electronics

MDPI

Article

High-Level Sensor Models for the Reinforcement Learning Driving Policy Training

Wojciech Turlej [1,2]

[1] Aptiv Services Poland S.A., ul. Podgórki Tynieckie 2, 30-399 Cracow, Poland; wturlej@agh.edu.pl
[2] AGH University of Science and Technology, Al. Mickiweicza 30, 30-059 Cracow, Poland

Abstract: Performance limitations of automotive sensors and the resulting perception errors are one of the most critical limitations in the design of Advanced Driver Assistance Systems and Autonomous Driving Systems. Ability to efficiently recreate realistic error patterns in a traffic simulation setup not only helps to ensure that such systems operate correctly in presence of perception errors, but also fulfills a key role in the training of Machine-Learning-based algorithms often utilized in them. This paper proposes a set of efficient sensor models for detecting road users and static road features. Applicability of the models is presented on an example of Reinforcement-Learning-based driving policy training. Experimental results demonstrate a significant increase in the policy's robustness to perception errors, alleviating issues caused by the differences between the virtual traffic environment used in the policy's training and the realistic conditions.

Keywords: sensor modeling; traffic simulation; reinforcement learning; autonomous driving

Citation: Turlej, W. High-Level Sensor Models for the Reinforcement Learning Driving Policy Training. *Electronics* **2023**, *12*, 71. https://doi.org/10.3390/electronics12010071

Academic Editors: Paweł Skruch, Joanna Jaworek-Korjakowska, Saleh Mobayen and Damian Grzechca

Received: 9 November 2022
Revised: 16 December 2022
Accepted: 20 December 2022
Published: 25 December 2022

1. Introduction

Advanced Driver Assistance Systems (ADAS) and Autonomous Driving Systems (ADS) possess the potential to significantly increase traffic safety by reducing the impact of human errors, which are by far the most common cause of accidents. Unfortunately, these systems do not come without their own risks - Highly Automated Vehicles (HAVs) may be susceptible to hardware and software malfunctions, as well as errors caused by the performance limitations of their perception systems. For these reasons HAVs require extensive validation and verification efforts to ensure that they meet their safety goals.

An intuitive way to assess the safety of ADAS/ADS and acquire data needed to develop and improve them is to use physical test drives. While this approach has been applied successfully for the development of ADAS features with a low level of autonomy, its application in HAVs poses a few severe challenges. End-to-end validation of ADS requires immense quantities of data to be gathered [1]. Furthermore, the development of certain ADAS/ADS algorithms, such as driving policies based on Reinforcement Learning (RL), requires triggering potentially dangerous behaviors and situations, which would be unacceptable to do in public traffic.

Traffic simulation tools are thus commonly utilized in the development and testing of ADAS/ADS [2]. The simulated road environment can be employed in both the open-loop testing, where actions of the vehicle equipped with the tested system, called the ego vehicle, do not impact the environment, and in closed-loop setups, where the traffic participants actively respond to the ego's actions. The simulation is also a key component in RL-based driving policies training techniques, which typically utilize it to learn proper responses to various situations observed in the ego's environment [3].

Both the testing and training applications require an accurate representation of the environment as perceived by automotive sensors, such as radars, LiDARs, or cameras. All of these sensors suffer however from various errors and performance degradation, that may be difficult to accurately recreate in a simulation environment. In this paper,

I propose a set of high-level sensor models that can be used to realistically disturb the ground-truth descriptions of both the static and dynamic environments of the ego vehicle in a closed-loop simulation setup. Proposed models are applied to the RL-based driving policy training and their impact on the system's performance is analyzed in a series of virtual driving experiments.

1.1. Related Work

Sensor models used for simulating the automotive perception systems can be most broadly assigned to two major classes: low-level sensor-specific models, that attempt to simulate error modalities of a particular sensor type, often through relatively precise simulation of underlying physical phenomena, and high-level generic models, that attempt to coarsely mimic perception errors through sensor-type-invariant statistical methods or heuristics.

Sensor-specific models of the automotive radar typically attempt to re-create error patterns related to physical properties of the radar wave, that lead to false-negative detections due to e.g., occlusions or weak reflections, and false positives caused by multi-path reflections of the wave. Depending on the computational resources and desired model's accuracy, these phenomena may be either simulated using high-fidelity models in a relatively accurate manner, e.g., taking into account precise antenna characteristics [4], or approximated using statistical methods or low-fidelity models. High-fidelity methods include the use of the ray-tracing algorithms [5] that utilize geometric models of road users [6], often with pre-determined virtual scattering centers [7].

Due to the complexity of the radar modeling task and the nonlinear nature of the underlying models, Machine Learning techniques are often proposed to simulate radar's error patterns. Muckenhuber et al. in [8] evaluate several machine-learning radar models trained on a labeled dataset. The authors of [9] utilize deep learning techniques to create a model that takes into account both static and dynamic environment features to produce a realistic model of the sensor's output.

While the accurate simulation of underlying physical phenomena that impact radar sensors is a relatively complex task, high-fidelity simulation of the raw camera output can be performed by rendering the simulated scene and applying appropriate lens distortion models [10] and color space conversion [11]. It should be noted, however, that high-resolution rendering and execution of the camera's object detection algorithms to produce an object list often require immense computational effort. Furthermore, acquiring high-quality results requires an accurate model of the environment, including 3D models of traffic participants, high-resolution textures, and realistic simulation of weather conditions.

Similarly, as in the case of radar sensor models, the use of machine learning techniques may help to achieve realistic output with a lower computational effort. Generative Adversarial Networks are often proposed for this task, due to their capability of producing realistically-looking images [12].

Due to the high computational cost of certain tasks that require sensor modeling, such as training of Reinforcement-Learning-based driving policies, low-fidelity models that operate on the object list instead of raw data are often proposed as an efficient alternative [13]. Object-level models may be also useful in situations where perception output incorporates a fusion of data from several types of sensors, and/or tracking algorithms. Examples of such setups include radar and vision fusion for detection of road barriers [14], road curbs tracking [15], and radar-camera sensor fusion for objects detection [16].

Analysis of the perception errors' impact on driving policies is also an important area of research that utilizes sensor models. In [17], the authors proposed an error model for the LiDAR sensor and analyzed its impact on the control algorithms used for cooperative driving. [18] indicated that generic sensor models may play an important role in the training of Reinforcement-Learning-based driving policies, improving the robustness of the trained policy to the real-life performance limitations.

1.2. Motivation

The importance of sensor modeling in the validation and verification of the ADAS and AD systems is well understood, and models of various levels of fidelity are relatively commonly used in such applications. The emergence of the ADAS/AD algorithms based on Reinforcement Learning techniques poses however novel challenges related to the simulation and sensor modeling. Training of such algorithms often requires massive-scale simulations and the efficiency of the simulation and sensor models have a critical impact on the development of such systems.

While performance requirements in RL training setups suggest that the use of simple low-fidelity models, or even training the driving policies on ideal data is a desirable solution in this context, such a solution may contribute to the so-called sim-to-real gap. Differences between the simulated training environment and the real-world data streams may lead to the policy's severe performance degradation when executed in the actual vehicle.

The impact of the sensor models on the RL training, performance of the driving policy, and overall safety of the AD system remain poorly understood. In this paper, I propose a set of low-fidelity sensor models intended to imitate errors in static and dynamic environment perception and apply them in the training of RL-based driving policy. Evaluation of the policy's performance in various environments provides valuable insight into the sensor modeling's impact on RL-based policies.

2. Sensor Models

Sensor modeling approaches vary with the required accuracy, computational resources available, type of sensor, and desired interfaces. While models proposed in this publication may find use in various applications, such as testing and performance evaluation of ADAS/AD algorithms, a main intended application is in the training of RL-based driving policies. The models are intended to operate on ground truth data provided by an arbitrary simulation package and modify them to reflect the typical performance of an automotive sensor stack and perception algorithms. The models are generic (not sensor-specific) and can be calibrated to reflect an arbitrary perception algorithm.

RL training usually requires gathering a massive amount of data in the simulation environment, resulting in a demand for highly effective, scalable simulation environments. For this reason, low-fidelity sensor models that do not have high computational requirements come across as an adequate solution. While low-fidelity models are typically less accurate compared to high-fidelity alternatives, RL-based driving policies tend to feature good generalization skills and do not require as a precise imitation of error profiles as perception performance evaluation applications.

2.1. Dynamic Environment Perception

Dynamic environment perception systems typically provide information about other traffic participants in the proximity of the ego vehicle, such as other vehicles and pedestrians. In this section, I propose models that can be used for various perception systems but are mainly intended to mimic the behavior of systems with time-correlated errors and frequent false positive and false negative detection errors. A good example of such a setup is a radar-based perception system that utilizes a tracking module to produce filtered object-level detections of road vehicles.

2.1.1. Interfaces

There is no common agreement on the simulation interfaces in the automotive industry, even though a few promising standards were already proposed, most notably the Open Simulation Interface [19]. Various simulation tools offer different output data formats, depending on the type of simulation and intended applications. RL-based driving policies typically utilize object lists that encode states of individual traffic participants separately for the dynamic environment description and various types of static environment representations, such as lane lists that gather information about road geometry or lane markers.

This relatively high-level description of the vehicle's environment is easy to process and does not create excessive bandwidth or memory requirements.

The low-fidelity sensor models presented in this paper utilize an interface that describes the ego's dynamic environment as a list of dynamic objects that represent traffic participants such as vehicles. Dynamic environment at a time t in a scene composed of n_d traffic participants is represented as a set of dynamic objects $\mathbf{S_d}(t) = \left\{\mathbf{S}_{d_i}(t)\right\}_{i=1..n_d}$, where each traffic participant is described with a state vector $\mathbf{S_d}_i \in \mathbb{R}^m$, $for\ i = 1..n_d$ composed of m state variables:

$$\mathbf{S_{d_i}} = \begin{bmatrix} \mathbf{q}_i \\ \mathbf{x}_i \\ \psi_i \\ v_i \\ a_i \end{bmatrix}, \tag{1}$$

where $\mathbf{q}_i \in \mathbb{R}^2$ denotes the size of the i-th traffic participant's bounding box (length and width), $\mathbf{x}_i \in \mathbb{R}^2$ denotes its position in a Cartesian coordinates system, $\psi_i \in \mathbb{R}$ describes its rotation relative to the reference frame, $v_i \in \mathbb{R}$ denotes its absolute velocity, and $a_i \in \mathbb{R}$ denotes its absolute acceleration.

Observation of a dynamic scene representation $\mathbf{S}_d$ is generated by an arbitrary simulation package and processed by the sensor models to acquire an approximation of a perception stack's dynamic environment estimate $\hat{\mathbf{S}}_d = \left\{\hat{\mathbf{S}}_{\mathbf{d_j}}\right\}_{j=1..n_o}$, composed of n_o objects' state estimates $\hat{\mathbf{S}}_{d_j} = [\hat{\mathbf{q}}_j,\ \hat{\mathbf{x}}_j,\ \hat{v}_j,\ \hat{a}_j]^T$.

Similarly as in the notation proposed by [20], the sensing process can be described as a mapping:

$$M(\mathbf{p}) : \left\{\mathbf{S}_{d_i}\right\}_{i=1..n_d} \rightarrow \left\{\hat{\mathbf{S}}_{d_j}\right\}_{j=1..n_o}, \tag{2}$$

where $\mathbf{p}$ denotes sensor properties (calibration parameters). Note that during the sensing process certain objects may be omitted (false negative detections), while non-existing ones may be added (false positive detections), and thus, in general, $\mathbf{S}_{d_i} \not\rightarrow \hat{\mathbf{S}}_{d_i}$, and $n_d \neq n_o$.

As proposed in [20], sensing process M may be described as a series of n_m mapping operations $M^{(k)}$ for $k = 1..n_m$, yielding following sensor model:

$$M(\mathbf{p}) = M^{(n_m)}(\mathbf{p}_{n_m}) \circ ... \circ M^{(2)}(\mathbf{p}_2) \circ M^{(1)}(\mathbf{p}_1) \tag{3}$$

where $M^{(n)}$ denotes the n-th mapping operation, and $\mathbf{p}_n$ is a vector of configuration parameters relevant to the n-th mapping operation. Note that configuration parameters may include the model's outputs from previous observations if the mapping takes time correlations into account. n-th mapping operation can be defined as:

$$M^n(\mathbf{p}_n) : \{\mathbf{S_d}_i\}^{(k-1)}_{i=1..n_{k-1}} \rightarrow \left\{\mathbf{S_d}_j\right\}^{(k)}_{j=1..n_k}, \tag{4}$$

where $\{\mathbf{S_d}_i\}^{(0)}_{i=1..n_{k-1}}$ is equal to the ground-truth description of the dynamic environment $\mathbf{S}_d$, and $\left\{\mathbf{S_d}_j\right\}^{(k)}_{j=1..n_k}$ corresponds to the approximation of the sensing stack's dynamic environment's estimate $\hat{\mathbf{S}}_\mathbf{d}$, while n_k denotes the number of objects after k-th mapping operation.

2.1.2. Model of Dynamic Environment Perception Stack

The low-fidelity model of a dynamic environment perception stack is intended to imitate the main error types observed in object-detection systems used in automotive. A common example of such system is a set of short-range millimeter-wave Phase-Modulated Continuous Wave (PMCW) Radars placed in the vehicle's corners and a long-range PMCW radar in the front of the vehicle. Data from the sensors, for instance in a form of object lists, is typically fused using tracking algorithms based on derivatives of a Kalman Filter.

Errors commonly observed in such systems can be roughly assigned to the categories listed below.

- False positive detections.Various physical phenomena, such as multipath reflections of a radar wave, may result in an introduction of non-existing objects to the object list $\hat{\mathbf{S}}_d$, leading to the situation in which ADAS/AD algorithms assume the presence of potentially dangerous objects in unoccupied areas.
- False negative detections. Limited performance of the sensors, as well as performance degradations caused by difficult environmental conditions such as bad weather, may lead to missed object detections, i.e., situations in which a dynamic object present in ego's vicinity is not represented in the object list $\hat{\mathbf{S}}_d$.
- State estimation errors. Partial occlusions and performance degradations may lead to potentially dangerous differences between ground truth state description $\mathbf{S}_{d_i}$ of a particular object, and its estimate $\hat{\mathbf{S}}_{d_i}$ composed by the sensor stack. It should be noted, that due to the filtering properties of the tracking algorithm, state estimation errors may be time-correlated.

Differences between the ground truth $\mathbf{S_d}$ and the estimate $\hat{\mathbf{S}}_\mathbf{d}$ may also be caused by the range and angle limitations of the sensors, as well as occlusions.

Before modeling stochastic errors related to the sensors' performance, deterministic sensors' limitations are modeled according to the specification of the sensor stack. Objects that are outside of the sensors' detection area are removed from the ground-truth objects list. Similarly, objects that are occluded or partially occluded by other traffic participants or obstacles can be removed. These operations are denoted as $M^v(\mathbf{p}_v) : \{\mathbf{S_d}_i\}_{i=1..n_d} \rightarrow \left\{\mathbf{S_d}_j\right\}^{(v)}_{j=1..n_v}$, where calibration parameters $\mathbf{p}_v$ describe the sensing stack's detection area and the occlusion level above which the object is removed, while n_v denotes the number of the unoccluded objects in the detection area.

Two types of false negative object detections are simulated in the proposed generic sensor models. The first type is the detection delay. Due to the latencies in the vehicle's internal communication network, as well as properties of the tracking algorithms, that often need to confirm the existence of the object in a few consecutive scans before adding it to the object list, a random delay between the object entering the sensors' detection area and actually detecting it can be observed in the most of the sensor stacks. This property of the sensing system is modeled by assigning a random detection delay of T_{d_i} for $i = 1..n_v$ seconds to each object newly introduced to the $\{\mathbf{S_d}_i\}_{i=1..n_d}$ set, which value is sampled from a normal distribution:

$$T_{d_i} = \max\left(p_{d_{\mu\_delay}}, |\sim \mathcal{N}(0, p^2_{d_{\sigma\_delay}})|\right), \tag{5}$$

where p_{μ_delay} and p_{σ_delay} are calibration parameters.

Additionally, to model random losses of already detected objects, each object at each sensing update can be marked as a false-negative detection with a certain probability $p_{d\_fn\_prob}$, and assigned a $T_f = \max\left(p_{fn\_\mu}, |\sim \mathcal{N}(0, p^2_{fn\_\sigma})|\right)$ value that denotes duration for which it will remain marked as a false-negative. Parameters p_{μ_delay}, p_{σ_delay}, $p_{\text{fn\_prob}}$, p_{fn_μ}, p_{fn_σ}, as well as detection delays T_{d_i}, false-negative flags, detection durations T_f, current timestamp and timestamps at which objects were first observed or marked as false-negatives are included in the parameters vector $\mathbf{p_d}$, allowing to remove objects that are newly detected and marked as a false-negative in a following mapping:

$$M^d(\mathbf{p}_d) : \{\mathbf{S_d}_i\}^{(v)}_{i=1..n_v} \rightarrow \left\{\mathbf{S_d}_j\right\}^{(d)}_{j=1..n_d d}. \tag{6}$$

False positive detection errors, especially in radar sensors, often exhibit behavior similar to actual road users, posing a serious challenge to ADAS/AD systems. In the proposed approach, a false positive object can be introduced to the sensed objects set

with a probability $p_{\text{fp\_prob}}$. Since false positive detection errors often persist for multiple sensing updates, duration $T_{fp} = \max\left(p_{\text{fp\_}\mu}, |\sim\mathcal{N}(0, p^2_{\text{fp\_}\sigma})|\right)$ is assigned to the newly introduced false positive object, during which the object will persist in the dynamic scene approximation set. A newly created object is initialized with a random state $\mathbf{S}_{fn}$, which values are sampled from normal distributions:

$$\mathbf{S_{fp}} = \begin{bmatrix} \mathbf{q} \sim \mathcal{N}_2(\mu_q, \Sigma_q) \\ \mathbf{x} \sim \mathcal{N}_2(\mu_x, \Sigma_x) \\ \psi \sim \mathcal{N}(\mu_\psi, \sigma_\psi) \\ v \sim \mathcal{N}(\mu_v, \sigma_v) \\ a \sim \mathcal{N}(\mu_a, \sigma_a) \end{bmatrix}, \tag{7}$$

where μ_q, Σ_q, μ_x, Σ_x, μ_ψ, σ_ψ, μ_v, σ_v, μ_a, and σ_a, are the calibration parameters included in the parameters vector $\mathbf{p}_p$. All of the false positive objects are assumed to move in the direction of their orientation ψ with a constant acceleration, and thus their state (included in the parameters vector $\mathbf{p}_p$) is updated according to these assumptions at each sensing update.

Operations of false positive objects creation, update, and removal after corresponding T_{fp} are denoted as the following mapping:

$$M^p(\mathbf{p}_p) : \{\mathbf{S_d}_i\}^{(d)}_{i=1..n_{dd}} \rightarrow \left\{\mathbf{S_d}_j\right\}^{(p)}_{j=1..n_p}. \tag{8}$$

Lastly, state estimation errors are introduced. State errors in the automotive perception systems tend to be time-correlated due to the filtering properties of tracking and fusion algorithms used in them, as well as due to the persistency of environmental triggers (it is rare for an environmental trigger to impact only a single perception update). In order to reflect the randomness of the errors, as well as their time-correlated nature, a multivariate stochastic process based on the Ornstein–Uhlenbeck process is utilized for state values modeling. The value of an n_c-dimensional state estimate vector chunk $\hat{\mathbf{s}}_c$ at i-th perception update is calculated according to the process:

$$P_{OU_c}(\hat{\mathbf{s}}^{(i)}_c | \mathbf{s}^{(i)}, \hat{\mathbf{s}}^{(i-1)}, \mathbf{p}_{s_c}) = \begin{cases} \mathbf{p}_{\text{ou\_}\lambda_c} * (\mathbf{s}^(i) - \hat{\mathbf{s}}^{(i-1)}) * dt + \mathbf{W}^{(i)} * dt, & \text{for } i > 1 \\ \mathcal{N}_{n_c}(\mathbf{s}^{(i)}, \mathbf{p}_{\text{ou\_}\Sigma\text{\_init}_c}), & \text{for } i = 0, \end{cases} \tag{9}$$

where $\mathbf{s}^{(i)} \in \mathbb{R}^{n_c}$ is the ground-truth value at i-th perception update, $\hat{\mathbf{s}}^{(i-1)} \in \mathbb{R}^{n_c}$ is the previous value of state estimate, dt is the time between perception updates, and $\mathbf{W}_i \in \mathbb{R}^{n_c}$ is drawn from a multivariate normal distribution $\mathbf{W}_i \sim \mathcal{N}_{n_c}(\mathbf{0}, \mathbf{p}_{\text{ou\_}\Sigma\text{\_}u_c})$. $\mathbf{p}_{\text{ou\_}\lambda_c}$, $\mathbf{p}_{\text{ou\_}\Sigma\text{\_init}_c}$ and $\mathbf{p}_{\text{ou\_}\Sigma\text{\_}u_c}$ are the calibration parameters included in the vector $\mathbf{p}_{s_c}$.

The state of the dynamic environment is updated according to the mapping:

$$M^{\text{state\_est}}(\mathbf{p}_{\text{state\_est}}, \mathbf{S}^{(t-1)}_{\text{state\_est}}) : \{\mathbf{S_d}_i\}^{(d)}_{i=1..n_{dd}} \rightarrow \left\{\mathbf{S}^{(t)}_{\text{state\_est}_j}\right\}^{(p)}_{j=1..n_p}, \tag{10}$$

where $\mathbf{S}^{(t)}_{\text{state\_est}_j}$ is calculated using a process:

$$\mathbf{S}^{(t)}_{\text{state\_est}_j} = P_{OU}(\hat{\mathbf{S}}^{(\mathbf{t})} | \mathbf{S}^{(t)}, \hat{\mathbf{S}}^{(t-1)}, \mathbf{p}_{\text{state}}), \tag{11}$$

parameterized with a vector $\mathbf{p}_{\text{state}}$ consisting of $\mathbf{p}_{\text{ou\_}\lambda}$, $\mathbf{p}_{\text{ou\_}\Sigma\text{\_init}}$ and $\mathbf{p}_{\text{ou\_}\Sigma\text{\_}u}$ calibration matrices.

The final model can be described as the mapping:

$$M_r(\mathbf{p}) = M^{(v)}(\mathbf{p}_v) \circ M^{(d)}(\mathbf{p}_d) \circ M^{(p)}(\mathbf{p}_p) \circ M^{(\text{state\_est})}(\mathbf{p}_{\text{state\_est}}). \tag{12}$$

The proposed model, depending on the calibration parameters, may imitate an arbitrary dynamic environment perception system, including false positive and false negative

errors, as well as time-correlated state perception errors. Values of calibration parameters used in a further evaluation are provided in Appendix A.

2.2. Static Environment Perception

Features of a static environment of the ego vehicles relevant to ADAS/AD algorithms can be categorized into two main types: road features such as the geometry of the road, lane markers or barriers, and static obstacles such as parked cars, or debris. Since static obstacle detection is often based on similar sensor stack and perception algorithms as the dynamic environment perception, they can be represented using the $\mathbf{S}_d$ state vector with reasonable accuracy. Depending on the object type, $\hat{v}$ and $\hat{a}$ may be set to 0, though in certain situations allowing for non-zero velocity and acceleration may be desirable to reflect potential errors e.g., in state estimation of recently stopped or potentially moving vehicles.

The road features most relevant to RL-based driving policies are represented as a set $\mathbf{S_r}(t) = \{\mathbf{S}_{r_i}(t)\}_{i=1..n_s}$ composed of n_s l-dimensional vectors $\mathbf{S_r}_i \in \mathbb{R}^l \; for \; i = 1..n_s$ representing lane markers in the vicinity of the ego vehicle. Each vector describes a single lane marker in a following form:

$$\mathbf{S_{r_i}} = \begin{bmatrix} \mathbf{c} \\ h \end{bmatrix}, \tag{13}$$

where $\mathbf{c} \in \mathbb{R}^4$ denotes coefficients $\mathbf{c} = [c_0, \, c_1, \, c_2, \, c_3]$ of a cubic polynomial $d(s) = c_3 * s^3 + c_2 * s^2 + c_1 * s + c_0$ that encodes the lateral offset d of a lane marking (road edge or line) from the vehicle's longitudinal axis as a function of a longitudinal distance from the vehicle's rear axis, and h denotes the longitudinal distance up to which the lane marking is observed. Similarly, as in the case of the dynamic environment, the ground truth of the static scene representation $\mathbf{S}_{r_i}(t)$ is generated by a simulation package and processed by the static environment sensor models to acquire an approximation of a perception stack's static environment estimate $\hat{\mathbf{S}}_r(\mathbf{S}_r, \mathbf{p}, t)$ composed of an arbitrary number of lane markers state estimates $\hat{\mathbf{S}}_{r_i}(t) = \left[\hat{\mathbf{c}}(t), \hat{h}(t)\right]^T$.

The sensing process for the static environment can be thus modeled as a series of mapping operations:

$$M_r(\mathbf{d}) = M_r^{(n_m)}(\mathbf{d}_{n_m}) \circ ... \circ M_r^{(2)}(\mathbf{d}_2) \circ M_r^{(1)}(\mathbf{d}_1), \tag{14}$$

defined analogically to the mappings proposed for the dynamic environment sensing model.

Model of Static Environment Perception Stack

Static environment errors modeled in the proposed approach include marker length limitations (e.g., due to occlusions or detection performance limitations), false negative detections, and geometry estimation errors.

The detection range limitation model is intended to imitate the lane detection quality decrease on long distances due to the perspective, occlusions, and environmental factors such as fog or rain. It should be noted that the range limitation impacts the quality of lane markers geometry estimation - the model assumes that the lane marker geometry can be estimated solely based on the part of the lane assumed to be visible.

The model uniformly samples the ground truth lane markers geometries $\mathbf{S}_r$, mapping each lane marker definition $\mathbf{S}_{r_i}$ to a set $\{\mathbf{s}_i\}_{j=1..n_{samples}}$ of samples $\mathbf{s}_{i_j} = [x_s, y_s]^T$, where x_s and y_s are the longitudinal and lateral position of the lane marker sample in the Vehicle Coordinates System (VCS). The occlusion model is defined as a mapping:

$$M^{(\mathrm{occ})}(\mathbf{p}_{\mathrm{occ}}) : \{\mathbf{s}_i\}_{j=1..n_{\mathrm{samples}}} \rightarrow \{\mathbf{s}_i^{\mathrm{occ}}\}_{j=1..n_{\mathrm{occ}}} \tag{15}$$

evaluates whether each lane sample is occluded by any of the dynamic objects from the $\mathbf{S}_d$ set and discards them from the final samples set $\left\{\mathbf{s}_i^{\mathrm{occ}}\right\}_{j=1..n_{\mathrm{occ}}}$. If at least n_{cons} consecutive

samples in a lane marker are occluded, all of the samples placed farther from the vehicle are discarded as well.

The distance of the farthest sample from each lane marker from the set is treated as a base marker length h_{gt}. The value of the marker length is calculated similarly to dynamic objects state estimates, using a stochastic model based on the Ornstein–Uhlenbeck process:

$$P_h(\hat{h}^{(t)}|h_{gt}^{(t)}, h_{gt}^{(t-1)}, \hat{h}^{(t-1)}, \mathbf{p}_h) = \begin{cases} \left(p_{\text{lm\_ou\_}\lambda\text{\_}h}\left(\left(h_{gt}^{(t)} - p_{\text{lm\_lim}}\right) - \hat{h}^{(t-1)}\right) + W^{(t)}\right) * dt \\ \qquad \text{if } t > 0 \text{ and } \left(h_{gt}^{(t)} - h_{gt}^{(t-1)}\right) \geq p_{\text{lm\_jump}} \\ \mathcal{N}\left(h_{gt}^{(t)} - p_{\text{lm\_lim}}, p_{\sigma\text{\_}h}\right) \\ \qquad \text{otherwise,} \end{cases} \tag{16}$$

where $p_{\text{lm\_ou\_}\lambda\text{\_}h}$, $p_{\text{lm\_lim}}$, $p_{\text{lm\_jump}}$, and $p_{\sigma\text{\_}h}$ are the calibration parameters included in parameters vector $\mathbf{p}_h$. Note that if the ground truth lane length is severely decreased between the time updates, i.e., $\left(h_{gt}^{(t)} - h_{gt}^{(t-1)}\right) \geq p_{\text{lm\_jump}}$, the Ornstein-Uhlenbeck process is reset, and the length value is drawn from a normal distribution to more accurately model sudden lane occlusions.

The geometry of the lane markers is calculated using a mapping $M^{(geom)}(\mathbf{p}_c, \mathbf{p}_h)$, that calculates the geometry of lane markers as:

$$\mathbf{S}_j^{(t)} = \begin{bmatrix} P_{OU}(\hat{\mathbf{c}}^{(t)}|\mathbf{c}^{(t)}, \hat{\mathbf{c}}^{(t-1)}, \mathbf{p}_\mathbf{c}) \\ P_{OU}(\hat{h}^{(t)}|h_{gt}^{(t)}, h_{gt}^{(t-1)}, \hat{h}^{(t-1)}, \mathbf{p}_h) \end{bmatrix} \text{ for } j = 1..n_{\text{occ}}, \tag{17}$$

where $\mathbf{p}_c$ is a vector of calibration parameters, consisting of calibration matrices $\mathbf{p}_{\text{lm\_ou\_}\lambda}$, $\mathbf{p}_{\text{lm\_ou\_}\Sigma\text{\_init}}$ and $\mathbf{p}_{\text{lm\_ou\_}\Sigma\text{\_}u}$.

Finally, the false negative lane markers detection model is applied. The model is based on intuition, that lane markers with a higher lateral offset from the ego vehicle, as well as shorter (occluded) ones, have a higher probability of being not detected by the perception system. Lane offset $o \in \mathbb{N}$ denotes the number of lane markers between the ego vehicle and the given marker, where $o = 0$ corresponds to a lane marker on either the left or right side of the ego's current lane.

The mapping $M^{\text{fp}}(\mathbf{p}_{\text{fp}}) : \left\{\mathbf{S}_i^{\text{geom}}\right\}_{i=1..n_{geom}} \rightarrow \left\{\mathbf{S}_j^{\text{fp}}\right\}_{j=1..n_{\text{fp}}}$ is randomly discarding lane markers according to the probability P_{discard}:

$$P_{\text{discard}_i}(\mathbf{p}_{\text{fn}}) = \begin{cases} p_{\text{lm\_disc\_c\_0}} + p_{\text{lm\_disc\_l\_0}} * \frac{h_{\max} - h_i}{h_{\max}} & \text{if } o = 0 \\ p_{\text{lm\_disc\_c\_1}} + p_{\text{lm\_disc\_l\_1}} * \frac{h_{\max} - h_i}{h_{\max}} & \text{if } o = 1 \\ p_{\text{lm\_disc\_c\_2}} + p_{\text{lm\_disc\_l\_2}} * \frac{h_{\max} - h_i}{h_{\max}} & \text{otherwise,} \end{cases} \quad \text{for } i = 1..n_{\text{geom}}, \tag{18}$$

where $p_{\text{lm\_disc\_c\_0}}$, $p_{\text{lm\_disc\_l\_0}}$, $p_{\text{lm\_disc\_c\_1}}$, $p_{\text{lm\_disc\_l\_1}}$, $p_{\text{lm\_disc\_c\_2}}$, and $p_{\text{lm\_disc\_l\_2}}$ are the calibration parameters included in the parameters vector $\mathbf{p}_{\text{fn}}$, and $h_{\max}$ denotes the maximum length of the lane marker that can be detected by the modeled perception system.

Discarded lane markers remain false negatives, with a probability $P_{\text{lm\_recovery}}$ of returning to their true positive state on each perception update:

$$P_{\text{lm\_recovery}} = p_{\text{rec\_hyst}} * (1 - P_{\text{discard}}) + \min(p_{\text{rec\_pps}} * t_{\text{disc}}, p_{\text{rec\_sat}}), \tag{19}$$

where t_{disc} is the duration of the false negative, and $p_{\text{rec\_hyst}}$, $p_{\text{rec\_pps}}$, $p_{\text{rec\_sat}}$ are the calibration parameters.

The lane markers perception system is thus modeled as a mapping:

$$M_{\text{static}}(\mathbf{p}) = M^{(\text{occ})}(\mathbf{p}_{\text{occ}}) \circ M^{(\text{geom})}(\mathbf{p}_{\text{geom}}) \circ M^{(\text{fn})}(\mathbf{p}_{\text{fn}}). \tag{20}$$

3. Driving Policy

Autonomous Driving (AD) tasks are known to be challenging due to the imperfect environment perception, uncertainty around the future actions of other road users, as well as comfort and performance goals that the AD vehicle is expected to fulfill. Since deterministic rule-based control methods rarely can achieve the required performance in such difficult environments, Reinforcement Learning (RL) approaches are a frequently proposed alternative [21].

In a typical RL setup for the AD applications, a traffic simulator in which a virtual ego vehicle can interact with the surrounding road users and infrastructure is described as an environment ϵ, that maps the agent's actions a_t to the environment's transition from a current state s_t to the new state s_{t+1}. Actions a are selected by the agent's stochastic policy $\pi_\theta(a|o)$ parametrized by parameters vector θ based on an environment's observation o_t.

3.1. Proximal Policy Optimization

Proximal Policy Optimization (PPO)[22] is an on-policy RL algorithm commonly used for training deep neural networks used for control tasks. The parameters update in the PPO is performed according to the following equation:

$$\theta_{k+1} = \arg\max_{\theta} \hat{\mathbb{E}}_{\theta \sim \pi_{\theta_k}} [\mathcal{L}(s, a, \theta_k, \theta)], \tag{21}$$

where $\mathcal{L}(s, a, \theta_k, \theta)$ is a clipped loss function defined as:

$$\mathcal{L}(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t, s_t)} \hat{A}^{\pi_{\theta_k}}(s, a), \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t, s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}^{\pi_{\theta_k}}(s, a) \right), \tag{22}$$

where $\pi_\theta(a_t|s_t)$ is the action probability under the new policy, $\pi_{\theta_k}(s_t|s_t)$ is the probability under the current policy, $\hat{A}^{\pi_{\theta_k}}(s, a)$ is the estimated advantage at the time t, and ϵ is a hyperparameter. Advantage estimation is performed by a Generalized Advantage Estimator (GAE) [23] in form $\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l (r_{t+l} + \gamma V_\phi(s_{t+l+1}) - V_\phi(s_{t+l}))$, where λ and γ are calibration parameters, r_t denotes the reward at time t, and $V\phi(s)$ is an estimate of the value function, performed by a learned neural network (critic network) parametrized by a parameter vector ϕ. Advantage estimation is based on a set $\mathcal{D}_k = \{\tau_i\}_{i=1..n_{episodes}}$ of trajectory segments $\tau_i = \{s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, ...\}$ collected at each iteration by observing the interaction of the policy π_{θ_k} with the environment e, typically in a parallelized simulation setup.

3.2. Direct Control Policy

Both observations and actions chosen by the policy may take various forms. Observation typically includes the state of the ego and the environment description, either in a form of grid maps or object vectors. Output actions may select the high-level semantic actions (such as the "change lane" action), describe the ego's trajectory followed by a deterministic controller, or control the vehicle's actuators directly.

For the purpose of studying the perception errors impact on the performance of the driving policies, I focus on the direct control network with object-lists inputs, due to its lightweight implementation and high impact of individual actions on the agent's performance.

Inputs to the network are composed of three main sets described below.

- Ego state observed at the time t defined as $\mathbf{o}_{ego}^{(t)} = \left[v_{s_e}^{(t)}, v_{ex_e}^{(t)}, a_{s_e}^{(t)}, a_{s_e}^{(t-1)}, \gamma_e^{(t)} \right]$, where $v_{s_e}^{(t)}$ is a current longitudinal velocity, $v_{ex_e}^{(t)}$ is a speed limit execution, defined as ratio of $v_{s_e}^{(t)}$ to the current speed limit, $a_{s_e}^{(t)}$ denotes current longitudinal acceleration, $a_{s_e}^{(t-1)}$ describes the acceleration at the previous time update, and $\gamma_e^{(t)}$ is the ratio of the current yaw rate to the absolute velocity.

- Other road users state, where each vehicle perceived by the ego is described with a vector $\mathbf{o}_{obj_i} = [\mathbf{q}_i, \mathbf{x}_i, \psi_i, \mathbf{v}_i, \mathbf{a}_i]^T$ for $i = 1..n_{\text{obj\_max}}$, where $\mathbf{q_i} \in \mathbb{R}^2$ is the width and length of the vehicle, $\mathbf{x_i} \in \mathbb{R}^2$ its position relative to the ego vehicle, $\psi_i \in \mathbb{R}$ rotation with respect to the ego, $\mathbf{v_i} = [v_{s_i}, v_{d_i}]^T$ denotes the vehicle's velocity relative to the ego, and $\mathbf{a_i} \in \mathbb{R}^2$ its relative acceleration. Depending on a setup, the observation is created based on the sensor models' output, or the ground truth data.
- Lane markers state, where each lane marker registered by the ego's perception system is encoded with a vector $\mathbf{o}_{\text{lm}_i} = \left[\mathbf{d}_{\text{lm}_i}, h_{\text{lm}_i}, \gamma_{\text{lm}_i}, m_{\text{lm}_i}\right]^T$, where $\mathbf{d}_{\text{lm}_i} \in \mathbb{R}^{10}$ is a vector of uniformly placed lateral position samples that describe the lane marker's geometry, $h_{\text{lm}_i} \in \mathbb{R}$ is the observed length of the marker, $\gamma_{\text{lm}_i} \in \mathbb{R}$ is the marker's rotation at the point adjacent to the ego's position, and $m_{\text{lm}_i} \in [0, 1]$ encodes the marker type, where $m_{lm_i} = 0$ if marker is a broken line, and $m_{\text{lm}_i} = 1$ otherwise.

All of the observation values are normalized and processed by the transformer input model, which encodes them as shown in Figure 1 and passed to a deep fully connected network. The network returns output vector $a_\phi(\mathbf{o}_{\text{ego}}, \mathbf{o}_{\text{obj}}, \mathbf{o}_{\text{lm}}) = [a_{\text{acc}}, a_{\text{steer}}]^T$ of values ranged $[-1, 1]$ that control the acceleration and steering angle of the ego vehicle. Both values are scaled by calibration parameters $p_{a_{\text{acc}}}, p_{a_{\text{steer}}}$ accordingly.

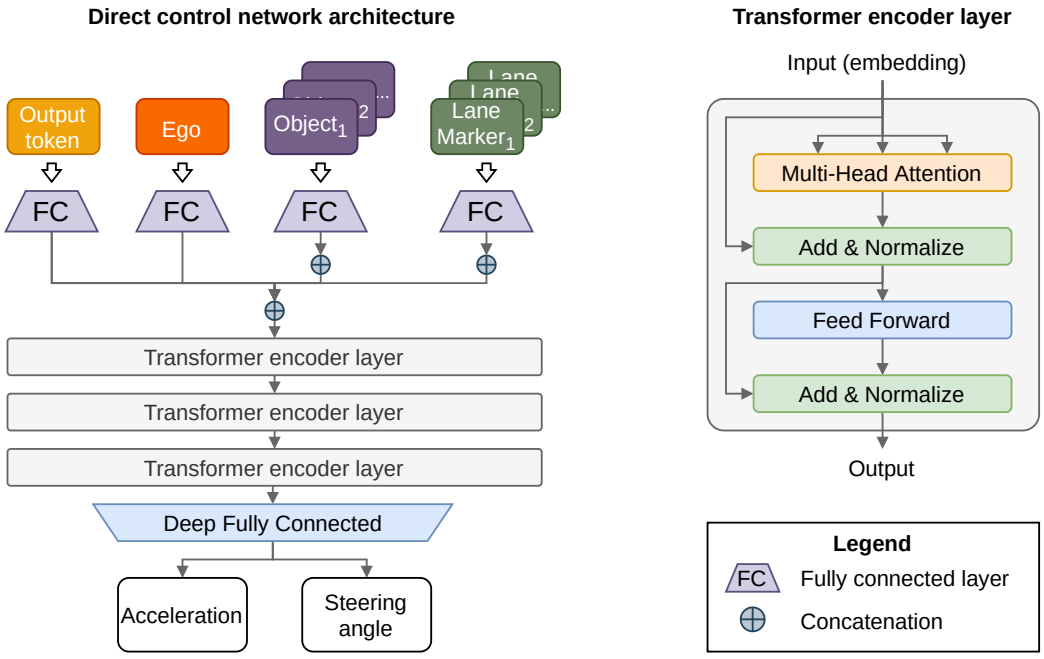

Figure 1. Architecture of the direct control network. n_{embd}-dimensional input embeddings of a const output token, Ego features ($\mathbf{o}_{\text{ego}}$), objects $\mathbf{o}_{\text{obj}_i}$ for $i = 1..n_{\text{max\_obj}}$, and lane markers $\mathbf{o}_{lm_i}$ for $i = 1..n_{\text{max\_lm}}$ are concatenated into matrix $I \in R^{n_{embd} \times (1+1+n_{\text{obj}}+n_{\text{lm}})}$ and consumed by the transformer encoder layers. Encoders use a masking mechanism to prevent non-existing objects or lane markers from impacting the outputs. Ultimately transformer encoder layers produce the output $O \in R^{n_{embd} \times (1+1+n_{\text{obj}}+n_{\text{lm}})}$. The first column of O is then processed by a deep fully connected network to generate the control values a_{acc} and a_{steer}. Note that the transformer structure used for observation lacks positional encodings, as there is no need for sorting or prioritizing the environmental features.

3.3. Rewards

The control policy described in the previous section is trained with a PPO algorithm with a reward function composed of several terms listed below.

- Speed limit execution calculated at each step as a ratio of ego's current velocity to a current speed limit, multiplied by a factor $r_{\text{speed\_limit}}$.

- Action values, specifically a squared acceleration and squared steering angle values, scaled by factors r_{acc} and r_{steer} respectively.
- Lane centering defined as a current distance of the ego vehicle's center from the lane center multiplied by a factor $r_{centering}$.
- Time To Collision calculated as a time at which a collision between the ego and another road user would collide if neither of them would change their current longitudinal acceleration nor lateral position. If constant accelerations would not lead to a collision or the time to collision is lower than r_{ttc_max}, the value for this reward component is set to 0, otherwise, TTC scaled by r_{TTC} factor is assigned.
- Terminal states reward component assigned in events of a collision between the ego and other road user or a road barrier and exceeding the speed limit by 10 m/s or more.

3.4. Training Setup

The experimental setup used for the evaluation of the proposed methods consisted of the simulation environment representing a randomly-generated multiple-lane highway featuring merge-in lanes, exit lanes, and vehicle traffic of varied density (see Figure 2). The movement of the road users (except for the ego vehicle) has been governed by a proprietary simulation package TrafficAI with rule-based semi-random driving policies. Simulation has been updated in 0.05 s timesteps, with the ego's control values updated at every two steps. The direct control policy described in a previous chapter was trained for roughly 24 h in a distributed computing setup with 100 simulation threads used for data collection.

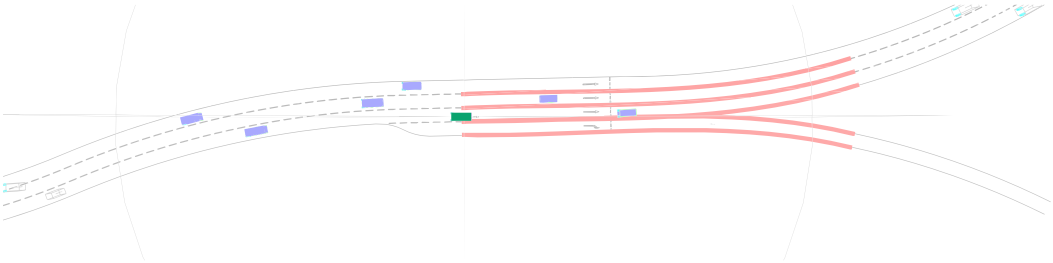

Figure 2. Simulation environment. Training and evaluations were performed in the simulation of a randomly-generated multi-lane highway. Ground truth vehicles and road (light gray on the figure) are parsed by the sensor models to produce vehicle state estimations (in blue) and lane markers geometry model (in red). The ego vehicle (green) is controlled by the direct control policy.

4. Evaluation Setup

Evaluation of the proposed sensor models in the context of the Reinforcement Learning policy training task comes with several challenges. Since one of the main purposes of the RL-based driving policy is to govern interactions with other road users, the end-to-end evaluation typically requires a closed-loop setup. On-road testing, while most informative, poses severe collision risks due to inherent limitations of the policy trained in a ground-truth environment.

Certain open-loop tests could be performed to assess policy's robustness, e.g, through the comparison of the actions chosen based on ground-truth data to the sensor-based actions, similarly to Probably Approximate Correct sensing system evaluation methodology described in [24]. Since models proposed in this article focus mostly on object-level radar-based sensing systems, this method is unfortunately not feasible due to the lack of open datasets with radar-based object detection and tracking outputs.

Considering these limitations, I evaluate the policy's performance in two ways: through large-scale driving tests in the simulation environment, and in the distribution of pre-defined test scenarios. In order to provide an insight into the impact of the sensor model's use in the training process on the end policy performance, an additional set of baseline sensor models (described in the next subsection) is introduced for testing and comparison purposes. Policies trained in environments with both types of sensor models

(proposed and baseline), as well as in the ground-truth environment, are cross-tested in all three types of training environments.

4.1. Baseline Sensor Models

A set of baseline sensor models is used to evaluate the impact of the particular sensor model's design on the end policy's performance, as well as to enable comparison of the policy trained in the ground-truth environment to the policy trained with the proposed sensor model in a relatively independent experimental setup. Baseline models fulfill the following set of tasks:

- introduction of state estimation errors through the addition of Gaussian noise,
- limitation of the observed lane markers distance to a value drawn from a normal distribution,
- simulation of the false negative object detection errors by random assignment of the binary visibility flag at each timestep,
- simulation of the false positive object detections through the creation of single-timestep objects with normally distributed state values,
- disturbance of the observed lane markers geometry performed through adding the Gaussian noise to the coefficients of the lane markers polynomials.

The parameterization of the models is described in Table A5.

4.2. Test Scenarios

In order to acquire an in-depth understanding of how various error patterns impact the trained policies, they are additionally evaluated in a set of short test scenarios. Each scenario is defined by a set of parameters that are drawn from pre-defined distributions. This approach allows running evaluation over a large distribution of scenarios of a given type and gathering statistical information about the agents' performance.

Test scenarios incorporate common error patterns such as late detection of a vehicle, state estimation errors, and false negative detection errors. Scenarios parameter distributions are defined in a way that makes scenarios relatively challenging, and the performance is evaluated by counting a fraction of scenarios that ended in a collision with other vehicles or a road barrier.

Each of the evaluated policies is tested in 100 sampled variants of each scenario class, providing an insight into agents' robustness to different types of errors.

A detailed description of the test scenarios with the parameter distributions is provided in Appendix B.

4.3. Evaluated Policies

Training has been performed in three setups: one with the sensor models described in a previous chapter (denoted Ornstein–Uhlenbeck-based sensor models environment, or OU-SM), one with baseline sensor models (denoted Gaussian-based sensor models environment, or G-SM), and one with ground-truth data (GT environment). All three trainings were performed with identical values of training hyperparameters and simulation parameters, listed in Table A4.

5. Results

Progress of the training in all three environments (ground-truth (GT), with baseline Gaussian-based sensor models (G-SM), and with sensor models proposed in previous chapters (OU-SM)) is presented in Figure 3. In all cases, the training progressed in a similar manner, showing a rapid increase in the terminal states' reward value, followed by a fast increase in speed limit execution value, and a long phase of entropy decrease, resulting in a slow improvement of rewards related to steering angle and acceleration. The final values of all rewards are relatively close, with a noticeable advantage of the agent in the GT environment.

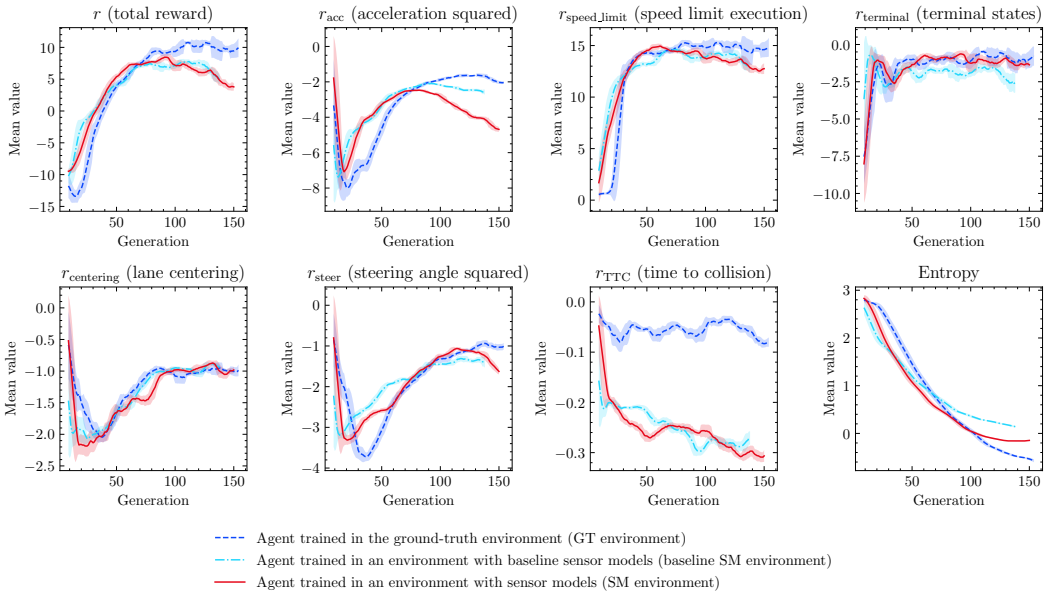

Figure 3. Training progress. Training was performed separately in environments with the described sensor models based on Ornstein–Uhlenbeck processes (OU-SM), baseline sensor models that utilize Gaussian noise (G-SM), and with the ground-truth sensor data (GT). Training in all environments progressed similarly, with the agent trained in the OU-SM environment reaching a slightly lower performance.

Evaluation of the trained agents was performed in the highway simulation environment by running 100 simulation episodes, terminated after a collision, or reaching 1000 simulation steps. The evaluation was performed in nine experimental setups, testing the performance of each of the agents (trained in GT environment, G-SM environment, and OU-SM environment) in each of the three training environments.

A set of Key Performance Indicators (KPIs) was calculated based on performed evaluations, including the mean length of the episode (while the maximum length is 1000 simulation steps, an episode can be terminated early due to collisions), the average number of heavy braking events (defined as situations in which the agent applies acceleration below 2.0 $\frac{\mathrm{m}}{\mathrm{s}}$), a fraction of episodes failed (terminated before reaching 1000 simulation steps due to collision with other vehicles or road barrier), and other indicators. A full list of the KPIs with the values evaluated in all experimental setups is presented in Table 1.

Table 1. Key Performance Indicators for evaluated agents. The table summarizes the evaluation of three driving policies (GT agent - policy trained in the ground-truth environment, G-SM agent - policy trained in the environment with baseline sensor models based on Gaussian noise, and OU-SM agent, which was trained in an environment with sensor models proposed in previous chapters). Each policy was evaluated in three simulation environments - with GT data, with G-SM setup, and with OU-SM setup.

Performance Indicator	**GT Agent**			**G-SM Agent**			**OU-SM Agent**		
	in GT env	**in G-SM env**	**in OU-SM env**	**in GT env**	**in G-SM env**	**in OU-SM env**	**in GT env**	**in G-SM env**	**in OU-SM env**
Mean episode length (sim steps)	977.4	716.8	291.5	925.8	927.2	804.0	907.6	941.8	910.8
Average speed [$\frac{\mathrm{m}}{\mathrm{s}}$]	27.6	27.9	26.0	29.1	29.4	29.0	26.0	26.2	27.1
Average abs steering angle [rad]	0.35	0.56	0.30	0.54	0.57	0.62	0.41	0.46	0.48
Average abs acceleration [$\frac{\mathrm{m}}{\mathrm{s}^2}$]	0.64	0.91	1.23	0.68	0.76	0.82	0.98	0.91	0.99
Heavy braking events	1.7	8.7	1.5	2.9	7.6	4.1	2.7	8.2	3.5
Fraction of episodes failed	0.03	0.48	0.84	0.04	0.07	0.27	0.03	0.02	0.03

Trained policies were additionally evaluated in a set of test scenarios described in Appendix B. Results of the evaluation are presented in Table 2, with a performance expressed in a fraction of the scenario episodes that ended in a collision of the ego vehicle with a road barrier or other vehicle. Note that the parameters of the scenarios are drawn from random distributions, and a certain subset of scenarios may incorporate situations, in which a collision is unavoidable.

Table 2. Performance of the agents in test scenarios. Each agent was evaluated in each scenario type 100 times, where scenarios for each run were sampled from the distributions described in Appendix B. Performance in the table is measured by a fraction of sampled scenarios evaluations that ended in collisions (the lower the better).

Scenario Type	GT Agent	G-SM Agent	OU-SM Agent
A. Late detection of a slow-moving object in front, empty highway	0.16	0.23	0.06
B. A constant error of front object's speed estimation	0.41	0.73	0.31
C. Normally distributed error of front object's speed estimation	0.45	0.30	0.24
D. Normally distributed front object's lateral position estimation error	0.26	0.37	0.12
E. Random occurrences of false negative detection errors of front object	0.12	0.14	0.06
F. Frequent false negative road markers detections	0.11	0.06	0.04

6. Discussion

An agent trained in the ground-truth (GT) environment achieves satisfactory performance in the randomly-generated GT evaluation episodes. The ego controlled by the trained policy reaches the speed limit in a smooth manner and is able to keep constant velocity on an empty road. The presence of the slower-moving vehicles results in an expected velocity decrease, where the agent adjusts its velocity to a vehicle in front of it, keeping a moderate distance from it. The agent typically moves near the lane center, sporadically performing a lane-change maneuver, e.g., when the front vehicle moves with a velocity significantly below the speed limit.

As shown in Table 1, evaluation in the environment with proposed sensor models (OU-SM environment) results in a significantly lowered performance. The presence of the sensor errors results in increased absolute accelerations, due to frequently observed late sudden breaking, and lowered overall speed. The most significant issue, however, is related to collisions with the road barriers, reflected in significantly lowered mean episode length. Analysis of the evaluation episodes shows three major situations in which the agent is prone to road barrier collisions, listed below.

- Lane geometry errors in absence of nearby vehicles. Even minor geometry errors frequently result in situations, where the agent drives close to the side of the road, triggering the road barrier collision terminal state due to touching the road barrier.
- Late detection of the vehicle in front. Since the situation in which a slow-moving vehicle appears in close proximity to the ego cannot be observed in the ground-truth environment, where the vehicle is observed as soon as it enters the detection area, the driving policy is not trained to handle such situations properly. Late detection typically results in a severe steering maneuver in an attempt to avoid the collision. Excessive control values applied to achieve this however result in a sharp turn, causing a severe collision with a road barrier.
- False-positive object detections. False positives appearing in front of the ego vehicle result in behaviors similar to the ones described in a previous point. The ego attempts to avoid the collision through a severe steering maneuver, crashing into a road barrier

due to an excessively sharp turn. Interestingly, false-positive objects that appear outside the road (behind a road barrier) also seem to destabilize the control policy - often triggering unexpected steering maneuvers that result in a collision.

An agent trained with an environment with baseline sensor models (G-SM) demonstrates significantly improved robustness to such issues, although it still does not achieve satisfactory performance in terms of collision avoidance, with 27% episodes in OU-SM environment ending prematurely due to a collision.

The use of the sensor models in the training alleviates described issues. The performance of the driving policy trained this way is slightly lower compared to the policy trained in GT and G-SM environments, likely due to more cautious behaviors, such as keeping larger distances from other vehicles. The overall behavior of the OU-SM agent however remains similar to the GT and G-SM agents evaluated in their respective training environments. Interestingly, the OU-SM policy tested in the G-SM environment slightly exceeds the G-SM policy's performance in aspects related to collision avoidance. This may be due to more cautious behaviors learned in a response to long-lasting time-correlated errors observed in the OU-SM environment.

Analysis of the test scenarios evaluation results leads to similar conclusions as the evaluation in the highway driving environments. The OU-SM agent demonstrates increased robustness to common error patterns, such as late detections and state estimation errors, compared to the policies trained in GT and G-SM environments.

The difference in the performance is especially visible in cases of long-lasting errors, such as constant velocity estimation error present in *Scenario B*. The presence of time-correlated state estimation errors in the OU-SM environment likely prevented the policy from overly relying on a small subset of observed environmental features, and the resulting agent seems to perform reasonably well even if one of the state parameters is severely disturbed.

Interestingly, in several test scenarios agents trained in the GT environment achieved better performance compared to the G-SM environment. This may be caused by the G-SM policy's tendency to avoid severe actions. Since the observation in G-SM frequently changes due to introduced state estimation noise, quick responses in form of large accelerations and/or steering angles would lead to significantly lower reward values in the training. The resulting policy is thus unable to quickly react to dangerous situations, leading to poor performance in challenging test scenarios. Both GT and OU-SM policies are able to react to sudden risks with more appropriately severe actions, executing harsh braking and steering maneuvers to avoid collisions.

7. Conclusions

Performed experiments allow drawing several conclusions related to the robustness of driving policies based on Reinforcement Learning and the impact of sensor errors on their training and performance.

Reinforcement Learning (RL) is often proposed as a candidate for driving policies training due to its good generalization capabilities and robustness to slight variations in the environment. Experiments performed in this study seem to partially confirm the generalization capabilities of RL-based policies, as the agent trained in the GT environment is able to navigate in the traffic environment, even if the observation is severely disturbed by the introduced sensor models. Nonetheless, the experiments with sensor models expose several safety hazards caused by the sim-to-real gap in the policies trained in GT environments, such as the tendency to overreact in events of late detection or false positive detection errors.

Relatively good performance of the agent trained in GT environments may result in a false sense of safety - so it is especially important to ensure the presence of the additional safety mechanisms and to extensively test all machine-learning-based driving policies in challenging environments. While the errors introduced in the SM environment in this study were severe enough to expose the hazards, realistic sensor errors observed in good

weather and lighting conditions may not be sufficient to trigger safety-critical errors in the evaluation with real-world data. Possible ways to alleviate this issue include testing in various SM environments, or the use of automatically-generated adversarial test scenarios to ensure good coverage of edge cases [25].

High-level sensor models proposed in this publication may be utilized for both testing and training driving policies. Policy trained in the environment with the sensor models is able to achieve performance levels similar to ones trained in GT environments while providing robustness against false positive and false negative object detection errors, object and lane markers state estimation errors, as well as false negative lane markers detection errors.

Funding: Industrial PhD carried out at the AGH University of Science and Technology realized in cooperation with Aptiv Services Poland S.A.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Calibration Parameters

Table A1. Values of sensor models calibration parameters used in the experimental setup.

Parameter Name	Value	Unit	Description
p_{μ_delay}	0.3	s	Mean object detection delay
p_{σ_delay}	0.55	s	Object detection delay standard deviation
$p_{\text{fn\_prob}}$	0.001	-	Probability of false negative object detection
p_{fn_μ}	1.47	s	Mean duration of false negative object detection
p_{fn_σ}	1.5	s	False negative object detection duration standard deviation
$p_{\text{fp\_prob}}$	0.0175	-	Probability of false positive object detection
p_{fp_μ}	0.5	s	Mean duration of false positive object detection
p_{fp_σ}	2.8	s	False positive object detection duration standard deviation
μ_q	$[4.34, 1.89]^T$	-	Mean false positive object size
Σ_q	$\begin{bmatrix} 0.21 & 0 \\ 0 & 0.01 \end{bmatrix}$	-	False positive object size covariance matrix
μ_x	$[45.1, 0]^T$	-	Mean false positive object position
Σ_x	$\begin{bmatrix} 19.3 & 0 \\ 0 & 0.97 \end{bmatrix}$	-	False positive object position covariance matrix
μ_ψ	0.0	-	Mean rotation of false positive object
σ_ψ	0.44	-	Standard deviation of false positive object rotation
μ_v	0.0	$\frac{\text{m}}{\text{s}}$	Mean speed of false positive object relative to ego
σ_v	11.7	$\frac{\text{m}}{\text{s}}$	Standard deviation of false positive object speed
μ_a	0.0	$\frac{\text{m}}{\text{s}^2}$	Mean acceleration of false positive object relative to ego
σ_a	3.46	$\frac{\text{m}}{\text{s}^2}$	Standard deviation of false positive object speed
$\mathbf{p}_{\text{ou}\_\lambda}$	diag(0.5, 0.65, 0.11, 0.45, 0, 0.5, 0)	-	State estimation noise parameter
$\mathbf{p}_{\text{ou}\_\Sigma\_\text{init}_i}$	diag(1.3, 1.0, 1.4, 0.7, 0, 2.2, 0)	-	State estimation noise parameter
$\mathbf{p}_{\text{ou}\_\Sigma\_u}$	diag(2.0, 1.6, 1.3, 0.7, 0, 2.5, 0)	-	State estimation noise parameter
$p_{\text{lm\_ou}\_\lambda\_h}$	0.4	-	Lane markers length noise parameter
$p_{\text{lm\_lim}}$	5.0	m	Mean lane markers length shortening
$p_{\text{lm\_jump}}$	15.0	m	Min lane markers length change for noise reset
p_{σ_h}	5.6	-	Lane markers length noise parameter
$p_{\text{lm\_disc\_c\_0}}$	0.001	-	Marker false negative probability parameter
$p_{\text{lm\_disc\_l\_0}}$	0.01	-	Marker false negative probability parameter
$p_{\text{lm\_disc\_c\_1}}$	0.01	-	Marker false negative probability parameter
$p_{\text{lm\_disc\_l\_1}}$	0.01	-	Marker false negative probability parameter
$p_{\text{lm\_disc\_c\_2}}$	0.02	-	Marker false negative probability parameter
$p_{\text{lm\_disc\_l\_2}}$	0.01	-	Marker false negative probability parameter
h_{max}	90.0	m	Maximum length of lane marker
$p_{\text{rec\_hyst}}$	0.005	-	Marker false negative recovery probability parameter

Table A1. *Cont.*

Parameter Name	Value	Unit	Description
$p_{\text{rec\_pps}}$	0.05	-	Marker false negative recovery probability parameter
$p_{\text{rec\_sat}}$	0.3	-	Marker false negative recovery probability parameter
$\mathbf{p}_{\text{lm\_ou\_}\lambda}$	diag(5.5, 5.5, 1.5, 2.5)	-	Lane marker geometry noise parameter
$\mathbf{p}_{\text{lm\_ou\_}\Sigma\text{\_init}}$	diag(2.5, 0.05, 0.001, 0.0001)	-	Lane marker geometry noise parameter
$\mathbf{p}_{\text{lm\_ou\_}\Sigma\text{\_}u}$	diag(0.15, 0.007, 10^{-4}, 10^{-6})	-	Lane marker geometry noise parameter

Table A2. Values of parameters related to direct control network and its inputs/outputs.

Parameter Name	Value	Description
$p_{a_{acc}}$	3.5	Acceleration output scaling.
$p_{a_{steer}}$	0.125	Steering output scaling.
$n_{\text{max\_obj}}$	10	Max number of observed objects.
$n_{\text{max\_lm}}$	6	Max number of observed lane markers.

Table A3. Values of reward components weights for the direct control driving policy training.

Parameter Name	Value	Description
$r_{\text{speed\_limit}}$	0.04	Speed limit execution squared.
r_{acc}	-0.003	Ego acceleration squared.
r_{steer}	-1.0	Steering angle squared.
$r_{\text{centering}}$	-0.006	Lane centering.
$r_{\text{TTC\_max}}$	6.0	Max Time To Collision to be included in reward in m/s.
r_{TTC}	-0.01	Time to collision (inversed).
r_{terminal}	-10.0	Terminal states (collisions, speed limit violations.)

Table A4. Proximal Policy Optimization training hyperparameters.

Hyperparameter	Value
Train batch size	250,000
Minibatch size	5000
Num epochs	15
Discount (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.3
KL coefficient	0
Entropy coefficient	0
VF coefficient	1.0

Table A5. Parameters of baseline sensor models (G-SM).

Parameter	Value
Object position error covariance matrix	$\begin{bmatrix} 1.2 & 0 \\ 0 & 0.7 \end{bmatrix}$
Velocity error variance	2.0
Object length error variance	0.5
Object length error lower limit	-1.0
Object width error variance	0.5
Object width error lower limit	-1.0
False positive detection probability [1]	0.0575
False negative detection probability [2]	0.1
Lane marker mean observed length	87.0
Lane marker observed length variance	5.0
Lane marker observed length upper limit	90.0
Lane marker coefficients errors covariance matrix	diag(0.005, 0.0005, 0.00005, 0.000005)

[1] Parameters of the false positive object detections are drawn from the same distributions as described in Table A1.
[2] Duration of the false negatives is fixed to a single timestep.

Appendix B. Test Scenarios

Appendix B.1. Scenario A: Late detection of a slow-moving object in front, empty highway

The ego vehicle is placed on a random lane of a straight two-lane highway. After 2 s of the simulation, the object appears in front of the ego.

Table A6. Distributions of Scenario A parameters.

Parameter	Value
Ego's initial velocity $[\frac{\mathrm{m}}{\mathrm{s}}]$	$\mathcal{U}(20.0, 30.0)$
Object's initial relative longitudinal position [m]	$\mathcal{N}(30.0, 3.0^2)$
Object's initial relative lateral position [m]	$\mathcal{N}(0.0, 0.3^2)$
Object's initial absolute speed $[\frac{\mathrm{m}}{\mathrm{s}}]$	$\mathcal{N}(10.0, 2.0^2)$
Object's acceleration $[\frac{\mathrm{m}}{\mathrm{s}^2}]$	$\mathcal{N}(0.0, 1.0^2)$.

Appendix B.2. Scenario B: Constant error of front object's speed estimation

The ego vehicle is placed on a random lane of a straight three-lane highway, with a slower-moving vehicle placed in front of it. Velocity estimation performed by the ego's perception algorithm is disturbed by a constant value (object is perceived to drive faster), while other state values remain accurate.

Table A7. Distributions of Scenario B parameters.

Parameter	Value
Ego's initial velocity $[\frac{\mathrm{m}}{\mathrm{s}}]$	$\mathcal{U}(20.0, 30.0)$
Object's initial relative longitudinal position [m]	$\mathcal{N}(40.0, 3.0^2)$
Object's initial relative lateral position [m]	$\mathcal{N}(0.0, 0.3^2)$
Object's initial absolute speed $[\frac{\mathrm{m}}{\mathrm{s}}]$	$\mathcal{N}(10.0, 2.0^2)$
Object's acceleration $[\frac{\mathrm{m}}{\mathrm{s}^2}]$	$\mathcal{N}(0.0, 1.0^2)$.
Constant velocity estimation error $[\frac{\mathrm{m}}{\mathrm{s}}]$	20

Appendix B.3. Scenario C: Normally distributed error of front object's speed estimation

The ego vehicle is placed on a random lane of a straight three-lane highway, with a slower-moving vehicle placed in front of it. Velocity estimation performed by the ego's perception algorithm is disturbed by a value sampled at each time step from a normal distribution, while other state values remain accurate.

Table A8. Distributions of Scenario C parameters. Ego and object initial state parameters are identical as in Scenario B.

Parameter	Value
Velocity estimation error $[\frac{\mathrm{m}}{\mathrm{s}}]$	$\mathcal{N}(0.0, 10.0^2)$

Appendix B.4. Scenario D: Normally distributed front object's lateral position estimation error

The ego vehicle is placed on a random lane of a straight three-lane highway, with a slower-moving vehicle placed in front of it. Lateral position estimation performed by the ego's perception algorithm is disturbed by a value sampled at each time step from a normal distribution, while other state values remain accurate.

Table A9. Distributions of Scenario D parameters. Ego and object initial state parameters are identical as in Scenario B.

Parameter	Value
Lateral position estimation error $[\frac{\mathrm{m}}{\mathrm{s}}]$	$\mathcal{N}(0.0, 3.5^2)$

Appendix B.5. Scenario E: Random occurrences of false negative detection errors of front objects

The ego vehicle is placed on a random lane of a straight three-lane highway, with a slower-moving vehicle placed in front of it. Front object perception is impacted by a randomly occurring false negative detection error.

Table A10. Distributions of Scenario E parameters. Ego and object initial state parameters are identical as in Scenario B.

Parameter	Value
Probability of false negative detection occurrence at each time step.	0.1
Duration of false negative detection error events. [s]	0.2

Appendix B.6. Scenario F: Frequent false negative road markers detection errors

The ego vehicle is placed on a random lane of an empty straight three-lane highway. Lane markers can be randomly removed (treated as false-negative detection errors).

Table A11. Distributions of Scenario F parameters. Ego initial state parameters are identical as in Scenario B.

Parameter	Value
Probability of lane marker false negative detection.	0.4

References

1. Kalra, N.; Paddock, S.M. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transp. Res. Part A: Policy Pract.* **2016**, *94*, 182–193. https://doi.org/10.1016/j.tra.2016.09.010.
2. Chao, Q.; Bi, H.; Li, W.; Mao, T.; Wang, Z.; Lin, M.C.; Deng, Z. A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving. In *Computer Graphics Forum*; Wiley Online Library, Hoboken, NJ, USA, 2019; Volume 39, pp. 287–308.
3. Shalev-Shwartz, S.; Shammah, S.; Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv* **2016**, arXiv:1610.03295.
4. Slavik, Z.; Mishra, K.V. Phenomenological modeling of millimeter-wave automotive radar. In Proceedings of the 2019 URSI Asia-Pacific Radio Science Conference (AP-RASC), New Delhi, India, 9–15 March 2019. https://doi.org/10.23919/URSIAP-RASC.2019.8738137.
5. Hirsenkorn, N.; Subkowski, P.; Hanke, T.; Schaermann, A.; Rauch, A.; Rasshofer, R.; Biebl, E. A ray launching approach for modeling an FMCW radar system. In Proceedings of the 2017 18th International Radar Symposium (IRS), Prague, Czech Republic, 28–30 June 2017; pp. 1–10. https://doi.org/10.23919/IRS.2017.8008120.

6. Jasiński, M. A Generic Validation Scheme for real-time capable Automotive Radar Sensor Models integrated into an Autonomous Driving Simulator. In Proceedings of the 2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 26–29 August 2019; pp. 612–617. https://doi.org/10.1109/MMAR.2019.8864669.
7. Schuler, K.; Becker, D.; Wiesbeck, W. Extraction of Virtual Scattering Centers of Vehicles by Ray-Tracing Simulations. *IEEE Trans. Antennas Propag.* **2008**, *56*, 3543–3551. https://doi.org/10.1109/TAP.2008.2005436.
8. Muckenhuber, S.; Museljic, E.; Stettinger, G. Performance evaluation of a state-of-the-art automotive radar and corresponding modeling approaches based on a large labeled dataset. *J. Intell. Transp. Syst.* **2021**, 655–674. https://doi.org/10.1080/15472450.2021.1959328.
9. Wheeler, T.A.; Holder, M.; Winner, H.; Kochenderfer, M.J. Deep stochastic radar models. In *2017 IEEE Intelligent Vehicles Symposium (IV)*; IEEE: Piscataway, NJ, USA, 2017; pp. 47–53. https://doi.org/10.1109/IVS.2017.7995697.
10. Lelowicz, K. Camera model for lens with strong distortion in automotive application. In Proceedings of the 2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 26–29 August 2019; pp. 314–319.
11. Lelowicz, K.; Jasiński, M.; Piłat, A.K. Discussion of novel filters and models for color space conversion. *IEEE Sens. J.* **2022**, Volume: 22, Issue: 14, 15 July 2022, pp. 14165 - 14176.
12. Wang, T.C.; Liu, M.Y.; Zhu, J.Y.; Tao, A.; Kautz, J.; Catanzaro, B. High-resolution image synthesis and semantic manipulation with conditional gans. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18-23 June 2018; pp. 8798–8807.
13. Genser, S.; Muckenhuber, S.; Solmaz, S.; Reckenzaun, J. Development and experimental validation of an Intelligent Camera Model for Automated Driving. *Sensors* **2021**, *21*, 7583.
14. Kim, T.; Song, B. Detection and tracking of road barrier based on radar and vision sensor fusion. *J. Sens.* **2016**, *2016*.
15. Romero, L.M.; Guerrero, J.A.; Romero, G. Road curb detection: A historical survey. *Sensors* **2021**, *21*, 6952.
16. Nobis, F.; Geisslinger, M.; Weber, M.; Betz, J.; Lienkamp, M. A deep learning-based radar and camera sensor fusion architecture for object detection. In Proceedings of the 2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, Germany, 15–17 October 2019, pp. 1–7.
17. Segata, M.; Cigno, R.L.; Bhadani, R.K.; Bunting, M.; Sprinkle, J. A LiDAR Error Model for Cooperative Driving Simulations. In Proceedings of the 2018 IEEE Vehicular Networking Conference (VNC), Taipei, Taiwan, 5–7 December 2018. https://doi.org/10.1109/VNC.2018.8628408.
18. Pankiewicz, N.; Wrona, T.; Turlej, W.; Orłowski, M. Promises and Challenges of Reinforcement Learning Applications in Motion Planning of Automated Vehicles. In *International Conference on Artificial Intelligence and Soft Computing*; Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 318–329.
19. Hanke, T.; Hirsenkorn, N.; van Driesten, C.; Garcia-Ramos, P.; Schiementz, M.; Schneider, S.; Biebl, E. A Generic Interface for the Environment Perception of Automated Driving Functions in Virtual Scenarios. 2019. Available online: https://www.hot.ei.tum.de/forschung/automotive-veroeffentlichungen (accessed on 9 November 2022).
20. Hanke, T.; Hirsenkorn, N.; Dehlink, B.; Rauch, A.; Rasshofer, R.; Biebl, E. Generic architecture for simulation of ADAS sensors. In Proceedings of the International Radar Symposium, Dresden, Germany, 24–26 June 2015; pp. 125–130. https://doi.org/10.1109/IRS.2015.7226306.
21. Zhu, Z.; Zhao, H. A Survey of Deep RL and IL for Autonomous Driving Policy Learning. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 14043–14065. https://doi.org/10.1109/TITS.2021.3134702.
22. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
23. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arXiv:1506.02438.
24. Shalev-Shwartz, S.; Shammah, S.; Shashua, A. On a formal model of safe and scalable self-driving cars. *arXiv* **2017**, arXiv:1708.06374.
25. Turlej, W.; Pankiewicz, N. Adversarial Trajectories Generation for Automotive Applications. In Proceedings of the 2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland, 23–26 August 2021, pp. 115–120.