulations were selected out of the initial 39.

The features of the simulations for this work were chosen based on what was available in the found simulations and what is important for training ML models for off-road vehicles. The information what is important for ML models was gathered from the course Introduction to Computational Neuroscience in University of Tartu (UT), from cooperation with the UT Computational Neuroscience Lab and from the found research papers [17,3,5,4].

[13] https://www.nvidia.com/en-us/gtc/

As the documentation about the simulations varied in its presence and detail, missing information was requested via direct email or a provided online contact form. From those requests only one received an answer (from Craig Quiter, the creator of the simulation Deepdrive 2.0). The remaining missing information is marked as not available (NA).

The features interesting for training ML models from simulation data, were introduced in the previous chapter. Here the licence type and documentation availability are added which are useful when choosing a simulation for further validation or use.

The following features are finally taken into account for each simulation:

1. **Licence** – the licence type and source code availability (open source).
2. **Documentation** – online documentation availability for different usage steps:
    - Installation
    - Environment usage
    - Sensor configuration
    - Output labels usage.
3. **Engine** – the used engine.
4. **Driving environment** – included driving environments:
    - Urban
    - Off-road.
5. **Actors in environment** – included actor assets of:
    - Cars
    - Humans.
6. **Sensors** – availability (out of the box) of the surveyed sensors:
    - Cameras (RGB, depth, thermal / IR
    - Lidar
    - Radar.
7. **Output training labels** – availability (out of the box) of the surveyed output labels:
    - Semantic segmentation
    - 2D bounding box
    - 3D bounding box.

Licence types for the simulations are roughly divided to free and commercial (not free). Here a 'free' licence is a free-to-use licence (i.e. GPL, MIT) which is included with the simulation. It is important to note here that if an open source simulation has for example

MIT licence but is built on a game engine such as UE, then the simulation and its engine component (UE) are clearly separated in the terms of licencing. UE has its own licencing terms. Similarly, a simulation can have other components included which have their own licence. Commercial licences are usually not free-to-use and can have different terms/costs for users. Here they are simplified to one type ’proprietary’.

Based on those features the information is collected throughout the work and presented in chapter 5 – Results.

4 Simulations

Here the 12 selected simulations are listed. For each, a brief description of its general information, environments, sensors and output training labels is provided. Here, ‘featured’ means what is surveyed in the simulations (e.g. featured sensors are the cameras, lidar and radar that are listed in the previous chapter).

4.1 CARLA

CARLA[14] is an open source simulation for autonomous driving research (Figure 6). It is created by research scientists from Intel Labs, Toyota Research Institute and Computer Vision Center in Barcelona [2]. The simulation is built on the UE game engine. It was published in March 2018 and its latest version at the time of the survey was 0.9.5.

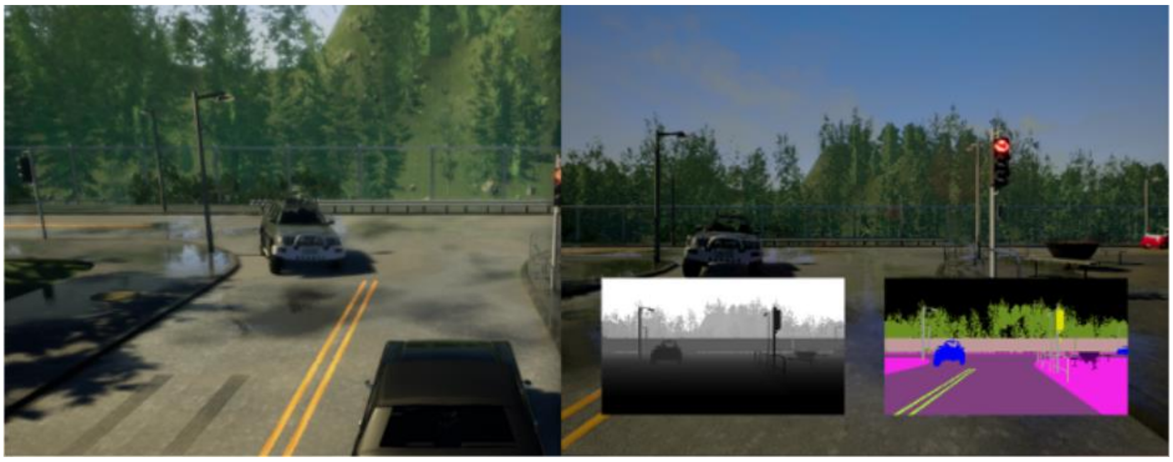

Figure 6. Example images from CARLA simulation.

Environments

For the driving environment, CARLA provides 7 ready made towns, from which one is rural (using different combinations of provided urban assets), and 15 different predefined weather conditions. It is possible to build new custom towns (with provided urban assets) and configure the weather. No off-road environment is provided. From the actor assets, humans and cars are provided.

Sensors

From the featured sensors the simulation provides cameras (RGB, depth) and lidar. No radar is provided. Additional sensors provided in CARLA, but not featured in the survey, are a collision detector, lane detector, obstacle sensors, accelerometer and GPS. In CARLA it is

[14] http://carla.org/

also possible to add other types of sensors through the API. This is a good customization option but it is not clear within what limits the other types of sensors can be added (e.g. is the limitation within the used game engine or are there any additional constraints). The configurable parameters for sensors are what is typically available for the cameras and lidar (see chapter 2). More exact details about their sensor configurations are available on their documentation site[15].

Output Training Labels

The simulation provides 12 semantic segmentation classes (Unlabeled, Building, Fence, Other, Pedestrian, Pole, Road line, Road, Sidewalk, Vegetation, Car, Wall, Traffic sign) and both 2D and 3D bounding boxes for all the dynamic objects in the simulation (Figure 7).

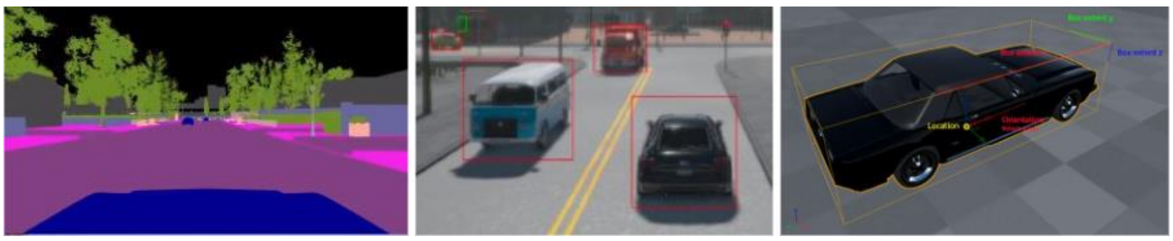

Figure 7. From the left: semantic segmentation, 2D bounding boxes, 3D bounding boxes in CARLA.

Additionally, CARLA provides information about the states of the traffic lights and the speed limit at the current location of the vehicle.

4.2 AirSim

AirSim[16] is an open source simulation for autonomous systems, including aerial and ground vehicles. It is developed by Microsoft for AI research in computer vision and reinforcement learning. Its first documented release was published in September 2017 and the latest version at the time of the survey was 1.2.0. For driving environments, their newer pre-release v1.2.1 is taken into account. The simulation is available on UE and Unity, but as the Unity version is still experimental at the time of writing, then only the UE version is surveyed. Example images from the simulation are in Figure 8.

[15] https://carla.readthedocs.io/en/latest/
[16] https://github.com/Microsoft/AirSim

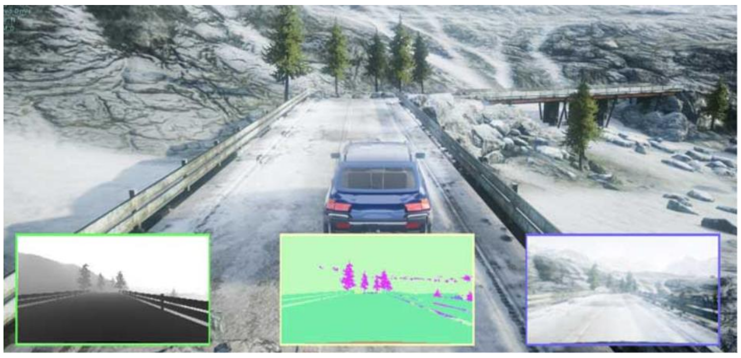

Figure 8. Example images from AirSim simulation.

Environments

AirSim is developed as a UE plugin so it can be easily added to an UE environment. This is useful if custom driving environments are needed. AirSim has 8 different weather options and 9 different driving environments (small urban neighbourhood block, large city with moving vehicles and pedestrians, Zhangjiajie mountains in China, Redwood forest, dynamic forest environment for camera detection of animals, plains with windmill farm, short coastline segment similar to Maui's Hana road, Africa terrain and animated animals, mountain landscape with power lines for drone camera detection). From these environments 5 are off-road. While it is rare to have off-road environments available, AirSim was initially made for aerial vehicles and does not provide large variety of objects for ground vehicles there. This constraint can be overcome by importing a new environment from a 3rd party or creating a new custom environment. The latter option can be time consuming and complex process though, if a user has no prior experience in environment design. From the actor assets, cars are provided. No human assets are provided.

Sensors

From the featured sensors AirSim provides cameras (RGB, depth, thermal IR) and a lidar. No radar is provided. Additional non-featured sensors in AirSim are barometer, magnetometer, IMU, GPS and distance sensors. The configuration options are comparable to what is typically available for the cameras and lidar (see chapter 2). More exact details about their sensor configurations are available on their documentation site.

Output Training Labels

The simulation provides semantic segmentation and bounding boxes (2D and 3D). For semantic segmentation (Figure 9), a segmentation class is provided for every mesh object (in the range of 0-255)[17] in the simulation. Using their API, existing segmentation can be modified to set desired semantic information (within provided range).

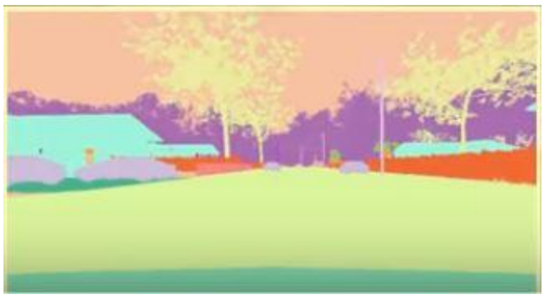
Figure 9. Example image of the semantic segmentation output label in AirSim.

For bounding boxes, AirSim does not provide API tools. While it is a significant constraint, it can be overcome using UE-s own tools to get 2D and 3D bounding boxes of desired objects.

4.3 Deepdrive 2.0

Deepdrive 2.0[18] is an open source simulation for experimenting with reinforcement learning of autonomous driving (Figure 10). It is built on UE and supports an external Python plugin[19] for simplified customization in UE. The simulation is developed under the initiative of Craig Quiter and should not be confused with its earlier version which was a different product but named similarly Deepdrive (without 2.0). In addition to the online documentation, the following includes also information received directly over email.

Figure 10. Example image from Deepdrive 2.0 simulation.

[17] https://microsoft.github.io/AirSim/docs/image_apis/#segmentation
[18] https://deepdrive.io/
[19] https://github.com/20tab/UnrealEnginePython

Environments

The simulation comes with one 3 km track for road driving. No other urban environments are included, neither any off-road. From the actor assets, cars are provided. No human assets are provided.

Sensors

From the featured sensors, cameras (RGB, depth) are provided. From other non-featured cameras there are five additional ones (reflectivity, world normals, ambient occlusion, base color, roughness – see Figure 11). The latter ones are not specific to this simulation but they are the typical camera views provided in UE.

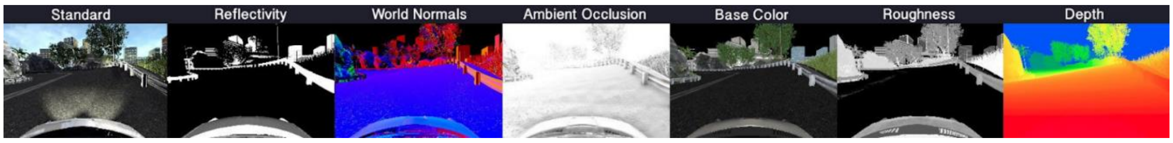

Figure 11. Example camera views from Deepdrive 2.0.

Lidar and radar sensors are not provided. From other non-featured sensors, many different ones are available, some of them vehicle and racing specific (e.g. IMU, brake, handbrake, distance, lap number, collision, position, steering, throttle). Additional information (within the limitation of the UE-s own possibilities) can be accessed using the external Python plugin.

Output Training Labels

Deepdrive 2.0 provides 3D bounding boxes for the cars in the simulation. Other training labels are not provided by the simulation itself. This constraint can be overcome using the Python plugin to extract the label information directly from UE.

4.4 LGSVL Simulator

The LGSVL Simulator[20] is a simulation software designed for developing and testing autonomous vehicles (Figure 12). It is built on Unity and includes a Python API. Created by the LG Electronics Silicon Valley Lab, its first release[21] came in December 2018 and the latest version at the time of the survey was 2019.03.

[20] https://www.lgsvlsimulator.com/
[21] https://github.com/lgsvl/simulator

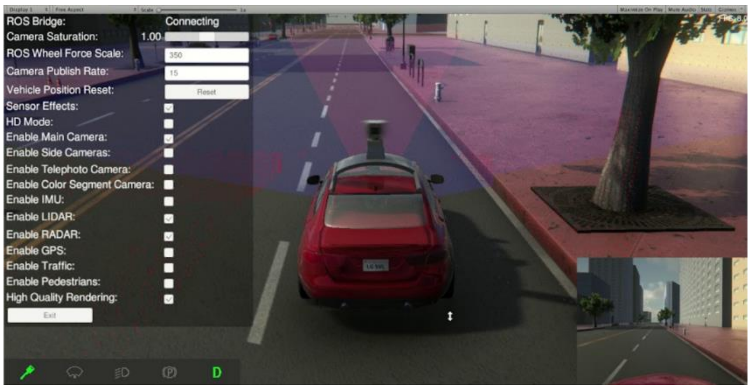
ROS Bridge:	Connecting
Camera Saturation:	1.00
ROS Wheel Force Scale:	350
Camera Publish Rate:	15
Vehicle Position Reset:	Reset
Sensor Effects:	✓
HD Mode:	☐
Enable Main Camera:	✓
Enable Side Cameras:	☐
Enable Telephoto Camera:	☐
Enable Color Segment Camera:	☐
Enable IMU:	☐
Enable LIDAR:	✓
Enable RADAR:	✓
Enable GPS:	☐
Enable Traffic:	☐
Enable Pedestrians:	☐
High Quality Rendering:	✓

Figure 12. Example image from LGSVL simulation.

Environments

By default the simulation provides one urban driving map and options to change the map to a different urban one. The map can be changed inside Unity. Also the parameters for weather, traffic and pedestrians can be changed. No off-road environment is provided. From the actor assets, humans and cars are provided.

Sensors

From the featured sensors, the simulation provides cameras (RGB, depth), lidar and radar. Additionally available non-featured sensors are GPS and IMU.

Output Training Labels

The simulation provides semantic segmentation, 2D and 3D bounding boxes (Figure 13) for the vehicles, pedestrians and unknown objects.

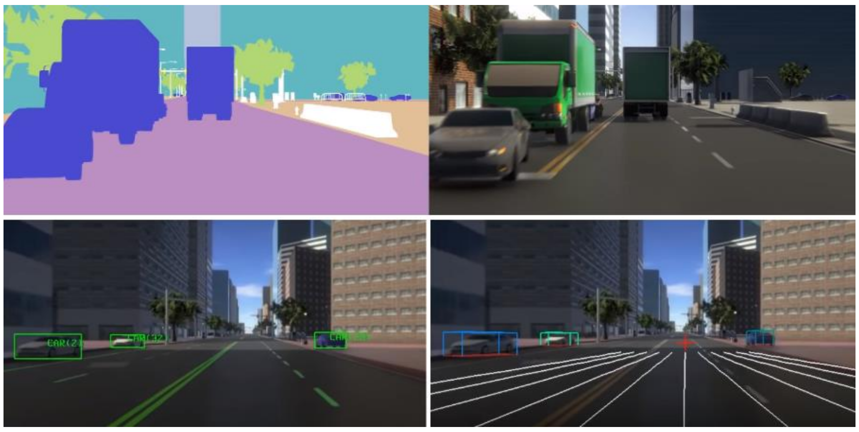

Figure 13. Example images[22] of different output labels from LGSVL simulation. Upper left: semantic segmentation (of the scene on its right), bottom left: 2D bounding boxes, bottom right: 3D bounding boxes.

[22] https://www.youtube.com/watch?v=NgW1P75wiuA

4.5 Sim4CV

Sim4CV[23] is a simulation for computer vision researchers. It is developed by King Abdullah University of Science and Technology. The simulation (Figure 14) is built on UE and its first release was published in 2017 as open source. The latest version at the time of the survey is not open source and its exact version remains unknown as no public information was found.

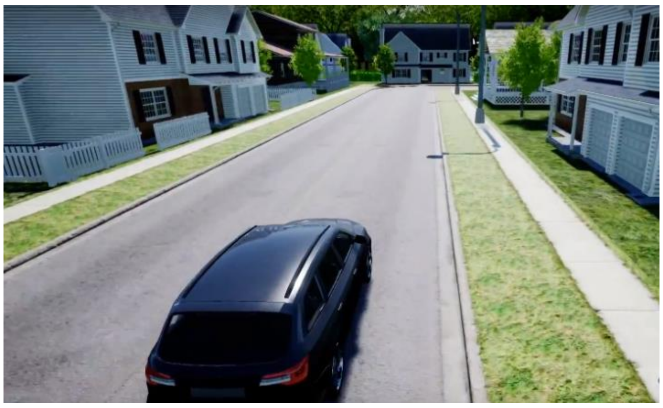
Figure 14. Example image from Sim4CV simulation.

Environments

From their video[24] and the related research papers [22,15] the simulation has both urban and off-road environments. It has one small town and for off-road it has one grassy field with trees and one desert environment (Figure 15).

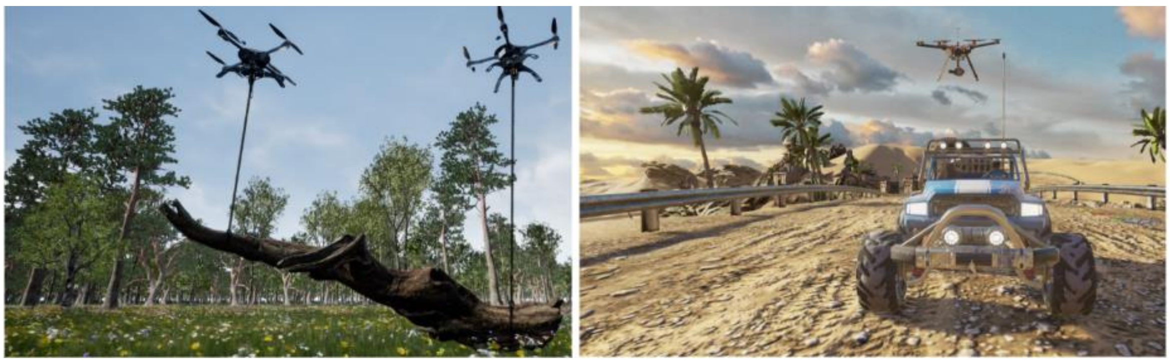
Figure 15. Examples of off-road environments in Sim4CV.

In their work [15] the creators describe adding environments easily with ’drag and drop’. This seemingly simple option for adding new off-road environments is certainly interesting for the scope of this survey but as no more information is provided, the final confirmation

[23] https://sim4cv.org/
[24] https://www.youtube.com/watch?v=SqAxzsQ7qUU

has to be done by evaluating their product. From the actor assets, humans and cars both are provided.

Sensors

From the featured sensors only the RGB and depth cameras are documented. The remaining featured sensors (thermal IR camera, lidar, radar) are not provided.

Output Training Labels

The simulation provides semantic segmentation, 2D and 3D bounding boxes. The 3D bounding box example was not found on any of their available outdoor environment images but the other two labels in Sim4CV are shown in Figure 16.

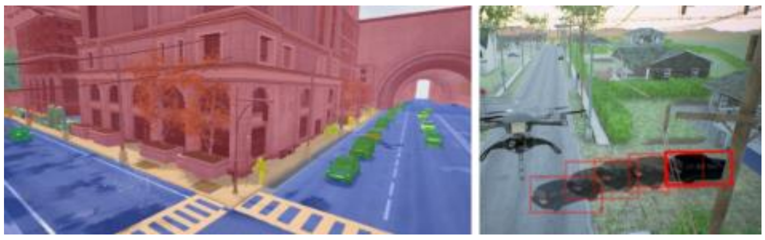
Figure 16. Examples of the output labels. From the left: semantic segmentation, 2D bounding boxes.

Having all three featured labels available in addition to off-road environments makes it very interesting product for further validation.

4.6 SynCity

SynCity[25] is a commercial simulation for autonomous vehicles (including ground, air and underwater). The simulation is built on the Unity game engine. It is developed by CVEDIA, a private company based in Netherlands. They focus on providing custom need-based environments and highly configurable sensors for their customers. The simulation is not available online for free. Some examples from the simulation are in Figure 17.

[25] https://syncity.com/

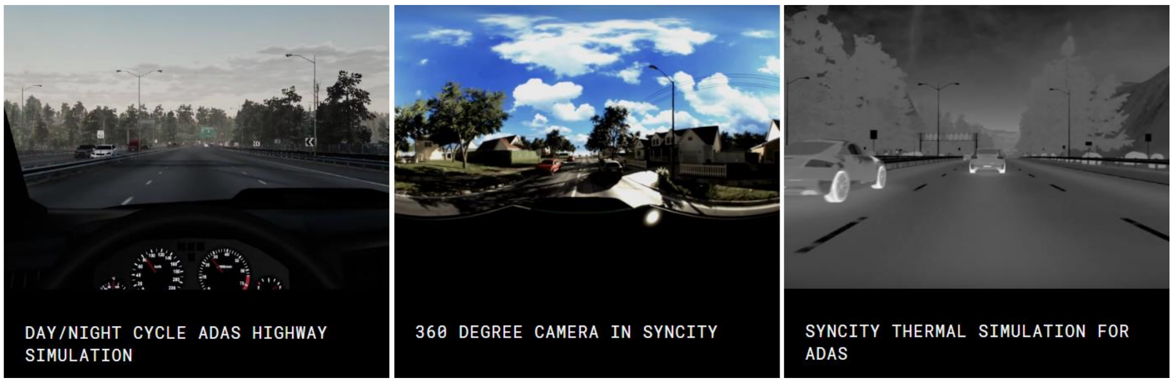
DAY/NIGHT CYCLE ADAS HIGHWAY SIMULATION
360 DEGREE CAMERA IN SYNCITY
SYNCITY THERMAL SIMULATION FOR ADAS

Figure 17. Example images from SynCity simulation.

Environments

The environments are provided together with the simulation according to the customer needs. Both urban and off-road environments are possible. From the examples on their homepage, it is possible to choose from at least the following off-road environments: desert, forestry and agricultural lands, savannah, marine (for ships/rescue), underground mines and aerial (for drones). See Figure 18 for examples of available off-road environments:

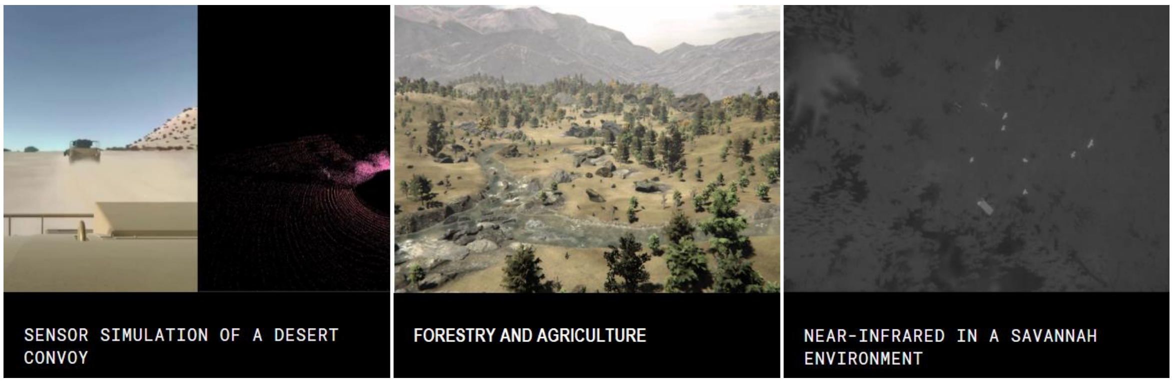
SENSOR SIMULATION OF A DESERT CONVOY
FORESTRY AND AGRICULTURE
NEAR-INFRARED IN A SAVANNAH ENVIRONMENT

Figure 18. Examples from SynCity off-road environments. From the left: desert, forest, savannah.

From the featured actor assets, humans and cars are available.

Sensors

From the featured sensors, all cameras (RGB, depth, thermal IR), lidar and radar are provided. It is possible to add also custom sensors to the simulation and have every attribute of a sensor configurable (within the limits provided by Unity).

Output Training Labels

All the featured labels are provided: semantic segmentation, 2D and 3D bounding boxes. The output data is not limited to SynCity documentation and can be expanded using Unity's own possibilities.

4.7 Unikie

Unikie Simulator[26] is a commercial simulation (Figure 19) mainly focusing on lidar based autonomous systems, including cars, ships and underground mine special vehicles. It is developed by Unikie, a private software company in Finland who provides tools also for collecting and processing data from the simulation for machine learning. The used engine remains unknown. The simulation itself and its documentation is not available for free.

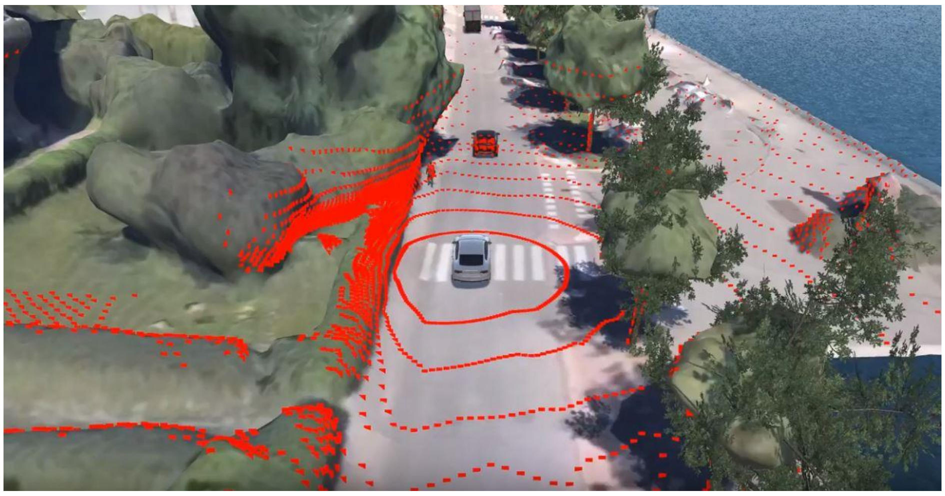
Figure 19. Example image from Unikie simulation.

Environments

From their homepage, Unikie provides both urban and off-road environments. From off-road, an underground mine and harbor are available. Having an underground mine is a rare environment to offer and in this survey it is considered as an off-road driving option. This unique environment is the main reason why this simulation is included in this work. The harbor environment, even if not in the scope of this work, shows the variety in possible environments and slightly distinguishes Unikie from many other surveyed simulations. In addition, in Unikie it is possible to add custom environments. From the actor assets, humans and cars are available.

Sensors

From the featured sensors only RGB cameras and lidar are found to be available. From the non-featured sensors, IMU and GPS are mentioned on their homepage. Information about other sensors remains unknown.

[26] https://www.unikie.com/en/solutions/unikie-simulator/

Output Training Labels

According to their homepage, Unikie simulation provides automatic labeling for machine learning data, but no exact information was found about what kind of labeling is provided or what are the label classes. So the information about semantic segmentation and bounding boxes remains unknown. This makes Unikie different from all other surveyed simulations here as it was one of the requirements in chapter 3 to have at least one of the featured output label available.

The reason why Unikie was exceptionally included in the survey, was mainly because of its uniqueness in environments and written statement on their homepage that they do provide labeled data from the simulation.

4.8 rFpro

RFpro[27] is a commercial simulation (Figure 20) for developing and testing autonomous vehicles, including ADAS (Advanced Driving Assistance Systems) and vehicle dynamics. It is developed by rFpro, a private company in UK who makes high fidelity simulations for professional motorsports and other road-only vehicles. The used engine for the simulation remains unknown. The simulation and the documentation are not available online for free.

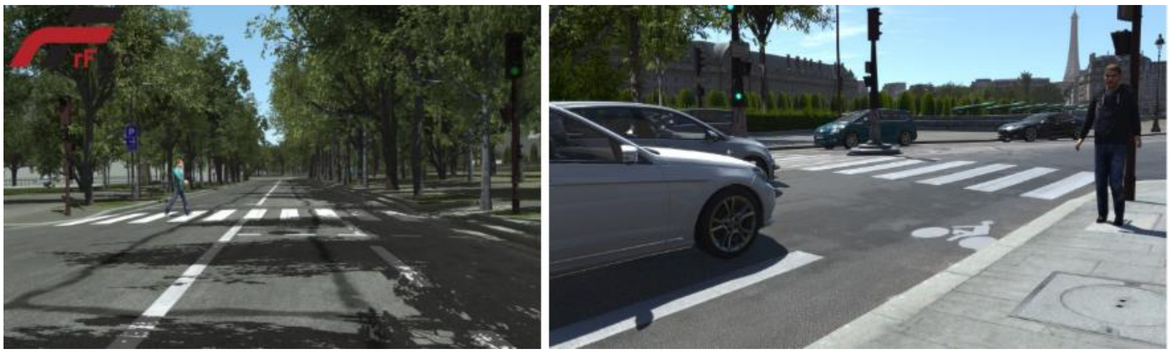

Figure 20. Example images from rFpro simulation.

Environments

RFpro provides physically accurate road surface environments (Figure 21), including motorsport tracks. The environments are created from real world HD maps and lidar scanning.

[27] http://www.rfpro.com/driving-simulation/

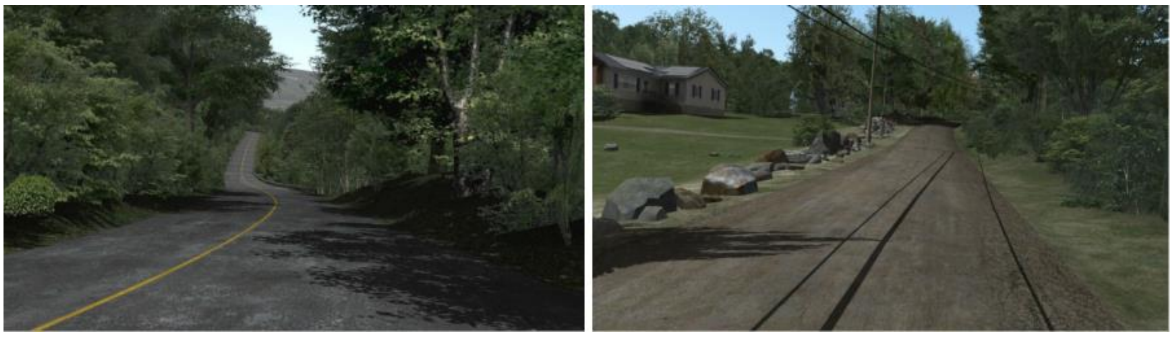
Figure 21. Example images from rFpro environments with road varieties.

Their idea is to copy the real world road surface and near surroundings (including atmosphere and weather) as physically closely as possible. No off-road environments are provided. From the actor assets, humans and cars are provided.

Sensors

From the featured sensors, the simulation provides cameras (RGB, depth), lidar and radar. No information was found about the thermal IR camera or any other non-featured sensors.

Output Training Labels

According to their homepage the simulation provides labels for training supervised learning models, but only semantic segmentation (Figure 22) was mentioned. No information about bounding boxes (2D or 3D) was found.

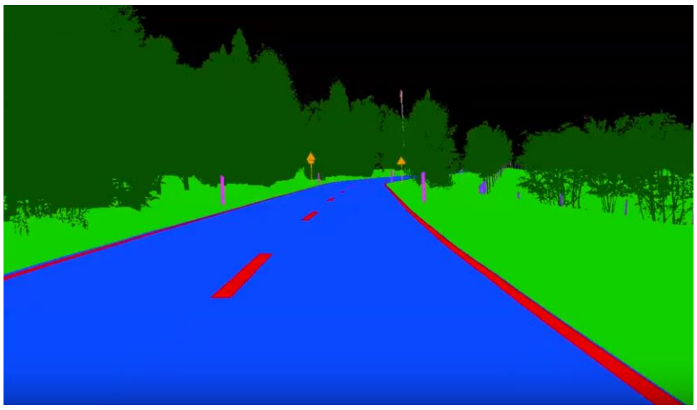
Figure 22. Example of semantic segmentation in rFpro[28].

During this survey, the exact semantic segmentation classes were not found to be available.

[28] https://www.youtube.com/watch?v=r2hJZu7qLO0

4.9 Cognata

Cognata[29] is a simulation (Figure 23) for autonomous vehicles and ADAS. It is developed by Cognata - a US based Israeli startup, founded in 2016. The simulation uses a custom engine and provides automatic urban environment generation (from provided urban assets). The simulation software and its documentation are not available online for free.

Figure 23. Example image from Cognata simulation.

Environments

The simulation provides a city environment with urban assets and automatic urban environment generation. No off-road environments are provided, neither any information about possibilities to add other custom environments in the simulation. From the actor assets, both humans and cars are available.

Sensors

From the featured sensors, RGB camera, lidar and radar are provided. No information about the depth or thermal IR cameras is found.

Output Training Labels

The simulation provides semantic segmentation (Figure 24), including the classes: cars, signs, buildings, humans, vegetation and electric pole.

[29] https://www.cognata.com/

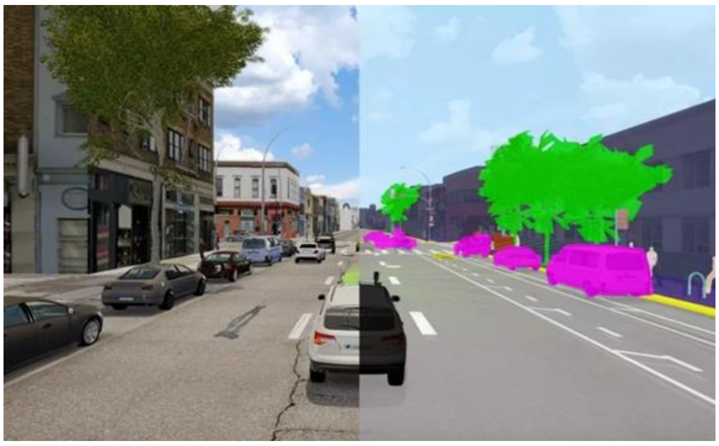
Figure 24. Semantic segmentation from Cognata simulation.

Regarding 2D and 3D bounding boxes, no information is available whether the simulation provides them.

4.10 SCANeR Studio

SCANeR Studio[30] is a commercial simulation (Figure 25) for autonomous vehicles. It is part of their simulation platform which includes testing and driving for ADAS, Human Machine Interface (HMI), headlight testing and vehicle dynamics. It is the product of a private French company AVSimulation and targets professional automotive industry customers. As a toolset for professionals, it also enables integration with driving simulators and dedicated hardware. The used simulation engine remains unknown. SCANeR Studio simulation software and documentation are not available online for free.

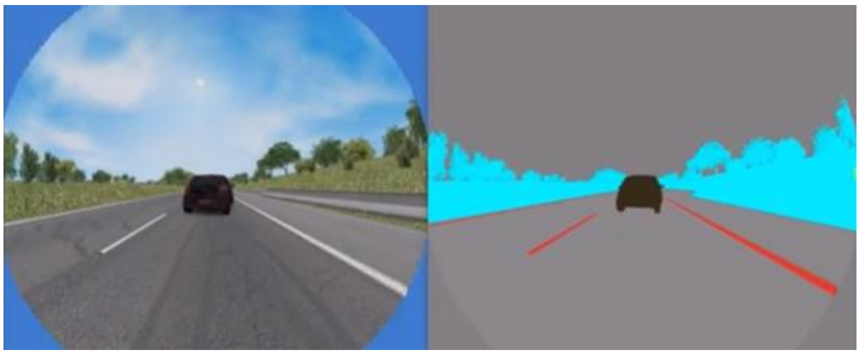
Figure 25. Example images from SCANeR Studio simulation, with semantic segmentation on the right.

Environments

The provided driving environments are urban only: a city and road (highway). It is possible to create new advanced environments, including the road, weather conditions, sensors,

[30] https://www.avsimulation.fr/solutions/

traffic, vehicle dynamics and drivers. No off-road environments are provided though. From the actor assets, cars are provided. About human assets, no information was available.

Sensors

From the featured sensors, RGB camera, lidar and radar are provided. Information about the depth and thermal IR cameras remains unknown. Also no additional non-featured sensors were brought out.

Output Training Labels

SCANeR Studio provides semantic segmentation (Figure 25, on the right) but no information about exact classes was found. Also information about bounding boxes (2D or 3D) remains unknown.

4.11 Highwai

Highwai[31] is a commercial simulation (Figure 26) focusing on training neural networks for autonomous vehicles. It is developed by Highwai, a private company in US who, starting from 2016, makes AI toolsets for object detection (including dataset creation and manipulation). Highwai simulation is built on Unity.

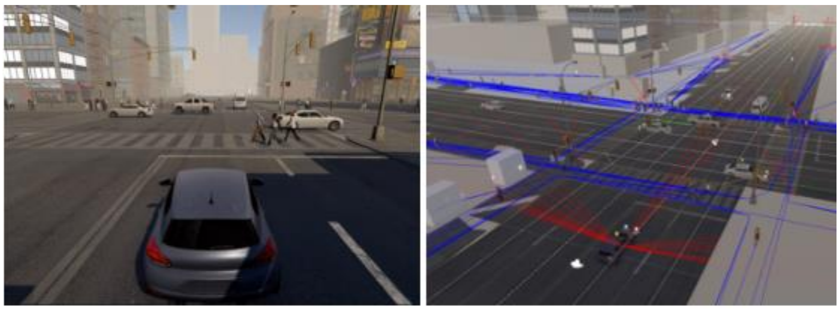
Figure 26. Example images from Highwai simulation.

Environments

The simulation provides urban environment (city) with customization options in scenes and scenarios. No off-road environment is provided. From the actor assets, both humans and cars are available.

Sensors

From the featured sensors, cameras (RGB, depth, thermal IR) and lidar are provided. No information about radar sensor is provided.

[31] https://www.highwai.com

Output Training Labels

The simulation provides bounding boxes (2D, 3D) and object segmentation (Figure 27). Semantic segmentation is not mentioned in their provided materials. From their video content[32], object classes such as vehicles, buildings, roads, pedestrians, street props (no information what this means), road markings and sidewalk are provided.

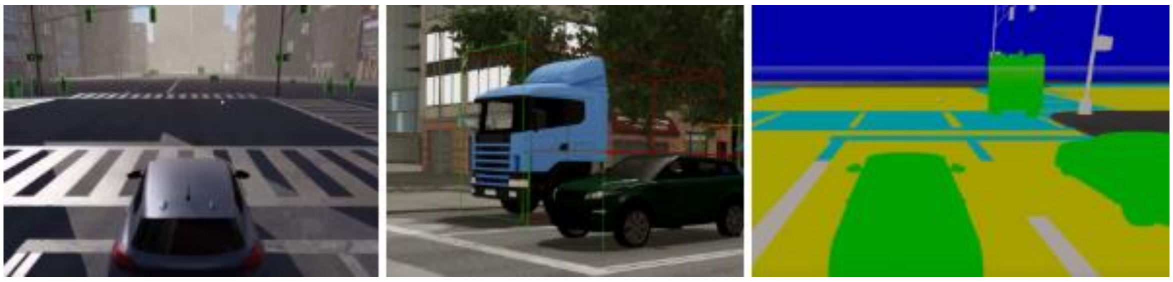

Figure 27. Output labels from the left: 2D bounding boxes, 3D bounding boxes and object segmentation.

No documentation is provided if the object segmentation classes can be customized.

4.12 NVIDIA Drive

NVIDIA Drive[33] simulation software is part of NVIDIA DRIVE Constellation, a simulation platform for developing autonomous vehicles which consists of hardware and software. The simulation (Figure 28) is available together with the DRIVE SDK to Nvidia's partners (e.g. developers and autonomous systems providers). From how it looks from one[34] of their videos[35], the simulation is built on UE. For the wider public, access to the simulation is restricted.

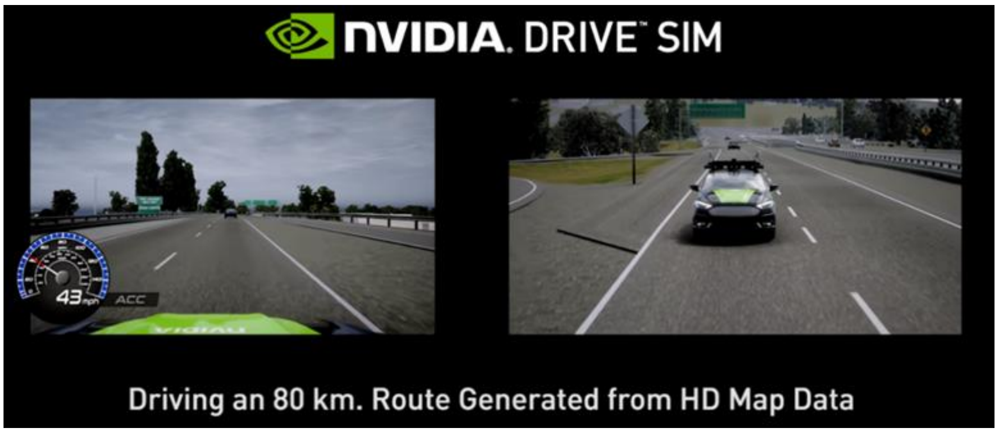

Figure 28. Example images of NVIDIA Drive simulation.

[32] https://www.youtube.com/watch?v=7i_xa9pYokY
[33] https://developer.nvidia.com/drive
[34] https://www.youtube.com/watch?v=DXsLDyiONV4 (NVIDIA Drive simulation in January 2019)
[35] https://www.youtube.com/watch?v=lVlqggTiTzY (NVIDIA Drive simulation introduction at GTC 2018)

Environments

Environments in NVIDIA Drive simulation are urban only. Mainly roads (highways) and some city environments are noted in their videos. The simulated environments are created from (real world) HD map data and are customizable. About the included actor assets no information is provided.

Sensors

From the featured sensors, the simulation provides RGB cameras, lidar and radar. No information was found about the depth and thermal IR sensors. Additional provided non-featured sensor is the IMU. There is an option to add custom sensors but no further information is provided about which sensors exactly.

Output Training Labels

The provided training labels are semantic segmentation and 2D bounding boxes. No information about the availability of the 3D bounding boxes was found.

5 Results

From the surveyed simulations and their features all the gathered information is in the four tables below. The first table (**Table 1**) shows the more general information about the simulation, including the licence type, availability of documentation and the used engine. The second table (**Table 2**) shows information about provided driving environments and what kind of actor assets they include. The third table (**Table 3**) shows the sensors. And finally the fourth table (**Table 4**) is for the output training labels. The cells marked with NA are unknown, meaning the information was not found to be available online and the requests for information sent over email were not answered. After each table the results are described.

Table 1. The information about licencing, documentation and the used engine.

		Documentation				
	Licence	Installation	Environments	Sensor	Output	Engine
CARLA	MIT	+	+	+	+	UE
AirSim	MIT	+	+	+	+	UE
Deepdrive	MIT	+	-	-	-	UE
LGSVL Simulator	Propr	+	+	+	+	Unity
Sim4CV	Propr	+	-	-	-	UE
SynCity	Propr	NA[36]	NA	NA	NA	Unity
Unikie	Propr	NA	NA	NA	NA	NA
rFpro	Propr	NA	NA	NA	NA	NA
Cognata	Propr	NA	NA	NA	NA	Custom
SCANeR Studio	Propr	NA	NA	NA	NA	NA
Highwai	Propr	NA	NA	NA	NA	Unity
NVIDIA Drive	Propr	NA	NA	NA	NA	UE

Table legend:
Licence cell colors: blue – open source, red – not open (commercial).
Other cell colors: green – available, red – not provided, grey – unknown.

[36] SynCity documentation site docs.syncity.com was taken down in April 2019 and since then no documentation about their current product is available online. The GitHub repository https://github.com/Cvedia/syncity-redist provides installation intructions to an SDK from 2018 but its relevance to their present product is unknown.

In overall the results show that 4 of the surveyed 12 simulations are open source (33%) from which 3 have a free-to-use licence (25%). All the open source solutions are documented well enough online to get a fairly good understanding about how to install and start using them. The commercial solutions are expected to provide the documentation to their customers but here almost no information was found online or received over email. Therefore validating a commercial simulation before deciding its suitability for purchase can take a significant amount of time and effort. Especially considering that there are many commercial simulation providers available in the field.

Regarding the used engines, the information was available in 9 cases out of 12 (75%). The UE game engine is noted in total 5 cases (42%) and then Unity in 3 cases (25%). Only in one case a custom built engine is noted (Cognata). In overall, the UE game engine seems to be currently the most popular choice for modern simulations found for this work.

Table 2. The information about the environments and included actor assets.

	Environment			
	For Driving		Actors	
	Urban	Off-road	Humans	Cars
CARLA	T	-	+	+
AirSim	T C	F M G	-	+
Deepdrive	R	-	-	+
LGSVL Simulator	C	-	+	+
Sim4CV	T	D G	+	+
SynCity	T C H	D F U H G	+	+
Unikie	C	U H	+	+
rFpro	T C R	-	+	+
Cognata	C	-	+	+
SCANeR Studio	C R	-	NA	+
Highwai	C	-	+	+
NVIDIA Drive	C H	-	NA	NA

Table legend:
Environment Urban: T - town, C - city, R - road track, H - highway
Environment Off-road: F - forest, D - desert, M - mountains, G - grassy field, U - underground mine, H – harbor

The results show that the main interest in this work to have the simulation being suitable out of the box for both urban and off-road vehicles, is satisfied only in 4 cases (33%). From those (AirSim, Sim4CV, SynCity, Unikie) only AirSim is open source and available for free. This is a surprising result that there is so few off-road options. In contrast, at least one urban environment is provided in every simulation.

Two unique environments were noted, besides the forest, mountains, desert and grassy plains. These two were the underground mine and harbor. Both provided by commercial providers (SynCity and Unikie).

The featured actor assets in simulations seem to be present in most cases. Usually this information tends to be not documented but available through alternative sources (i.e. videos and product presentations). Car assets (besides the main one) were found to be included in 11 cases out of 12 (92%) and human assets in 8 cases (67%). The information was not available for humans in 2 cases (17%) and for cars in 1 case (8%). The least available information was about NVIDIA Drive, which is surprising as they are a big provider on the market and have presented their simulation at large product shows since GTC 2018.

Table 3. The information about the sensors.

	Sensors				
	Cameras			Others	
	RGB	Depth	Thermal / IR	Lidar	Radar
CARLA	+	+	-	+	-
AirSim	+	+	+	+	-
Deepdrive	+	+	-	-	-
LGSVL Simulator	+	+	-	+	+
Sim4CV	+	+	-	-	-
SynCity	+	+	+	+	+
Unikie	+	-	-	+	-
rFpro	+	+	NA	+	+
Cognata	+	NA	NA	+	+
SCANeR Studio	+	NA	NA	+	+
Highwai	+	+	+	+	NA
NVIDIA Drive	+	NA	NA	+	+

Sensors are the most documented features throughout all the simulations. Starting from the cameras, besides the regular RGB one (which is available for all), the depth camera is provided in 8 cases out of 12 (67%) and the thermal IR camera in 3 cases (17%). No information was found in 3 cases for the depth camera and in 4 cases for the thermal IR camera (all commercial products). For the off-road vehicles the rare presence of the thermal

IR camera may be a significant constraint as other sensors do not provide enough information to detect humans in nature.

From the other sensors, lidar is the most provided sensor being present in 10 cases (83%). Radar is less common, noted in 6 cases (50%). For radar, only in one case (Highwai) the information was not found.

Table 4. The information about the output training labels.

	Output Training Labels		
	Semantic Segmentation	2D bounding box	3D bounding box
CARLA	+	+	+
AirSim	+	-*	-*
Deepdrive	-*	-*	+
LGSVL Simulator	+	+	+
Sim4CV	+	+	+
SynCity	+	+	+
Unikie	NA	NA	NA[37]
rFpro	+	NA	NA
Cognata	+	NA	NA
SCANeR Studio	+	NA	NA
Highwai	NA[38]	+	+
NVIDIA Drive	+	+	NA

*- Not provided by the simulation but can be extracted using the game engine's own tools.

Somewhat surprising is to see that even from the available documentation, not all simulations provide the featured output training labels. Still, most simulations provide at minimum one and for some it is at least mentioned in the documentation if it is a planned feature or what is the workaround options to get such information (e.g. retrieving the labels directly from the game engine).

From the output training labels, semantic segmentation is most often provided, being there in 9 cases (75%). For the bounding boxes the situation is less clear. 2D and 3D bounding

[37] Unikie homepage declares that in their simulation the data can be automatically labeled but the exact labels are not specified.
[38] Highwai provides object segmentation but no mentions noted about semantic segmentation.

boxes both in 6 cases (50%) indicate that it cannot be taken for granted that a simulation for training ML models has all the main labels out of the box. For the large amount of commercial cases (33-42% for the bounding boxes and 17% for semantic segmentation) the information remains unknown.

In case of common features across all the results, the urban environment and regular camera images are provided in all of the surveyed simulations. The information was not found only in case of commercial simulations. SynCity was the only commercial simulation for which all the featured environmens, sensors and output labels were available.

Finally, if one has to choose a simulation for further validation and off-road in mind, a free and open source one seems to be a good starting point. But the problem is that in this case there is no choice – none of the free ones include all the features. Only AirSim has off-road environments included and is free. Even though AirSim has its constraints (extracting important output training labels is only through the game engine's own tools), a fair amount of information is available about this simulation and it uses widely known game engine UE which means a potentially large support network for extracting missing labels using UE tools. AirSim has been a popular choice already for many users in the research field (during this work at least five published research papers encountered during this work used AirSim [5,6,16,17,19] and four more [13,14,15,18] mentioned it, all within 2017-2019). There are also some papers about CARLA, and their GitHub activity seems active, but there is no off-road environment. This is a significant constraint of CARLA.

In overall, the choices are limited and lack of information about most commecial products prevent making clear decisions about existing simulations. Which leads to a thought that perhaps developing a new free product with all the featured properties may be worth to evaluate further. Similarly, contributing to an existing open source solution could be of value.

5.1 Discussion

During the survey several thoughts and questions came up, some of those not fitting in the scope of this work (e.g. importance of high-fidelity, choosing the engine, simulation design and architecture, hardware requirements). Two of the closest ones to the scope of this work are listed here.

First, it became apparent that while the open source simulations are fairly well documented but almost none of the commercial products provide such detailed information online, then it is not really possible to compare the open source and commercial products in detail by simply surveying them. At least not without actually purchasing (or having some direct access to) the commercial products themselves. This more thorough evaluation and testing of some selected products is needed and could be done as a future work.

Secondly, before rushing into the process of creating a new simulation that would be suitable for both off-road and urban, one needs to re-evaluate the need for urban. Several seemingly high quality commercial simulations already exist for urban. If the main interest is on off-road then perhaps it is worth to consider contributing to an existing open source solution in order to improve it for off-road. And, if this is for some reason not possible (because of any existing design constraints etc.), then return to the idea of creating a new one, and then focus on off-road only.

6 Conclusion

This work guided the reader to the world of simulations and synthetic labeled data generation for training ML models for autonomous vehicles. The importance of labeled data is emphasized over the large quantity of unlabeled data. Briefly, the reason of this work was mentioned and the main questions introduced. Regarding vehicles, the focus was expanded from common urban-only to also off-road.

Different aspects of the simulations were described for better understanding of the features selected in the survey. Briefly, the simulation environments (including actors), sensors and output labels were introduced.

Then the methodology and features of this survey were clarified. The features of the simulations for this survey were: licensing, documentation, the used engine, included driving environments, sensors and the output training labels.

All together 12 simulations out of 39 were surveyed, each of the 12 holding a small descriptive chapter of its own, with found facts and images.

After the surveyed simulations the results were described and presented in a table form. The results were divided into four separate tables. **Table 1** – general information about the simulation, including licensing, documentation and the used engine. **Table 2** – provided driving environments with included actors. **Table 3** – provided sensors. **Table 4** – provided output training labels for ML. Also the amount of unknown information (not found to be available) was shown. The initial questions about simulations (see chapter 1) were answered and some noticed constraints and findings are brought out. Briefly, few thoughts which appeared during the process of this survey and led to the need for future work, were introduced.

In overall, this work is expected to fill the gap about simulations for ML and autonomous vehicles. It is a compact overview of currently (Q1 2019) provided solutions found online. There may be more simulations which were not found so the list is not absolute. But for choosing a simulation for further evaluation or to choose features for a new simulation, this work can be a useful starting point. For how long this information remains reasonably valid, is unclear, as the developing field of autonomous vehicles and ML applications seems to be changing rapidly today.

7 References

[1] Kalra N, Paddock M, Rand Corporation. (2016). Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?. https://www.rand.org/pubs/research_reports/RR1478.html.

[2] Dosovitskiy, A, Ros, G, Codevilla, F, López, A, & Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf.

[3] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, Alexey Dosovitskiy. 2018. End-to-end Driving via Conditional Imitation Learning. https://arxiv.org/abs/1710.02410.

[4] Stephan R. Richter, Zeeshan Hayder, Vladlen Koltun. (2017). Playing for Benchmarks. https://arxiv.org/abs/1709.07322.

[5] Shah, Dey, Lovett, Kapoor. (2017). AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. https://arxiv.org/abs/1705.05065.

[6] Elizabeth Bondi, Debadeepta Dey, Ashish Kapoor, Jim Piavis, Shital Shah, Fei Fang, Bistra Dilkina, Robert Hannaford. (2018). AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs. http://teamcore.usc.edu/papers/2018/bondi_camera_ready_airsim-w.pdf.

[7] Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, Alex Kendall. (2018). Learning to Drive from Simulation without Real World Labels. https://arxiv.org/abs/1812.03823.

[8] Seif. Semantic Segmentation with Deep Learning. https://towardsdatascience.com/semantic-segmentation-with-deep-learning-a-guide-and-code-e52fc8958823 (16.03.2019)

[9] Wang. Semantic Segmentation slides. http://www.cs.toronto.edu/~tingwuwang/semantic_segmentation.pdf (05.03.2019)

[10] Thomas Steil, Jurgen Roßmann. (2015). A Virtual Reality Testbed for Camera Simulation in Aerospace Applications. https://ieeexplore.ieee.org/document/7604564.

[11] Dean A. Pomerleau. (1988). ALVINN: An Autonomous Land Vehicle in a Neural Network. https://dl.acm.org/citation.cfm?id=89891.

[12] Zeyn Saigol, Alan Peters, Matthew Barton, Mark Taylor. (2018). Regulating and Accelerating Development of Highly Automated and Autonomous Vehicles Through Simulation and Modelling. https://s3-eu-west-1.amazonaws.com/media.ts.catapult/wp-content/uploads/2018/03/23113301/00299_AV-Simulation-Testing-Report.pdf.

[13] Viswanath P, Mody M, Nagori S, Jones J, Garud H. Virtual Simulation Platforms for Automated Driving: Key Care-About and Usage Model. Electronic Imaging. 2018 Jan 28;2018(17):1-6. https://doi.org/10.2352/ISSN.2470-1173.2018.17.AVM-164.

[14] Rosique F, Navarro PJ, Fernández C, Padilla A. A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research. Sensors. 2019 Jan;19(3):648. https://www.mdpi.com/1424-8220/19/3/648.

[15] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, Bernard Ghanem. Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications. 2017. https://arxiv.org/abs/1708.05869.

[16] Carrio A, Vemprala S, Ripoll A, Saripalli S, Campoy P. Drone detection using depth maps. In2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2018 Oct 1 (pp. 1034-1037). IEEE. https://arxiv.org/abs/1808.00259.

[17] Josephine Mertens. Generating Data to Train a Deep Neural Network End-To-End within a Simulated Environment. 2018. https://www.mi.fu-berlin.de/inf/groups/ag-ki/Theses/Completed-theses/Master_Diploma-theses/2018/Mertens/MA-Mertens.pdf.

[18] Li W, Pan C, Zhang R, Ren J, Ma Y, Fang J, Yan F, Geng Q, Huang X, Gong H, Xu W. AADS: Augmented autonomous driving simulation using data-driven algorithms. arXiv preprint arXiv:1901.07849. 2019 Jan 23. https://arxiv.org/abs/1901.07849.

[19] Jacques Valentin. Deep Reinforcement Learning for Autonomous Off-road Driving in Simulation. 2018. https://dspace.cvut.cz/handle/10467/77185.

[20] Kallin N. 2018. Sensor simulation: Is AGXUnity a viable platform for adding synthetic sensors. http://www.diva-portal.se/smash/get/diva2:1303744/FULLTEXT01.pdf.

[21] Matt Smith, Chico Queiroz. 2015. Unity 5.x Cookbook: Displaying a radar to indicate the relative locations of objects. https://subscription.packtpub.com/book/game_development/9781784391362/1/ch01lvl1sec20/displaying-a-radar-to-indicate-the-relative-locations-of-objects.

[22] Matthias Müller, Neil Smith, Bernard Ghanem. 2016. A Benchmark and Simulator for UAV Tracking. https://www.semanticscholar.org/paper/A-Benchmark-and-Simulator-for-UAV-Tracking-Mueller-Smith/27850781e39df9f750e05409b8072261124068e8

Appendix

I. Glossary

Autonomous Vehicle[39] (in this work: Fully autonomous vehicle) – a vehicle that is capable of sensing its environment and moving with no human input.

Machine Learning[40] (ML) – a computer science study field where a computer is "taught" through a mathematical model to do a task.

Ground Truth[41] – information provided by direct observation.

Labeled data[42] in ML – grouped data that is marked with one or more labels (meaningful tag for the group of data).

Manual labeling - drawing bounding boxes around humans and cars, marking different areas of the image as road, buildings, trees, sky, etc.

Automated Driving Systems[43] (ADS) – a term used in automotive industry to refer to certain amount of automated driving aid in vehicles.

Advanced Driver Assistance Systems[44] (ADAS) – a term used in automotive industry to refer to sensors, functions, technology suppliers used in vehicles to assist its driver with certain functions (e.g. parking assistance).

[39] https://en.wikipedia.org/wiki/Self-driving_car
[40] https://en.wikipedia.org/wiki/Machine_learning
[41] https://en.wikipedia.org/wiki/Ground_truth
[42] https://en.wikipedia.org/wiki/Labeled_data
[43] https://en.wikipedia.org/wiki/Automated_driving_system
[44] https://www.aaam.org/automated-driving-systems-ads-introduction-technology-vehicle-connectivity/

II. Table – All Results

		Documentation					Environment				Sensors					Output Training Labels		
							For Driving		Actors		Cameras			Others				
	Licence	Installation	Environments	Sensor config	Output labels	Engine	Urban	Off-road	Humans	Cars	RGB	Depth	Thermal / IR	Lidar	Radar	Semantic Segmentation	2D bbox	3D bbox
CARLA	MIT	+	+	+	+	UE	T	-	+	+	+	+	-	+	-	+	+	+
AirSim	MIT	+	+	+	+	UE	T C	F M G	-	+	+	+	+	+	-	+	-*	-*
Deepdrive	MIT	+	-	+	-	UE	R	-	-	+	+	+	-	-	-	-*	-*	+
LGSVL	Propr	+	+	+	+	Unity	C	-	+	+	+	+	-	+	+	+	+	+
Sim4CV	Propr	+	-	-	-	UE	T	D G	+	+	+	+	-	-	-	+	+	+
SynCity	Propr	NA[45]	NA	NA	NA	Unity	T C H	D F G U M	+	+	+	+	+	+	+	+	+	+
Unikie	Propr	NA	NA	NA	NA	NA	C	U H	+	+	+	-	-	+	-	NA	NA	NA[46]
rFpro	Propr	NA	NA	NA	NA	NA	T C R	-	+	+	+	+	NA	+	+	+	NA	NA
Cognata	Propr	NA	NA	NA	NA	Custo	C	-	+	+	+	NA	NA	+	+	+	NA	NA
SCANeR Studio	Propr	NA	NA	NA	NA	NA	C R	-	NA	+	+	NA	NA	+	+	+	NA	NA
Highwai	Propr	NA	NA	NA	NA	Unity	C	-	+	+	+	+	+	+	NA	NA[47]	+	+
NVIDIA Drive	Propr	NA	NA	NA	NA	UE	C H	-	NA	NA	+	NA	NA	+	+	+	+	NA

Figure 29.

Table legend - Urban and Off-road: Urban: T - town, C - city, R - road track, H - highway. Off-road: F - forest, D - desert, M - mountains, G - grassy field, U - underground mine, H - harbor

Licence: blue – open source, red – not open (commercial)

* - Not provided by the simulation but can be extracted using the game engine's own tools.

[45] SynCity documentation site docs.syncity.com was taken down in April 2019 and since then no documentation about their current product was found online. The GitHub repository https://github.com/Cvedia/syncity-redist provides installation intructions to an SDK from 2018 but its relevance to their present product is unknown.

[46] Unikie homepage declares that in their simulation the data can be automatically labeled but the exact labels are not specified.

[47] Highwai provides object segmentation but no mentions noted about semantic segmentation.

III. Initial List of Simulations

Here is the initial sample of 39 seemingly available simulations found online. The list is in no particular order.

1. Apollo
2. Autoware
3. ANVEL
4. Deepdrive 2.0
5. NVIDIA Drive
6. Waymo
7. LGSVL Simulator
8. Aurora
9. Wayve
10. Highwai
11. Unikie
12. Zoox
13. Siemens TASS PreScan
14. VIRES
15. Righthook
16. Truevision.ai
17. SYNTHIA Dataset
18. BI sim
19. VDrift
20. TORCS
21. AutonoVi
22. Aorta
23. Dash
24. Sim4CV
25. CARLA
26. AirSim
27. The CAT vehicle testbed
28. Holodeck
29. Gym-UnrealCV
30. SynCity
31. Cognata
32. Parallel Domain
33. ANSYS VREXPERIENCE
34. SCANeR Studio
35. rFpro
36. Gazebo
37. V-Rep
38. ARGoS
39. Udacity self-driving car sim

IV. Licence