Virtual urban traffic infrastructure for testing highly automated mobility systems

Rene Degen,
CAD CAM Center Cologne (4C) at Cologne University of Applied Sciences, Cologne, Germany.
Division of Electricity, Department of Electrical Engineering, Uppsala University, Uppsala, Sweden.

Harry Ott,
CAD CAM Center Cologne (4C) at Cologne University of Applied Sciences, Cologne, Germany.
Division of Electricity, Department of Electrical Engineering, Uppsala University, Uppsala, Sweden.

Fabian Overath,
CAD CAM Center Cologne (4C) at Cologne University of Applied Sciences, Cologne, Germany.

Florian Klein,
Hoersch und Hennrich Architekten GbR, HHVISION, Cologne, Germany.

Martin Hennrich,
Hoersch und Hennrich Architekten GbR, HHVISION, Cologne, Germany.

Dr. –Ing. Christian Schyr
Advanced Solution Lab, AVL Deutschland GmbH, Karlsruhe, Germany.

Prof. Dr. Eng. Mats Leijon,
Division of Electricity, Department of Electrical Engineering, Uppsala University, Uppsala, Sweden.

Prof. Dr. rer. nat. Margot Ruschitzka,
CAD CAM Center Cologne (4C) at Cologne University of Applied Sciences, Cologne, Germany.

Abstract

Recently, virtual realities and simulations play important roles in the development of urban traffic infrastructure. By an appropriate abstraction, they help to design, investigate and communicate inner-city development processes. Especially, to investigate interactions between infrastructure and future mobility participants, a valid virtual model is essential for functionality and reliability.
The aim of this study is the investigation of interactions between a virtual infrastructure model and virtual sensor models of highly automated mobility systems. The overall system consists of a georeferenced virtual city model and probabilistic sensor models, which are part of an automated vehicle model. The virtual environment comprises a comprehensive, virtual 3D model of a representative German inner-city scene, considering specific height coordinates. The probabilistic sensor models represent real radar and lidar sensors and comprehensively replicate their physical functionality in a virtual environment. Considering different levels of detail, the realistic representation of physical effects of the virtual city model on the virtual sensors is investigated. The investigated scenarios are derived from representative urban traffic situations. The complexity as well as the level of information of the virtual city scenarios is iteratively increased. Subsequently, the effects on the model validity of the overall system is checked and analysed.
The results show that the developed virtual environment performs well for different levels of detail of representative virtual traffic scenes. In addition, the selected modelling depth is very suitable for the high-performance investigation of interaction between virtual urban environment and virtual autonomous vehicle.

1. Introduction

Simulation methodologies play an important role in the development of modern mobility solutions. Thereby, mobility is often defined by technical vehicle solutions. In this paper the interaction of urban traffic infrastructure and technical mobility solutions is investigated. The aim is to generate information about how urban traffic infrastructure has to be designed to enable safe and future oriented automated mobility. Therefore, a virtual traffic infrastructure is developed, which includes mobility solutions as well as urban infrastructure. The structure of this paper is shown in Figure 1.

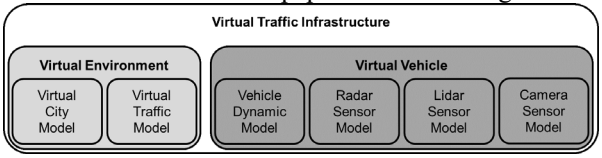

Figure 1: Structure of the virtual traffic infrastructure

2. Virtual (City) Environment

The virtual environment is separated into the geometry-based illustration of the city and the functional based implementation of traffic flows.

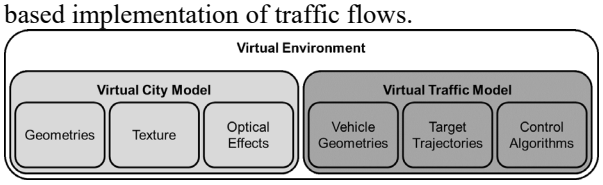

Figure 2: Structure of the Virtual Environment

2.1 Virtual City Model

The used virtual city model bases on a detailed laser scanned model of the city geometries. They include all kind of georeferenced parameters considering position, orientation and general dimensions. To decrease the amount of data und increase the quality of the surfaces, the laser-scanned data are reduced and retopologized first. Therefore, especially the level of detail is decreased and just the general city structure is captured. A example is show in Figure 3.

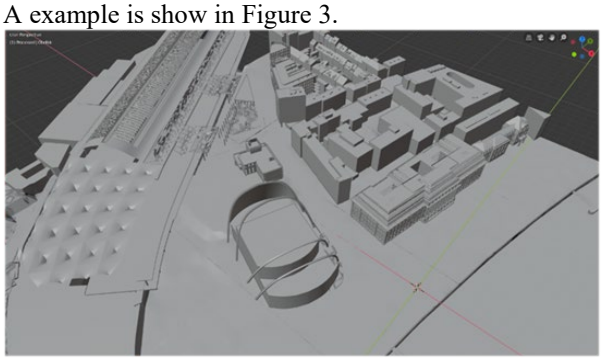

Figure 3: Reworked City Geometry Model

For the further procedure, the buildings, which are shown in the Figure 3 are separated out of the city geometry model and reworked individually up to a minimum detail level of Figure 3.With respect to the level of detail of the real building, the geometrical information of the virtual model is added. Especially, sharp contours of windows and balconies have to be reworked to approximate the reality. Afterwards the individual buildings were included separately info the Virtual Reality (VR) environment. The included model is schematically shown in Figure 4.

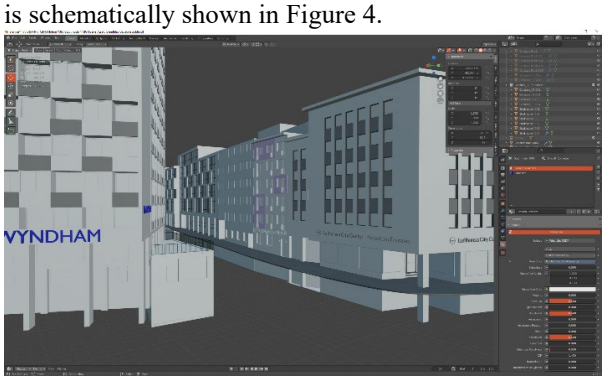

Figure 4: Geometrical building models in VR

In this model, each building is individually rebuild and textured to look like the real one. Additionally, the virtual buildings are fitted with meta-data to include specific information into the model. Afterwards, each building is positioned individually, whereby the virtual city model is recreated like the real city. In the same way, the city furniture is modelled and included into the city model in VR. Examples for these textured objects are shown in Figure 5.

Figure 5: Examples for city furniture in VR

Finally, the virtual city model is illuminated by virtual light effects. Therefore, a virtual light model is developed. This determines the individual interactions of lights and shadows depending the direction of the light sources in the scene. By using the virtual light model, individual scenarios like the sunset, a foggy day or the nocturnal darkness can be created.

2.2 Virtual Traffic Model

To supplement the virtual city environment, the geometrical city model is augmented by a virtual traffic model. Therefore, on the one hand, the georeferenced model is textured to look like real street illustration and on the other hand, virtual vehicles, including trajectory-planning algorithms, are added.
For texturing, especially two main aspects are represented. First, the traffic lines at the street-surface, second, the road composition are rebuild. An exemplary illustration of a virtual road is shown in Figure 6. It is important that the road signs and signal lights are correctly designed and placed in a country-specific manner, so that they can be identified by the sensors of the vehicles.

Figure 6: Exemplary illustration of a virtual road

To generate target trajectories for the simple virtual vehicles, the spline-path-follow-methodology is used. Therefore, the geometrical road is attributed by a spline. It is formed by connecting points between which the spline is interpolated. Thereby, each dot has three coordinates to specify its position. The course of the spline is generated depending on the previous and following dot, which smooths the spline course. Additionally, the spline course can be adjusted manually. Figure 7 shows a spline development based on dots.

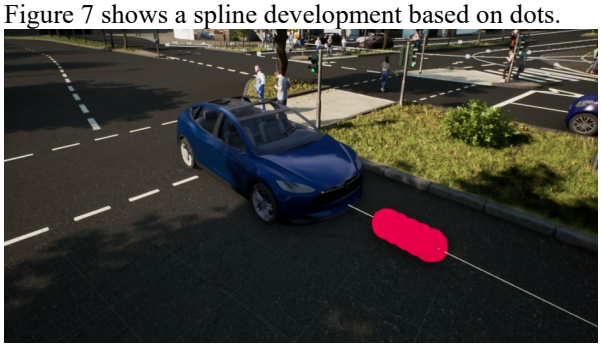

Figure 7: Schematically course of a spline path

By adding the spline-paths as target trajectories for the automated virtual vehicles, the virtual traffic is initialized. The virtual vehicle dynamic is based on a simplified vehicle dynamic model by the Nvidia PhysX Engine in the Unreal Engine. The trajectory controller is set up as a look ahead control. The steering angle is calculated by determining the difference between the look ahead point in front of the virtual vehicle and the target trajectory. Additionally, the vehicle velocity can be specified to complete vehicle trajectory design. The throttle input is calculated according to this speed limit. Additional to the speed limit, each spline contains references to possible succeeding splines. When a vehicle reaches the end of a spline, a succeeding spline is randomly chosen and the vehicle will follow a new spline. Each new spline contains the rules like, speed limits and possible spline changes.

Furthermore, the spline paths can be adapted by traffic functionalities like traffic lights. Methodically, bounding boxes around the objects are used for the interaction between the vehicles and the relevant traffic objects. If a vehicle bounding box crosses a traffic objects bounding box, for example a traffic light, a reference of the traffic light controller is passed to the controller of the vehicle. This reference contains the current state of the intersection e.g. the state of the traffic light. The controller of the vehicle then decides, depending on this state, whether it should continue driving or break and wait for the traffic light. A schematically illustration of an intersection traffic light scenario is shown in Figure 8.

Figure 8: Signal light intersection

In summary, the virtual city environment consists of a georeferenced inner city model attributed by an automated traffic model. In connection, a comprehensive urban traffic simulation is set up, which schematically represents real inner city traffic situations.

3. Virtual Vehicle

The interaction between automated mobilities and city infrastructures is an important issue in current automotive research activities. Hence, the following section will introduce an innovative virtual vehicle model for the representation of automated vehicle functionalities. The goal is the simulation of realistic vehicle behaviors in urban traffic situations. Figure 9 shows the general structure of the used and implemented model network.

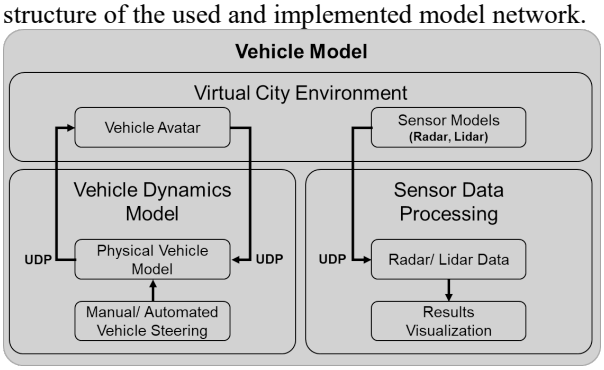

Figure 9: Model structure of the virtual vehicle

The core of the research environment is a highly authentic, visually realistic, georeferenced virtual reality city scene. To enable the mentioned interactions, the model is augmented by two submodels. A dynamic pedestrian avatar model, steered by a real-time network motion capturing and a vehicle model, including a physically correct dynamic model and three sensor models. The vehicle avatar serves as an interface to implement real vehicle functions into the scene. Its implementation is done in a closed loop communication between MATLAB and the Virtual Reality engine via network. Finally, three sensor models, implemented in the virtual city model, aim to simulate the recognized surrounding data by the vehicles sensors.

The visualization and potential further processing of the Radar and Lidar datasets takes place in MATLAB. The data transmission is done by network protocol

again. The aim is not only to simulate the data in a high realistic way, but also to create a decentralized structure that makes it possible to test complex scenarios independent of the location. The following sections show the detailed structure and the implementations of the described models.

3. 1 Vehicle Dynamic Model

To represent the real vehicles behavior of the ego vehicle, it is necessary to simulate the influences of the vehicle dynamics depending on the scenes environmental influences in a realistic way. Caused of real-time claims, it is important to find a complexity level that represents the cars movements in a sufficient accuracy, without taking too much computation resources. Most virtual reality environments, like the used Unreal Engine, aim not to implement physical simulation models. Therefore, the determination of the vehicle dynamics model is built in MATLAB Simulink. It is composed of different sub models, as shown in Figure 10.

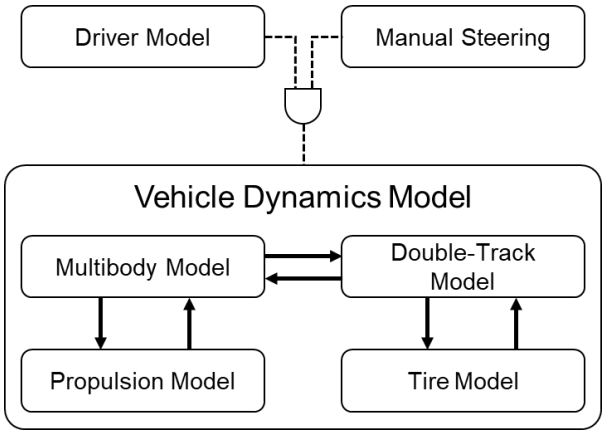

Figure 10 Vehicle dynamics model with sub models and steering input

By using a five body model, the vehicle vertical dynamics are simulated. The body's represent the vehicles main masses, consisting of the chassis and the four wheels. The wheels are stimulated by a foot-point stimulation depending of the surface-condition of the street. This allows the interaction of the model and the virtual city environment. A double track model simulates the lateral dynamics of the vehicle. Additional models also represent the drivetrain and the slip angle dependency of the tires as well as environmental influences like rolling, air and gradient resistances.

The steering of the model can either be controlled manually by a user input or automated by a driver model. For the manual steering, all common input devices can be connected to Simulink. This allows the user to move freely within the virtual scene and test the model network in various scenarios. For a better reproducibility, an automated driver model is implemented. It uses pre-defined spline paths, created in the virtual city environment and exported to MATLAB. The implemented path following algorithm bases on a pure pursuit controller, like described in (Samuel, 2016). It controls the vehicles steering input by following look ahead points in front of the car lying on the defined spline path. Theoretically, a dynamic path depending on the current driving state could replace the predefined spline path in MATLAB. Since path-planning algorithms are not part of the current work, this topic will not be further discussed. A vehicle avatar in the Unreal Engien connects the simulation model with vehicle geometries in the urban city environment. By connecting these models, a highly efficient Co-Simulation is realized.

3.3 Lidar Sensor Model

A typical environmental sensor for vehicle automation is the Lidar sensor. Therefore, in the following chapter, a Lidar sensor model for the application in VR is introduced. The technology uses laser beams to scan the surrounding. Multiple beams are sent either one after the other or at the same time. Depending on the transmission characteristics, the systems can be divided into single beam and multi beam scanning Lidars. Since the following model aims to simulate the physical properties of a Lidar, the scanning principle is not relevant for further considerations. It is only important that a possibility is provided, to control the shape and resolution of the Lidar field. For each sent Laser beam, time the light takes to travel to the target object and back to the sensor is determined. According to (Winner *et al,* 2016) this can be used to compute the distance as described in Equation 1.

$$d = \frac{c_0 \cdot t_{of}}{2} \tag{1}$$

In this equation, c_0 represents the speed of light, t_{of} is the duration the light travels and d is the distance to the target object. Since it is not possible to simulate the speed of light in a virtual environment, a substitution model is needed. Therefore, the so called linetracing or raytracing methodology is used. These virtual rays are defined by a starting point and an endpoint. If a ray hits an object within the scene, predefined information like the impact location were returned. To enable the coverage corresponding to a real sensor, the field in front of the virtual car is scanned at frequently. The azimuth and the elevation angle define the area of interest. For the discretization of the field, the angular resolution of the sensor in the respective direction is used. With that, the complete azimuth range is scanned, as shown in Figure 11. Then the elevation angle is incremented and

the azimuth angle get scanned again, until the whole field is covered.

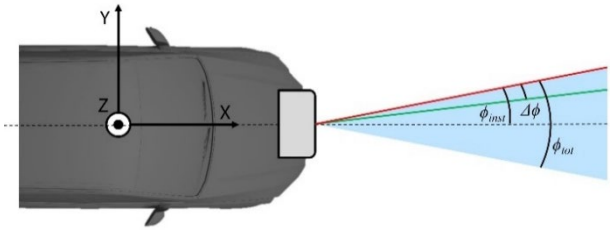

Figure 11 Discrete scanning of the Lidar field

If a ray hits an object, an algorithm is executed, computing the relevant data of the Lidar recognition. The aim is to determine whether a point is detected by the Lidar. An important value for this decision is the signal to noise ratio (*SNR)*, as defined in Equation 2.

$$SNR = \frac{P_r}{P_n} \tag{2}$$

Here P_r represents the power received from the Lidar and P_n is the sum of the induced noise powers. The higher the SNR value, the greater the probability of detection. Corresponding to (Winner *et al.*, 2016), (Kim *et al.*, 2013) and (Kernhof *et al.*, 2018) and through additional adaptations and assumptions the fraction of the received power can be expressed by Equation 3.

$$P_r = \frac{\rho_t \cdot A_r \cdot \tau^2 \cdot P_t \cdot \eta_{sys} \cdot \cos(\theta_i)}{Q_V \cdot \pi \cdot d^3} \tag{3}$$

In this equation, ρ_t is the reflectance coefficient of the target object; A_r represents the receiving lens area; τ stands for the atmospheric transmission coefficient; η_{sys} are the summarized system losses; θ_i is the incidence angle of the light beam on the objects surface and Q_V stands for the divergence of the shot beam. For the implementation of this equation into the virtual reality, most of the parameters can be passed as variables. Only the incidence angle and the targets distance depends of the linetraces. The distance can directly be read out of the linetrace results, the incidence angle can be computed by equation 4 using the incidence vector $\underline{i}$ and the surface normal $\underline{n}$ at the impact point.

$$\theta_i = \cos^{-1}\left(\frac{\underline{i} \circ \underline{n}}{|\underline{i}| \cdot |\underline{n}|}\right) \tag{4}$$

Besides the received power, the noise powers acting on the Lidars receiving systems need to be determined. They are mainly composed of the sun induced noise and the dark current noise. The sun induced noise is generated by sunlight illuminating the targets surface and impinging the sensor. Equation 5, according to (Kim et al., 2013), can compute the power of this noise source.

$$P_{sun} = E_{Si} \cdot B_\lambda \cdot \rho_t \cdot A_r \cdot \tau \cdot IFOV^2 \cdot \eta_{sys} \tag{5}$$

E_{Si} represents the illumination intensity of the sunlight, B_λ is the electromagnetic bandwidth of the receiving unit and IFOV is the instantaneous field of view. In case the same optics for receiving and transmitting the signals is used, the IFOV corresponds to the beam divergence. Thermal effects of the photo element generate the dark current noise. For the computations of this noise, Equation 6 is used.

$$P_{DK} = \frac{I_D}{\Re_{\max}} \tag{6}$$

Here, I_D stands for the dark current and $\Re_{max}$ represents the maximum sensitivity of the photo element. Both parameters can usually be found in the datasheets of photo elements. By this, it is now possible to compute the signal to noise ratio and make a decision whether a point gets recognized or not. The computation is computed for every ray hitting an object within the scene.

At every simulations step different metadata like the SNR-value and the determined powers are generated. However, it is problematic, that the coordinates of captured points are shown as perfectly accurate values. Real Lidar sensors have a limited resolution due to the time capturing system, amplifications and analog to digital conversions (Kernhof *et al.*, 2018). Since commercial sensors differ in capturing technologies and used hardware, it is not feasible to model these inadequacies accurately. Instead, the influence of the acting inadequacies are displayed. In the given case, this is done by multiplying the resolution, provided by most sensor manufacturer, with a white Gaussian noise and adding the result to the distance value.

Therefore, all relevant data are known and ready for further processing. To enable this, the values are stored in arrays and sent to MATLAB by an UDP communication. To generate a practical reference, the sensor is parametrized according to the *Valeo SCALA 3D Laser Scanner*, a commercial serial product for ADAS applications. The determined values shown in Table 1 are mainly taken from the sensors datasheet (Hexagon, 2021). All missing values are adopted from the datasheet of a typical photodiode (Hamamatsu, 2018) and the literatures (Weber, 2018) and (Kim, et al., 2013). The environmental parameters like the atmospheric transmission coefficient or the irradiance of the sun are set dynamically within the virtual scene, depending on the particular study.

Table 1 Parametrization of the virtual Lidar according to the Valeo Scala sensor

	Symbol	Value	Unit
Azimuth Angle	ϕ_{tot}	145	deg
Elevation Angle	ϑ_{tot}	3.2	deg
Azimuth Resolution	$\Delta\phi$	0.25	deg
Elevation Resolution	$\Delta\vartheta$	0.8	deg
Distance Resolution	ΔR	0.1	m
Radiated Power	P_t	80	W
Lens Area	A_r	0.0007	m²
Beam Divergence / Instantaneous Field of View	Q_v / IFOV	0.003	rad
Electromagnetical Receiver Bandwidth	B_λ	2	nm
Dark Current	I_D	10	nA
Sensitivity Photo Diode	$\Re_{max}$	0.5	A/W
System Efficiency	η	0.9	-

3.2 Radar Sensor Model

A further typical environmental sensor for ADAS application is the radar Sensor. It is commonly used for automotive since it is comparatively cheap, provides a large detection distance and is resistant against environmental influences. The virtual environment of this paper aims to simulate all necessary environmental data for the interaction of autonomous vehicles and driving functions with pedestrians in real time. Hence, a probabilistic radar model is used. It provides all relevant data in an object list and simulates the phenomena of the radar technology, without performing a full physical simulation.

The implemented model is largely based on the probabilistic radar model presented by (Muckenhuber, et al., 2013) and is extended by several assumptions. To represent the model as realistic as possible, the real commercially used radar sensor *Continental ARS 408* serves as a reference. (Liebske, 2015) provides the Datasheet. The execution of the model takes place in two steps, as shown in Figure 12.

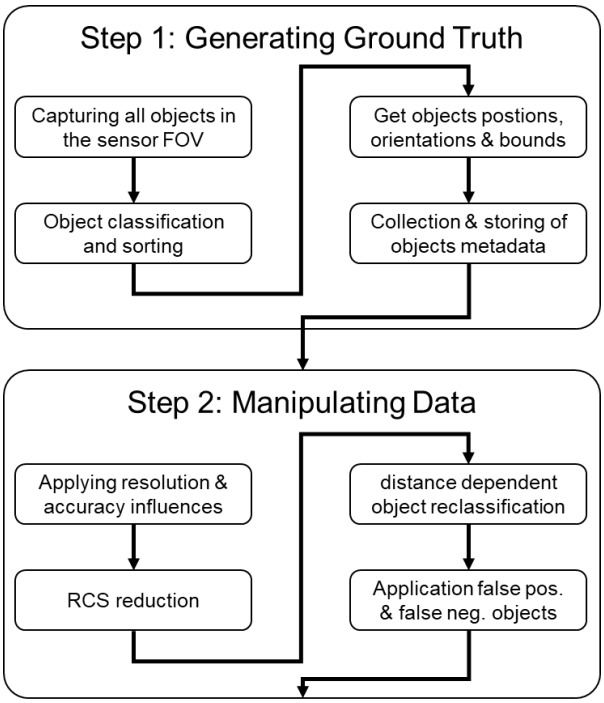

Figure 12 Overview of the probabilistic radar model execution

In a first step, the so-called Ground Truth Data need to be generated. It is a dataset, containing all possibly detectable objects in the sensors field of view and the corresponding metadata. After the perfect data are generated, the second execution step manipulates the dataset with respect to measurement errors, resolutions and inaccuracies. Hence, the phenomena of the sensor are mapped.

For the generation of the Ground Truth all relevant objects lying in the detection area of the sensor within the scenery need to be captured. The datasheet of the sensor provides the sensing areas displayed in Figure 13.

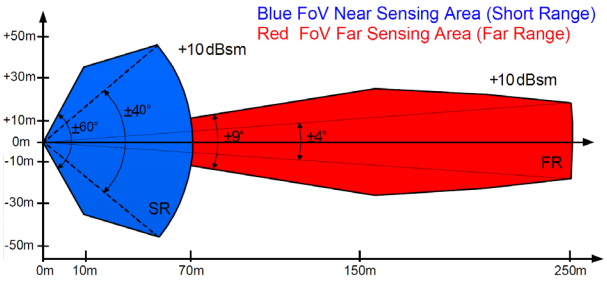

Figure 13 Sensing areas of the Continental ARS 408 (Liebske, 2015)

Caused of inner-city applications, only the near sensing area needs to be modeled. As with the previous described Lidar model, a linetracing method is used to get the relevant objects, similar to Figure 11. However, for this probabilistic model it is not necessary to get the data of the trace itself. Instead, the objects are passed

to an array, if they are hit the first time. The angular resolution of the linetracing needs to be much higher than at the Lidar model, not to omit any object. Moreover, the detection range of the sensor is not constant over the azimuth angle. Hence, the detection field is adapted to the near sensing area as shown in Figure 13.

As already mentioned, the result of the linetraces is an array containing all actors within the field of view of the virtual sensor. Since not all objects are relevant for the virtual sensor, the next step is the classification and sorting of the objects. Table 2 provides the types of the objects for the classification.

Table 2 Object classifications for the probabilistic radar model

Typ	Index	Color
Car	1	Cyan
Truck	2	Blue
Pedestrian	3	Red
Motorcycle	4	Yellow
Bicycle	5	Green
Unknown	6	Magenta

Based on that, all relevant actors within the virtual scene are augmented by a so called tags. These tags are detected, if the actor is hit by a linetrace. A color is assigned to each class for visualization purposes. The object designations are predominantly self-explanatory. The “Unknown” class contains all geometries, that are generally received by the radar sensor, but without classification, like postboxes, street signs, advertising pillars and other street furniture. Objects that have no predefined class are omitted. In a next step, the missing metadata of the relevant actors are determined. The hit objects directly return information according to positions and orientations of the object. To become local coordinates in the system of the sensor, a coordinate transformation from the global system has to be performed. Additionally, the size of the minimum surrounding box, the so called bounding box, is generated for every found object.

Finally, the ideal maximum radar cross sections (RCS) is missing for the recognized objects. This value gives an indication of how large the proportion of the reflected radiation energy is, that impinges on an object. In further processing steps the RCS value can be used to make a decision, if an object is recognized. As the objects index, the value is predefined in tags for each relevant actor within the scene. Since the value fluctuates depending on the aspect angle, the pre-defined value provides the maximum possible radar cross section. Previous works like (Degen, et al., 2021) show the general possibility to simulate radar cross section within a virtual reality engine. However, the paper also offers issues in real-time performance. Due to that, the current probabilistic model uses a pre-defined RCS for every object and manipulates it to get a realistic value. The method will be described in the further progress of this paper.

The physical phenomena of the sensor are replicated, without simulating its full physics. At first, the influences of the sensors resolution and accuracy are applied to the ground truth signal as displayed in Figure 14.

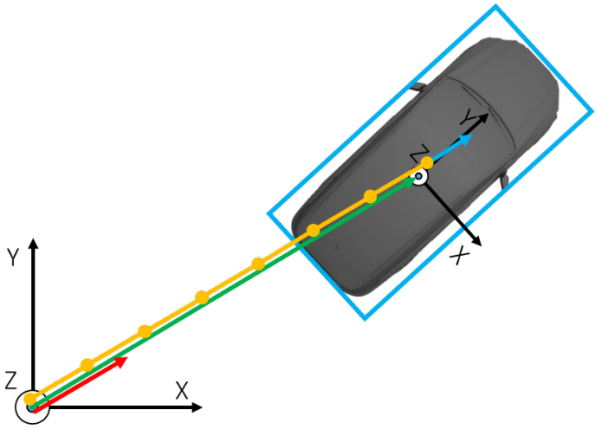

Figure 14 Visualization of the data manipulation to simulate the sensors resolution and accuracy.

The vehicle coordinate system and a theoretical target object are shown. The green vector represents the ideal and error-free position of the target, taken from the ground truth. To implement the resolution influences so called range gates are formed. The size of the range gates corresponds to the resolution. To sort the distance into the range gates, the length of the vector is divided by the resolution and rounded to an integer value. This results in the number of range gates. Subsequently the resulting integer value is multiplied with the resolution again. This leads to the yellow vector. The effect of the measurement accuracy is implemented next. This value fluctuates randomly in positive and negative direction. Thus, a static implementation is not possible. Instead the value of the measurement accuracy, taken from the datasheet (Liebske, 2015), is multiplied with a white Gaussian noise with a standard deviation of one. The resulting vector, visualized in blue, is added to the yellow in range gates sorted vector.

After the range manipulation, the next step is the implementation of aspect angle and coverage effects to the RCS value. The principal methodology is shown in Figure 15.

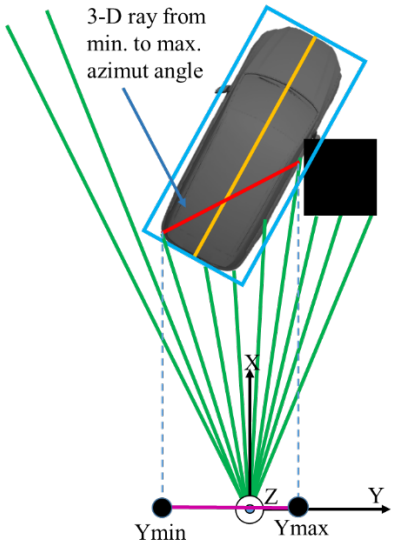

Figure 15 Method for the manipulation of the RCS value

The manipulation is based on the assumption, that the maximum RCS of an object is given at its longest side. In Figure 15 this is represented by the yellow line. To simulate coverage and aspect angle influences, the already carried out ray traces are used. First, the first and the last ray trace that hit the geometry are determined for each object recognized. It is obvious that covered parts of the targets objects are not noticed with this method. The result is shown as a red line. In a next step, this red line is projected onto the Y-axis of the local sensor coordinate system. To artificially reduce the ground truth RCS, it is multiplied with the quotient of the length of the yellow line, representing the objects longest side and the length of the purple line, representing the visible fraction of the object. This completes the reduction of the RCS.

According to Figure 12 the next step of the probabilistic model is the object reclassification. Real radar sensors classify objects on their radar signature. This is only possible in certain distances, depending on the objects type. So far, all objects get recognized and classified correctly. To simulate the real radars detection behavior, two thresholds are implemented. It is assumed, that cars, trucks and motorcycles are correctly classified up to a distance of 50 m. For Pedestrians and bicycles, the assumed threshold is 30 m. If the distance of any object exceeds the threshold defined for its class, it gets reclassified to the “unknown” class.

The last manipulation step visualized in Figure 12 is the implementation of false positive and false negative objects. With real radar sensors two phenomena can occur. On the one hand, objects can appear that are not physically part of the real surrounding. These are called false positive (FP) objects. On the other hand, objects that are physically part of the sensing area can stay undetected for a certain time. These are named as false negative (FN) objects. Both phenomena are implemented in the following, starting with the FN objects. A detection probability with a value between zero and one is pre-defined for every object class. On every execution step and for every actor a pseudo random value, also between zero and one, is generated. If the generated value is smaller than the detection probability, the object is added to the object list. If it is larger, the object gets neglected. With this, it is possible to adjust the average false negative rate for every object class. The implementation of false positive objects is more complex, as the objects are not only to be implemented, but also random positions have to be found for them. At a first step a value for the average number of recognized FP objects at every frame is defined depending on the object classes. Additionally, typically bounding box sizes are defined for every object class. The FP objects are added after the execution of the complete probabilistic radar model, when all objects with metadata reduced by the FN objects are known. The generation of the FP objects is done for every object class separately, through a loop that iterates the object types. In that loop the class individual average number of FP objects is used to compute the actual number of FP objects for the respective simulation step. This is done by applying a white Gaussian noise to the value and converting it to an integer. After that, a second loop is initiated for the generated integer value. Within that loop, the respective FP object is generated. The first value, that is generated for each false FP object is the extend. For that, the pre-defined average size of the class dependent bounding box is manipulated by a Gaussian noise. Thus, the average size of the bounding box corresponds to the predefined value, but the individual sizes vary within a certain range. The same procedure is also used to generate an artificial RCS value for the respective object. After all metadata are created, the respective object gets positioned at a random position with a random orientation within the sensors field of view. This is repeated until the computed number of false positive objects for the object class is reached. After that, the Object class is incremented and the algorithm is executed again.

4. Investigation of the Virtual Traffic Infrastructure

For the interdisciplinary city infrastructure planning, the impact of infrastructure and automated mobility is crucial. Therefore, a valid virtual environment, which enables investigation of interaction between mobility sensors and the traffic infrastructure is essential. In this chapter, the developed virtual city infrastructure is investigated to evaluate its quality for further applications. The base for the investigation are representative urban traffic scenarios, which illustrates typical challenges for ADAS sensors in inner-city environments.

4.1 Crossing with urban structure

Especially at urban environments, buildings and city infrastructure cause shadings. A typical scenario is the crossing, whereby urban development restricts the driver's field of view. Figure 16 shows the scenario from different perspectives.

Figure 16:
Top: Bird´s-eye view of the scene *Crossing with urban development,*
Bottom: Vehicle-top view of the scene *Crossing with urban development.*

The bird´s-eve view illustrates the general scene. Especially the building on the right-hand side of the ego-vehicle restricts the field of view. The view into the crossing street depends straight of the distance to the crossing. A holistic view is only possible just before entering the crossing.
Additionally, the Lidar-beam course is visualized in Figure 16 bottom. Next to the distance, which is shown by the color of the Lidar-dots, the georeferenced course of the virtual environment gets clear. The dots hit the street at a far distance in front of the ego-vehicle.

4.2 Street line with street furniture

Another interesting city application for the mentioned investigation is the effect of street furniture to the ADAS sensors. The general properties of a bin or an advertising pillar for example are quiet related to human properties. Figure 17 visualizes a *street line with street furniture* scene.

Figure 17:
Top: Bird´s-eye view of the scene *Street line with street furniture,*
Bottom: Vehicle-top view of the scene *Street line with street furniture.*

Besides to the urban structures, typical street furniture like bins, advertising pillars and traffic signs are included. Additionally, the course specific of the Lidar-sensor is shown. Thereby especially the falling course of the street is becomes clear.

4.3 Pedestrian crossover

Crossovers are the typical urban interfaces between pedestrians and vehicles. Here, traffic signs, other vehicles or buildings restrict the field of view often. The implementation in the virtual environment of this scenario is shown in Figure 18.

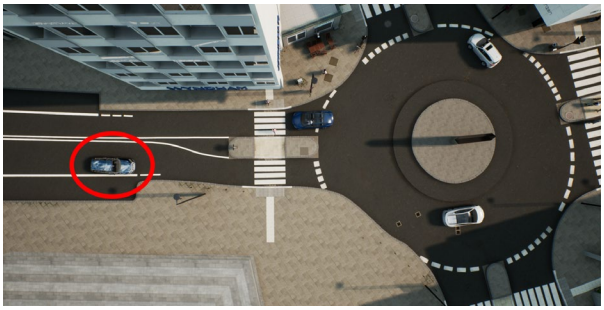

Figure 18:
Top: Bird´s-eye view of the scene *Pedestrian crossover,*
Bottom: Vehicle-top view of the scene *Pedestrian crossover.*

Next to the roundabout, which is not focused in this investigation, the pedestrian crosswalk with the approaching ego-vehicle is illustrated. Additionally, pedestrians on and at the beginning of the crossover are implemented. Finally, the scenario is filled with street and traffic furniture.

4.4 Street with structural separation

The final investigation-scene is the multi-lane street with a structural separation of the driving directions. In this implementation street greenings like bushes and trees provide the separation (shown in Figure 19).

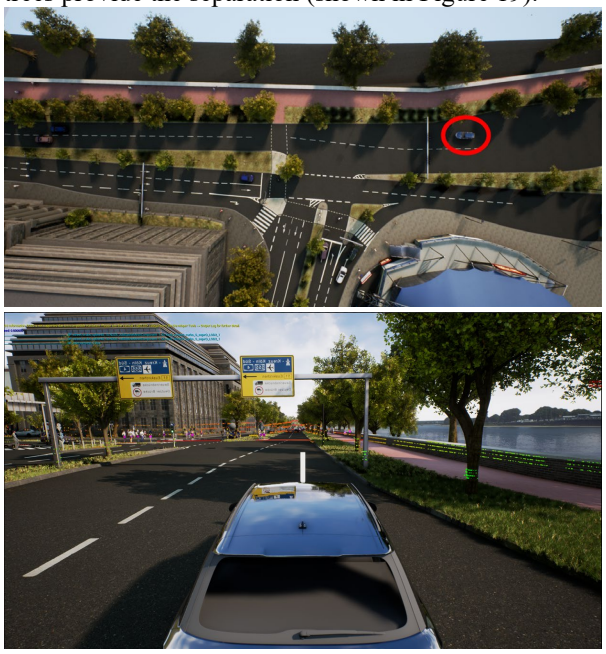

Figure 19:
Top: Bird´s-eye view of the scene *Street with structural separation,*
Bottom: Vehicle-top view of the scene *Street with structural separation*.

In Figure 19 bottom, the vehicle-top view of the scene is illustrated. It shows the diversity of a typical inner-city scene with a river and a bicycle and pedestrian sidewalk at the right-hand side and urban structures at the left-hand side. All lanes are divided by boundaries like greenings and walls.

5. Analysis and Results

To evaluate the suitability of the virtual city environment for the investigation of interactions between traffic infrastructure and ADAS developments, the data of the virtual sensors are analyzed. For this reason, the previous investigation study is evaluated.

5.1 Crossing with urban structures

First, the crossing scenario is analyzed by reviewing the radar-sensor model data and the lidar-sensor model data.
The measured data of the lidar-sensor model are shown in Figure 20.

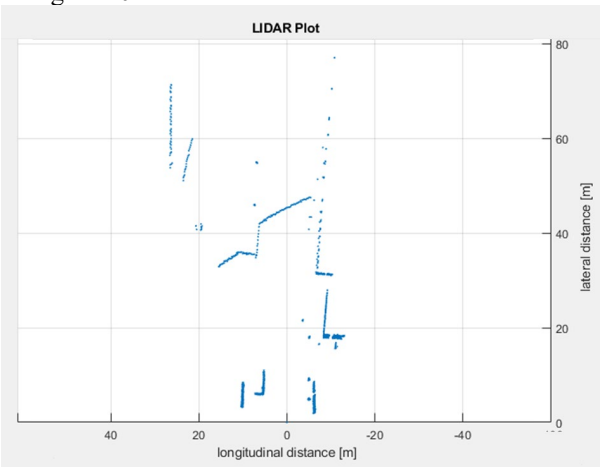

Figure 20: Lidar-Plot of the scenario *Crossing with urban development*

The plot of the lidar-sensor model shows quiet good results. First, the general edges of the urban structures are detected. Additionally, the vehicle standing on the left-hand side of the ego-vehicle is captured by the lidar. Further, the street-surface including the sidewalk-edge far in front of the ego-vehicle is perceived. Also, the street furniture like street lights and traffic signs are detected well. This is shown in the Lidar-plot at 20 m lateral distance and 6 m longitudinal distance. Additionally, the lidar detcts a short surface of the vehicle standing in the crossing street right-hand side to the ego vehicle. This can be seen at 18m longitudinal axis and 11m lateral axis. Finally, the two pillars of the building right-hand side of the ego vehicle are recognized, which correlates with the illustration at Figure 16 bottom. The course of the lidar dots, which represents the field of view, includes two pillars; the first one is out of the detection area.

Caused by the model setup, the radar sensor only detects classified objects. The plot of the sensor at the *crossing with urban development* scene is illustrated in Figure 21.

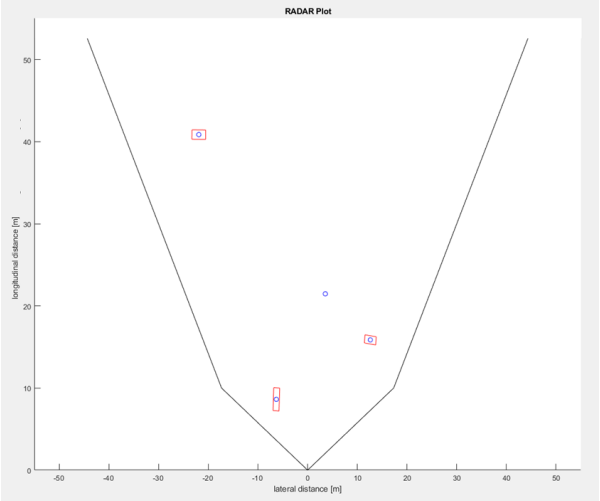

Figure 21: Radar-Plot of the scenario Crossing with urban development

A direct comparison with Figure 16 makes clear, that the consideration of the shadings due to the urban building is given. The vehicle standing at the beginning of the crossing road is recognized with reduced RCS. This could be seen by the minimized classification box around the detected object in the radar plot. Further, the vehicle at the sidewalk left-hand side to the ego vehicle is detected and classified. Additionally, one object in front of the ego-vehicle is received and not classified. The comparison with the lidar plot shows clearly, that this object has to be a traffic sign.

5.2 Street line with street furniture

To investigate the uniqueness of the street furniture detection, the sensor data of the *Street line with street furniture* scenario are analyzed.
The sensor data plots are shown in Figure 22 and Figure 23.

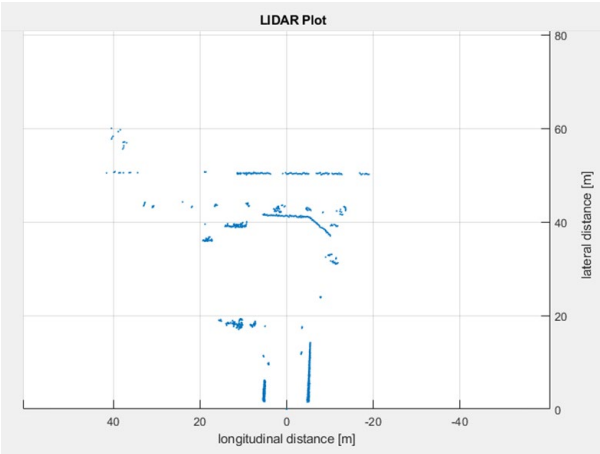

Figure 22: Lidar-Plot of the scenario *Street line with street furniture*

By comparing the results with Figure 17 especially the contours of the passing street becomes clear. The lidar plot shows the course of the street lane between 40 m and 50 m. A little wall and street greening border this part of the road. Additionally, the buildings directly next to the ego vehicle are detected, which is represented by straights in the plot. Furthermore, the lidar recognizes the diversity of street furniture in the longitudinal distance between 15 m and 40 m. Despite of the little dimensions of for example street lights and the big distance to the ego vehicle, the lidar-sensor model receives it quite well.

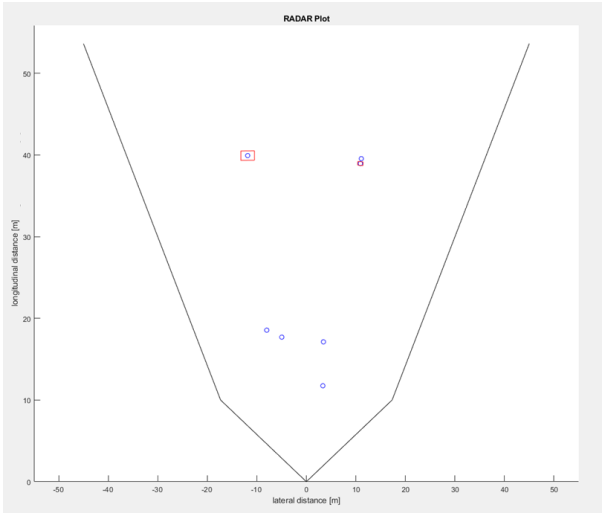

Figure 23: Radar-Plot of the scenario *Street line with street furniture*

The radar sensor detects less objects. Just one vehicle is detected and classified. However, the functionality is very well. As shown in Figure 17 top, the crossing vehicle is just hit at a little spot of the front (see the red dot). The radar sensor detects it, classifies it correct and reduces the RCS in dependency of the received area. The reduction of the RCS could be seen at the little dimensions of the object box in Figure 23 at 40 m longitudinal distance to the ego-vehicle. In addition, street furniture is also recognized. At the right-hand side in front of the ego-vehicle a bin and a traffic sign are detected. At the left-hand side an advertisement pillar and a traffic sign are recognized.

This examined scene shows well the individual fields of application of the implemented sensor models. While the radar sensor model returns a classified object, the lidar only shows the object contours. The classification has to be done by an external algorithm.

5.3 Pedestrian crossover

The crossover scene is one of the most complex scenarios investigated. Next to the diversity of participants, the road course is falling and behind the crossover is a roundabout. The complexity becomes clear by consideration of Figure 24. The lidar-plot shows the variety of contours an individual dots, which has to be assigned.

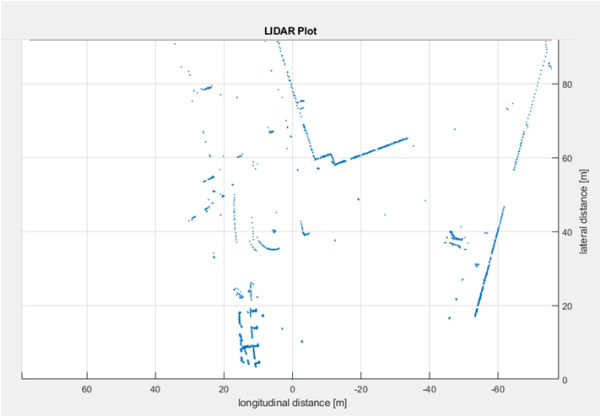

Figure 24: Lidar-Plot of the scenario *Pedestrian crossover*

First, the complex architecture of the building at the left-hand side of the pedestrian crossover stands out. The pillars and edges of the building are recognized well. Furthermore, one pedestrian at the crossover as well as the traffic sign in the middle of the crossover and the street light are detected, which could be seen in the lidar plot between 15m and -5m longitudinal distance and 5m and 20m longitudinal distance. Interesting is the effect, that the second person at the crossover is shaded by the traffic sign. The sensor does not receive the person. Additionally, around 40m in front of the ego vehicle, the edges of a vehicle as well as the statue in the middle of the traffic circle are detected. Next, the contours of two buildings at the right-hand side and quite in front of the ego-vehicle are detected, which gets obvious by comparison with Figure 18 bottom. Finally, street furniture like traffic signs and street lights as well as restaurant furniture is captured, which could be seen especially at the left-hand side in the height of the traffic circle.

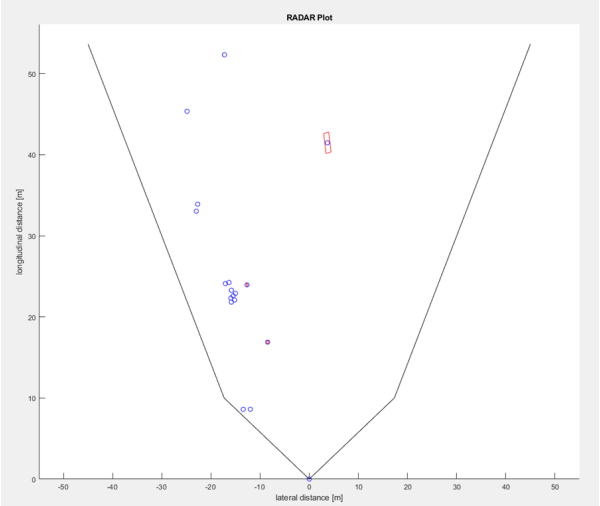

Figure 25: Radar-Plot of the scenario *Pedestrian crossover*

In detail, both the complexity and diversity of the lidar sensor results are confirmed by the radar plot data (Figure 25). One vehicle in the traffic circle is captured and classified. Additionally, one person at the pedestrian crossover as well as the street light are detected. Furthermore, the sensor captures the constellation of the street furniture and traffic signs at the left-hand side part of the traffic circle.

5.4 Street with structural separation

At least the sensor data of the *street with structural separation* scenario are analyzed. Like bevor, radar and lidar sensor data are used for the analysis.
The lidar-plot is shown in Figure 26 and the radar plot is shown in Figure 27.

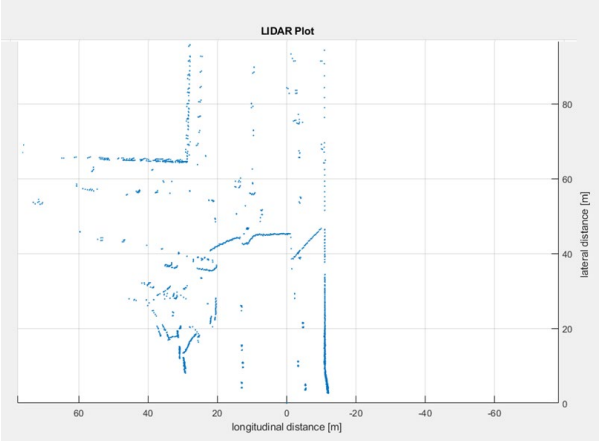

Figure 26: Lidar-Plot of the scenario *Street with structural separation*

In general, the lidar plot shows the course of the street quite well. The boundaries like the wall at the right-hand side of the vehicle as well as the building at the left-hand side further back are captured well. Furthermore, the diversity of bushes and trees as well as traffic signs at the greening around the road are detected correctly. Conspicuous is the red line in Figure 19 bottom. This line, which represents the lidar beams, is illustrated in the lidar plot between 40m and 50 m longitudinal distance, too. It is caused by the geo-individual course of the road. The course of the road increases in height. Finally, different pedestrian contours are captured at the pedestrian crossover at the left-hand side in front of the ego vehicle (see Figure 19 top).

In sum, the interpretation of the lidar plot is very complex. Without the direct comparison of the scene picture, especially the assignment of little contours is difficult. This shows quite well, how essential the connection of the different sensor information is for highly automated vehicle systems.

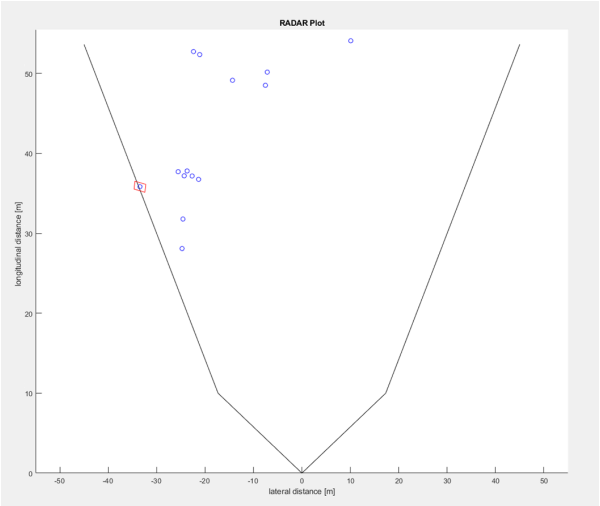

Figure 27: Radar-Plot of the scenario *Street with structural separation*

In this case, the radar plot shows a significantly reduced complexity. Mainly, the radar captures the pedestrians walking through the crosswalk between 25 m and 40 m longitudinal and around -25 m lateral in front of the ego vehicle. Additionally, the radar recognizes the vehicle standing in front of the pedestrian crosswalk.

Finally, the radar detects other pedestrian and street lights around 50 m in front.

In detail, this scenario shows the level of detail, which has to be captured by the different sensors. Either in reality or in virtual, especially object in far distance are difficult to detect and classify.

Conclusion

In summary, the virtual environment as well as the virtual vehicle model fulfill the given expectations. The interaction of virtual sensors and virtual traffic participants works reliably and reproducibly. The necessary metadata for the computation of the virtual sensor models are provided. Hence, it is possible to investigate border line cases of surrounding sensor detections within the virtual world. The model will help planning and designing street topologies, positions for city furniture and car to pedestrian interactions in future urban development projects. The next steps are to increase the level of detail of real traffic participants by adding motion realistic behavior, as well as extend the sensor model portfolio by implementing a camera sensor model.

Acknowledgements

The Project is funded by the Ministry of Economic Affairs, Innovation, Digitization and Energy of North Rhine-Westphalia. Additionally, the project is supported by HH Vision, Hoersch und Hennrich Achitekten GbR and AVL Germany GmbH.

References

Degen, R., Ott, H., Overath, F., Schyr, Ch., Leijon, M., Ruschitzka, M. (2021) *Methodical approach to the development of a Radar Sensor model for the Detection of Urban Traffic Participants Using a Virtual Reality Engine. Journal of Transportation Technologies.* 11, 02, 179-195.

Hamamatsu Photonics K.K. (2018). https://www.hamamatsu.com/resources/pdf/ssd/s12023-02_etc_kapd1007e.pdf

Hexagon Autonomy and Positioning (2021). https://autonomoustuff.com/products/valeo-scala

Kernhof, J., Leuckfeld,J. and Tavano, G. (2018). *LiDAR-Sensorsystem für automatisiertes und autonomes Fahren*, in: Thille, T. *Automobil-Sensorik 2*. Springer Vieweg. Berlin, Heidelberg.

Kim, S., Lee, I.and Kwon, Y. J.(2013). *Simulation of a Geiger-Mode Imaging LADAR System for Performance Assessment*. Sensors 13 ,7, 8460-8489

Liebske, R. (2015), Contionental Automotive. ARS 408-21 Premium Long Range Radar Sensor 77 GHz.

Muckenhuber, S., Holzer, H., Rübsam, J., Stettinger, G. *Object-based sensor model for virtual testing of ADAS/AD functions. 2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*

Samuel, M., Hussein, M. and Mohamad, M. B. (2016). *A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicle. International Journal of Computer Applications* **135, 1**, 35-38

Weber, H., (2018). *Funktionsweise und Varianten von Lidar-Sensoren*. Sick AG.

Winner, H., Hakuli, S., Lotz, F. and Singer, C. (2016). *Handbook of Driver Assistance Systems*. 1st edn. Springer International Publishing. Cham.