THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Machine Learning Based Methods for Virtual Validation of Autonomous Driving

TOBIAS JOHANSSON

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Machine Learning Based Methods for Virtual Validation of Autonomous Driving

Tobias Johansson

Licentiatavhandlingar vid Chalmers tekniska högskola
ISSN 1652-876X

Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
`www.chalmers.se`

This thesis has been prepared using LaTeX.
Printed by Chalmers Digitaltryck
Gothenburg, Sweden, December 2021

Machine Learning Based Methods for Virtual Validation of Autonomous Driving
Tobias Johansson
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

During the last decade, automotive manufacturers have introduced increasingly capable driving automation functions in consumer vehicles. As the functionality becomes more advanced, the task of driving moves from the human to the car. Hence, making sure that autonomous driving (AD) functions are reliable and safe is of high importance. Often, increased levels of automation result in more complex safety validation procedures, that may be both expensive, time consuming, and dangerous to perform. One way to address these problems is to move parts of the validation to the virtual domain.

In this thesis, we investigate methods for validating AD functionality in virtual simulation environments, using methods from machine learning and statistics. The main focus is on how to make virtual simulations resemble real-world conditions as closely as possible. We tackle this with an approach based on sensor error modeling. Specifically, we develop a statistical sensor error model that can be used to make ideal object measurements from simulations resemble measurements obtained from the perception system of a real-world vehicle. The model, which is based on autoregressive recurrent mixture density networks, was trained on sensor error data collected on European roads.

The second part considers system falsification using reinforcement learning (RL); a flexible framework for validation of system safety, which naturally allows for the integration of, e.g., sensor error models. We compare results of system falsification using RL to an exact approach based on reachability analysis.

With this thesis, we take steps towards more realistic statistical sensor error models for virtual simulation environments. We also demonstrate that approximate methods based on reinforcement learning may serve as an alternative to reachability analysis for validation of high-dimensional systems. Finally, we connect the RL falsification application to sensor error modeling as a possible direction for future research.

Keywords: Sensor Error Modeling, Autonomous Driving, Mixture Density Networks, Falsification, Validation, AD Simulation, Virtual Validation

List of Articles

This thesis is based on the following articles:

[A] **Tobias Johansson**, Anders Ödblom, Alexander Schliep, "Autoregressive Recurrent Mixture Density Networks for Sensor Error Generation in Autonomous Driving". *Submitted for publication.*

[B] **Tobias Johansson**, Angel Molina, Alexander Schliep, Paolo Falcone, "Reinforcement Learning as an Alternative to Reachability Analysis for Falsification of AD Functions". Machine Learning for Autonomous Driving Workshop at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Acknowledgments

This thesis would never have been written without the help and support from quite a few people. First and foremost, I would like to thank Anders and Fredrik for hiring me, and Alexander for taking on the project from the academic side. The discussions I have had with both Anders and Alexander during these two and a half years have been invaluable and I would not have gotten this far without your continuous support. I must also thank my co-advisor Morteza and examiner Devdatt for interesting discussions and for being part of the follow-up group.

I would like to thank all my friends and colleagues who have supported me in different ways. My PhD student colleagues: Angel, Shuangshuang, Niklas, Russ, Juliette, Emilio, Emil and the others at the Data Science and AI division; and my colleagues at Volvo Cars, who have brought different perspectives to my work and been great fika buddies: Micke, Melih, Goksan, Rickard, Zijian, Irene, Alexander, Francesco, Sadegh, Håkan and the rest of 96420.

Without the love and support from my family, I would most certainly not have pursued post-graduate research. I am forever grateful to my parents, Berit and Mikael who have always been there for me no matter what, Daniel for coming with good suggestions on the thesis and for being a good climbing buddy, and of course Ida and Gustav who mean the world to me.

Finally, I would like to thank Volvo Car Corporation for allowing me to do full time research on such an interesting subject.

Contents

Part I

Overview

CHAPTER 1

Introduction

The introduction of autonomous driving (AD) has the potential to change how transportation is perceived in a fundamental way. Time spent focused behind the steering wheel driving from one place to another can instead be dedicated to more rewarding endeavors. Moreover, without the need for a human driver, cars can be designed according to completely different requirements to enable a more pleasant journey, and to provide the passengers with an increased level of safety. There is, however, a long way to go before fully autonomous passenger vehicles will be standard on all public roads. For this to become reality, further developments in both software and hardware in essentially all levels of the AD stack are required, and evidence must be presented to convince consumers and regulatory agencies that each AD vehicle is safe in traffic. Exactly what evidence we need to collect is currently an open question, and a significant amount of time and effort is being put into coming up with answers to this, both from industry and academia.

The number of ways from which to approach safety validation of autonomous vehicles is vast. In this thesis, we focus on using tools from machine learning and statistics to propose methods for evaluating AD safety. Specifically, we focus on methods to be used in virtual simulation environments. There are many benefits with a successful implementation of such an approach in terms of experiment execution speed, safety, and cost. In the following section, we give a background on autonomous driving, explain the common approaches to AD validation, and contrast physical validation methods to virtual ones. After this, we formulate the problem treated in this thesis, and state our contributions.

1.1 Preliminaries

1.1.1 Autonomous Driving

In the last decade, car manufacturers have made tremendous efforts to develop autonomous driving functionality. While AD has been promoted for a very long time (with examples such as the General Motors futurama exhibition at the 1939 world fair, featuring radio controlled cars on so-called superhighways [1], and popular culture appearances, such as KITT in Knight rider), it is not until recently that computing power and sensor technology have become affordable and small enough for AD to be a potentially reachable target. The reasons to expand driving automation functionality for consumer vehicles are plenty. Of particular importance is the possibility to reduce the number of traffic accidents caused by human errors each year. Handing over critical collision mitigation decisions to a computer that does not experience fatigue and can react within milliseconds will undoubtedly save lives. This will not only benefit the occupants of the AD vehicle, but also vulnerable road users such as pedestrians and cyclists, which make up over half of all traffic deaths [2].

A modern car, capable of a high degree of driving automation, is a complex piece of machinery consisting of many interlinked components. Focusing on the specifics of automated vehicles, the functionality linked to the AD software is often divided into three parts: perception, planning and control [3]. In the perception stage, the vehicle collects and processes information about the surroundings and its own state, to create a representation of the environment for usage in the planning phase. The planning task is to determine a course of actions to take, in order to adhere to the driving objectives, such as following a predetermined map path, or avoiding sudden obstacles in the road. Finally, the control stage translates decisions of the planning stage to actuator re-

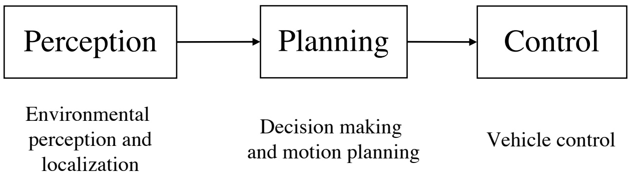
Perception
Planning
Control
Environmental perception and localization
Decision making and motion planning
Vehicle control

Figure 1.1: The three components of driving automation software.

quests. The flow is illustrated in Figure 1.1. Formally, the perception stage is made up of two parts: environmental perception and localization. The latter refers to the task of estimating entities such as pose, velocity and global position of the ego vehicle, i.e., the AD vehicle itself. In this thesis, we exclusively refer to environmental perception when we mention perception. This is the task of understanding the surrounding environment by, e.g., locating objects and drivable road area.

It is common to classify automated vehicles into different categories depending on the level of automation they are capable of. The most frequently used categorization is based on five classes [4]. Class 1 marks the lowest level of automation, and class 5 indicates fully autonomous driving. Examples of level 1 functionality are adaptive cruise control (ACC) and lane keeping assistance (LKA). By combining ACC and LKA we advance to level 2. Vehicles with AD functionality reaching level 3 and above, are often referred to as unsupervised self-driving cars. This reflects the fact that the driver does not have to monitor the driving, which is a requirement for levels 1 and 2. The differences between levels 3, 4 and 5 are mostly connected to when, or if, the driver needs to be attentive and take control of the vehicle, as well as the operational design domain (ODD) of the AD functions. Most functions on level 1 and 2 are referred to as advanced driver assistance system (ADAS) functions, but ADAS is a more general term, not exclusively connected to driving automation.

1.1.2 Validation and verification of AD systems

There is a long history of increasing the level of safety in passenger vehicles. The introduction of the three point seat belt [5], and the airbag [6], mark groundbreaking events, and structural safety improvements are continuously being made with the help of crash-test experiments and advanced simulation methods. With the inception of ADAS and AD, a new dimension is added to safety validation and verification. In particular, a much stronger emphasis is placed on software components. Standards and legal frameworks have been devised for the development and evaluation of such systems [7]–[9]. The standards help in defining a common ground for automated function development, but in terms of validating unsupervised self-driving functionality, several questions remain unanswered. One of the main reasons is that there is no consensus within industry and regulatory bodies on how to perform validation of such functionality. Hence, a large responsibility is put on automotive manufacturers and suppliers to demonstrate that their product is safe, and to propose methods for use in future standardization.

The methods currently employed to validate AD functionality range from fully physical testing on public roads or test tracks, to semi-physical setups such as hardware-in-the-loop (HIL) testing, all the way to fully virtual simulation environments with software-in-the-loop (SIL) validation. All of these have more or less desirable aspects. Public road testing often requires driving large distances [10] to get enough variation with respect to traffic scenario coverage, costing both time and money. While a trained test driver is typically present, there is always a possibility that the functionality being tested contains errors, putting the driver, and other road users at risk.

Test track experiments are time consuming to set up, and a large investment in infrastructure and personnel to manage the test equipment is often necessary. However, performing physical testing is useful to obtain an understanding of how the vehicle acts in real-world conditions.

By using virtual validation methods, experiments can be controlled to a much higher degree compared to the physical counterparts. They are also relatively inexpensive to perform and do not risk damaging people or property. A major challenge with virtual methods is to make them resemble real-world driving conditions as close as possible. This can mean implementing realistic driver models, accurate vehicle dynamics or to model the environmental perception system, of which the latter is treated in this thesis. Another challenge is to make virtual simulations fast and scalable enough for practical usage.

1.2 Problem Formulation

The problem treated in this thesis is how to validate autonomous driving functionality using computer simulations. We specifically ask: *How can virtual simulations be made more realistic through sensor error modeling?* As a secondary objective, we have touched upon the question: *How can reinforcement learning be used for system validation in virtual simulation environments?*

1.3 Contributions

The primary contribution of this thesis is a sensor error model based on autoregressive recurrent mixture density networks (Article A; see Section 3.1). With this model, we take steps towards a more realistic statistical sensor error modeling approach for virtual simulation environments. A secondary contribution is the comparison of falsification using reinforcement learning (RL) and reachability analysis for linear systems (Article B; see Section 3.2). While these contributions may seem far apart, there is a clear connection between them which is discussed in Section 2.2.1).

1.4 Thesis outline

Chapter 2 provides a background to the topics explored in this thesis. Focus is on sensor error modeling and its use in an AD validation framework, as treated in Article A. An introduction to the concept of falsification using reinforcement learning is also given, connecting to Article B (see Section 3.2). In Chapter 3, a summary of the included articles is presented, and Chapter 4 provides conclusions and directions for future research. The second part of this thesis contains the included articles, adapted to the layout of the thesis.

CHAPTER 2

Background

This chapter introduces the core concepts of the two articles on which this thesis is based. We begin with explaining the role of sensor models, specifically sensor error models, in the AD validation chain. The main building block of the model developed in Article A, namely mixture density networks, is explained in greater detail, and two important model validation metrics are discussed.

In Section 2.2, we describe the problem of falsification, a testing approach meant to make a system fail to perform according to some given specifications, by controlling system disturbances. This is the foundation of Article B, and to facilitate understanding we give a more in depth overview of the fundamental topics here. In particular, we cover the reinforcement learning algorithm used for the experiments. We also give a short description of reachability analysis and provide references for the interested reader.

2.1 Sensor error modeling

One of the most fundamental parts of an AD car is the perception system. There, measurements of the surrounding environment are made and processed into a digital representation which is used in the planning and control functions. It is clear, that if the vehicle perceives the environment in an inadequate way, it will not be able to make grounded decisions during the planning and control phases either. To make the perception as accurate as possible, an AD vehicle is typically equipped with a wide range of sensors with different capabilities and strengths. Common sensor setups include cameras, LiDARs, radars, and ultrasonic sensors. The sensor measurements are combined in a sensor fusion stage to form an as good approximation of the environment as possible. While sensor redundancy, and sensor fusion, makes the perception system significantly more accurate and robust, noise and erroneous measurements will still inevitably be present. From a validation perspective, it is important to take these errors into account when evaluating the system. In particular, this applies for virtual validation methods where input to the AD functions can be completely controlled by feeding them with ideal sensor data. These ideal measurements are extracted directly from the ground truth representations of the virtual simulation environment. However, ideal sensor data may not match corresponding measurements from real-world sensors. As a result, the planning and control functions may behave differently in simulations compared to reality. One way to address this is to incorporate sensor models, meant to reflect the characteristics of real-world vehicle sensors, in the simulation environment.

There are many ways in which sensor models can be implemented. For example, modeling physical sensor characteristics by simulating electromagnetic fields [11] is common for evaluating radar sensors. Ray tracing techniques have previously been used for both LiDAR and radar [12], [13], and many recent studies consider generation of realistic camera data from a virtual simulation [14], [15]. All of these approaches have benefits and drawbacks: modeling physical characteristics of radars using field simulations can result in highly accurate approximations, but are often very computationally demanding; there are efficient models for camera image generation, but evaluating the output is challenging; and ray tracing for LiDARs can be performed in near-real time, but may be misleading, depending on how well material properties are modeled. In this thesis, we take a statistical approach to sensor modeling, in which sensor errors are modeled based on collected expedition data. Here, we target positional measurements of objects that have been detected in the surrounding environment. This allows for fast error generation and enables

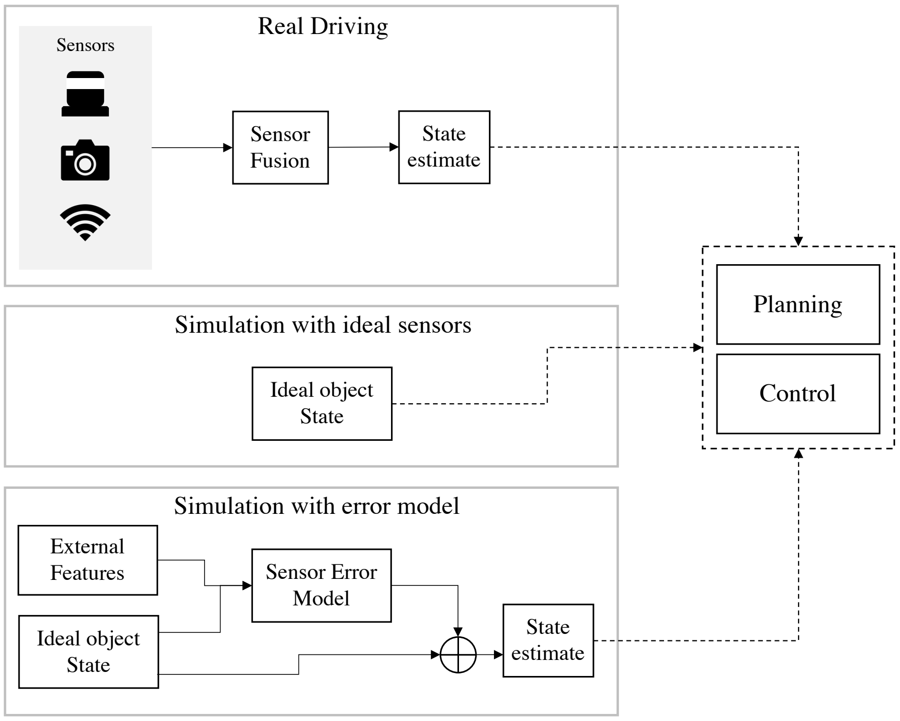

Figure 2.1: Input to the planning and control functions for real world driving (upper), simulations with ideal sensors (mid), and simulations with sensor error models (lower).

capturing errors arising from not only the sensors, but also the fusion of sensor measurements. In the following section, we describe how statistical sensor error modeling fits into an AD validation framework.

2.1.1 Fit into validation framework

As indicated in the previous section, feeding the AD planning and control functions with ideal sensor data in a simulation environment may lead to unrealistic simulation results. We can mitigate this by adding an error to the ideal measurement, and hence disturb a signal of interest, according to errors observed during data collection expeditions. This process is illustrated in

Figure 2.1. In the upper part, we see a simplified version of a typical perception chain for an AD car. The middle part of the figure shows the input to the AD functions in a simulation environment with ideal sensor measurements, and the lower part displays the same setting when a sensor error model is added to the simulation. The figure shows how, by adding a realistic sensor error to the ideal object state, we can generate input to the planning and control functions that corresponds to real-world data. Sensor error models can be constructed to disturb specific components of the ideal object state, such as the velocity or relative position of a tracked object. This thesis has focused on the generation of positional sensor errors, but in theory, any component in the ideal object representation can be modeled in similar ways.

2.1.2 Statistical sensor error modeling

By collecting large amounts of driving data and simultaneously record the perception error for tracked objects, it is possible to construct a statistical model relating ideal object states and other potentially influencing factors, such as weather conditions, to the sensor error. Some notable examples of sensor error models for automotive applications include [16], [17] and in particular [18], [19] which have been used as benchmarks for the model developed within the scope of this thesis.

Before creating a model we need access to data. In practice, collecting accurate sensor error data is a challenging task involving two main components: the collection of reference (or ground truth) data, and the matching of detected objects from the reference measurements with those of the original perception system. These components are described below.

Collecting reference data

The sensors of a production vehicle meant for private consumers must be affordable. Moreover, sensors should be relatively energy efficient and sufficiently small for practical integration in the vehicles. These restrictions naturally impact the performance of the perception system as well. When collecting sensor data with prototype equipment, the limitations are largely removed. We can use more expensive sensors, since they will only be mounted on a few prototype vehicles, and it is possible to add external processing power, together with additional energy sources, to allow for a substantial improve-

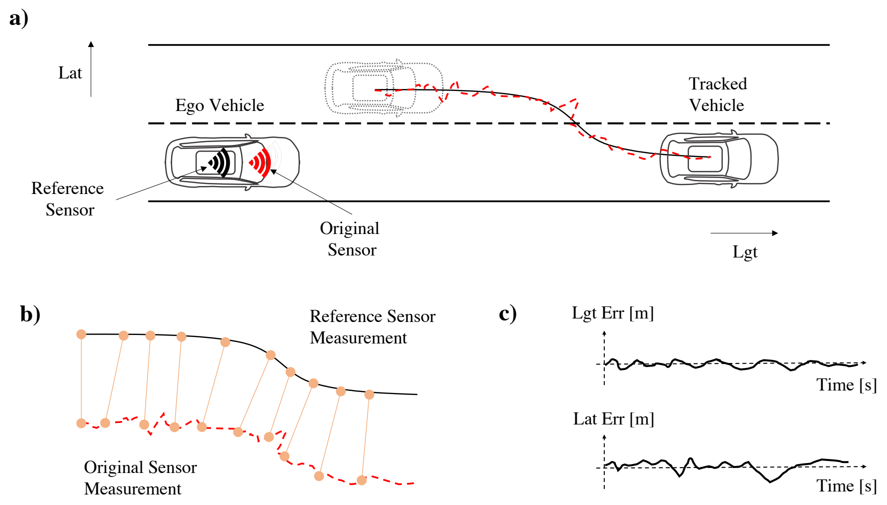

Figure 2.2: Illustration of sensor error data collection and matching: a) Both the reference system and original sensors are used to track the target vehicle. b) Sampling times are matched and the difference is computed between the measurements. c) Time series of longitudinal and lateral positional sensor errors resulting from the computations.

ment of the measurement fidelity. By using such a sensor system to collect object data, we obtain much more accurate estimates of the true object state than what is possible using the original perception system.

Matching objects

The computation of errors using collected reference data is a matter of subtracting state estimates obtained from the original perception system with those given by the reference system. However, since real world environments are often highly complex, where multiple objects may be simultaneously tracked, a matching procedure, such as [20], must be employed prior to computing this difference. The process of reference data object matching is visualized in Figure 2.2.

2.1.3 Mixture Density Networks

One of the main building blocks of the sensor error model presented in [21] is a mixture density network (MDN) [22]. In this section we present an overview of this type of model and discuss the reasons for using MDNs for sensor error modeling.

Mixture Density networks are constructed of a neural network that outputs parameters to a mixture distribution. By scoring observations using the likelihood of a mixture distribution with parameters given by the neural network, we have constructed a loss function that honors the distribution of residuals to a much higher degree than other commonly used losses. Before going into the details of MDNs we begin with an illustrative example of how a continuous variable would be predicted using mean squared error as the loss function, and point to situations where this approach may be problematic. Suppose that we observe data $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i \in \mathbb{R}^k,\ k \in \mathbb{N}^+$, are features and $y_i \in \mathbb{R}$ is a response variable of interest. Our goal is to fit a neural network f, parametrized by a set of weights $\mathbf{w}$, to input x_i and output y_i as well as possible. This can be achieved by optimizing a loss function with respect to the network weights $\mathbf{w}$. Often, when the response is continuous, the mean squared error loss function is used:

$$L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i; \mathbf{w}) - y_i)^2. \tag{2.1}$$

Here, $f(x_i; \mathbf{w}) \in \mathbb{R}$ denotes the output of the neural network for input x_i and weights $\mathbf{w}$. While we have made no explicit distributional assumptions on the residuals $\epsilon = f(x_i; \mathbf{w}) - y_i$ it can be shown that the problem can be solved in an identical fashion by assuming that y_i is normally distributed with conditional mean determined by the neural network, and a global variance parameter σ^2, that is

$$y_i \sim \mathcal{N}(y_i; f(x_i; \mathbf{w}), \sigma^2). \tag{2.2}$$

When maximizing the likelihood of $y_i,\ i = 1, \ldots, n$, by tuning the network weights, we get the same results as when minimizing the mean squared error loss (2.1). In other words, we obtain the least squares solution by maximizing a normal likelihood.

In some situations minimizing MSE loss is not appropriate, including when y_i exhibits a multimodal behavior that is not reflected in x_i, or when knowing precise distributional characteristics of the residuals are of high value. These issues can be mitigated by yet again taking the maximum likelihood approach when searching for optimal network weights $\mathbf{w}$, but instead of the conditional normal ansatz, we assume that y_i is distributed according to a mixture distribution. Most often Gaussian (normal) mixtures are used for this purpose. The resulting model is a mixture density network. Hence, by replacing the distributional assumption (2.2) with a mixture distribution and adapting equation (2.1) to a log-likelihood loss, the loss function for the MDN becomes

$$L(\mathbf{w}) = -\sum_{i=1}^{n} \log p(y_i|x_i),$$

where $\log p(y_i|x_i)$ is the log-likelihood of y_i under a (Gaussian) mixture density with M mixture components. That is,

$$p(y_i|x_i) = \sum_{m=1}^{M} \pi_m(x_i)\mathcal{N}(y_i; \mu_m(x_i), \sigma_m^2(x_i)),$$

where we have assumed that samples $\{(x_i, y_i)\}_{i=1}^n$ are independent. Here, the parameters for the mixture distribution $\pi_m, \mu_m, \sigma_m^2,\ m = 1, \ldots, M$ are the outputs of a neural network, and hence functions of x_i. Moreover, the mixture parameters are constrained as follows: the components of the mixture weight vector $\pi = (\pi_1, \ldots, \pi_M)$ are non-negative and sum to one; the variance σ_m^2 is positive; and the mean μ_m takes values in the real numbers. In practice, these constraints are met by letting the neural network have three output layers, denoted by $\pi,\ \mu$ and σ^2. That is, one for each set of parameters. The π layer uses a softmax activation function, σ^2 applies an exponential activation and μ a linear activation.

In Article A, MDNs are used as the main component of the proposed sensor error model. However, the model in the article is more complicated, involving keeping track of a scenario history implemented through recurrent neural networks, and an autoregressive connection of the sampled error. These additions were necessary to enable modeling of realistic error series.

Figure 2.3 illustrates how an MDN can be used for statistical sensor error modeling. From left to right: scenario information x_t is given as input to the

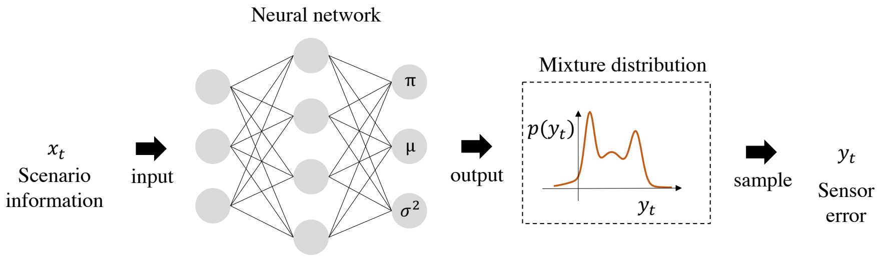

Figure 2.3: Sensor error generation using mixture density networks.

neural network; the network outputs vectors of mixture components, which are used to form a mixture distribution; a sensor error y_t is sampled from the mixture. Again, note that the figure displays a simplified version, without the inclusion of scenario history dependence and autoregression.

2.1.4 Model evaluation metrics

Evaluating a probabilistic sensor error model is a challenge in itself. Often multiple metrics are used to determine how well a model performs and it is not uncommon that different models perform well on some metrics and worse on others, making an objective judgment of model performance difficult. In this section, we give a more detailed description of some of the metrics used in Article A. In particular, we focus on the Jensen Shannon distance and the empirical likelihood for evaluating how well a model captures scenario information.

2.1.4.1 Jensen Shannon distance

Jensen Shannon distance (JSd) is a metric for comparing the similarity of two probability distributions. It is a symmetric version of the relative entropy, also known as the Kullback-Leibler (KL) divergence. In contrast to the KL divergence, the JSd defines a metric since it is symmetric, satisfies the triangle inequality, and is zero if two distributions are identical (see [23, p.13] for the definition of a metric). Formally, the JSd is defined as

$$\mathrm{JSd}(p\|q) = \sqrt{\frac{1}{2}D(p\|r) + \frac{1}{2}D(q\|r)},$$

where $r = (p+q)/2$, and $D(p\|q)$ is the KL divergence between distributions p and q, defined as

$$D(p\|q) = \mathbb{E}_p \left[\log \frac{p(X)}{q(X)} \right].$$

For evaluating sensor error model performance, the JSd is especially informative when considering performance on a global scale, such as the overall distributions of sensor errors, or the distribution of first-differences (error increments).

One important aspect of the JSd, as used in Article A, is that it effectively removes the time dependence of the errors. That is, by permuting the error series in time, e.g., sorting the errors in ascending order, we would still obtain identical JSd for the global error distribution. Hence, it is crucial to also consider metrics that are not permutation invariant.

2.1.4.2 Empirical likelihood

Using likelihood to evaluate statistical model performance is typical procedure. For the statistical sensor error models considered in this thesis, an important question is how well a model utilizes the scenario information to estimate sensor errors. One way to investigate this is to employ a likelihood based approach that uses Monte Carlo simulations to form approximate distributions of time series generated by the model, for any given scenario. This approach effectively disregards the autoregressive dependence of the error series and enables comparisons between arbitrary models that can be used to generate errors. In Figure 2.4, the process of computing the empirical likelihood is illustrated. A trained sensor error model is used to sequentially generate errors given scenario information $x_t \in \mathbb{R}^k,\ k \in \mathbb{N}^+$. The features in x_t can reflect the ideal object state, as given by either reference or ground truth data, and additional information, such as ego-vehicle state and weather conditions. By assuming that sensor errors y_t are conditionally independent under the scenario information, we can compute the log-likelihood of the observed error y_t under the empirical distribution given by the many generated time series for the same scenario $x = x_1, \ldots, x_n$, where n is the total number of time steps for which an object was tracked. We use Kernel Density Estimation (KDE) to compute estimates of $\log p(y_t|x_1, \ldots, x_t)$ for all $t \in \{1, \ldots, n\}$, i.e. the log-likelihood of the sensor error given the scenario history. By summing

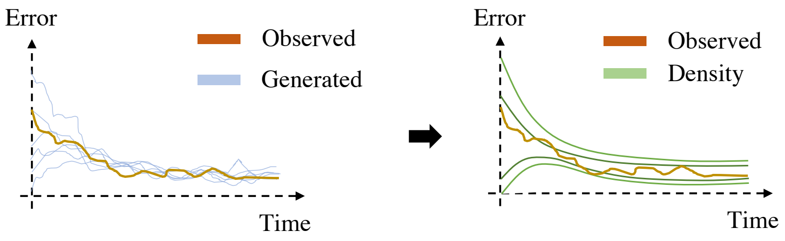

Figure 2.4: To compute the empirical likelihood metric, a model is used to generate multiple time series for the same scenario (blue lines). By smoothing the resulting empirical distribution using, e.g., Kernel density estimation (KDE), a probability density is obtained (green). This can be used to evaluate the likelihood of the observed error (orange) for each point in time.

these estimates over the entire trajectory we obtain a score on how well the scenario information is utilized by the model. For the exact procedure, see Algorithm 1 in Article A.

2.2 Falsification

The second part of this thesis treats an interesting approach to validation and verification of system safety, called falsification. The main goal in falsification problems is to disturb a system in such a way that it fails to perform according to specifications. In the context of validating autonomous driving functions, the framework fits nicely with the simulation based validation approach briefly described for sensor error modeling applications (see Section 2.1.1), where we have full control of the environment surrounding an AD enabled vehicle. Moreover, the flexibility of the falsification problem allows for a natural incorporation of, e.g., statistical object motion models, as is the case for [24], or statistical sensor error models analogously. While this thesis is mainly focused on validation methods for AD using perception error modeling, the investigation of falsification approaches which can incorporate such models, motivates the inclusion of Article B. In Section 2.2.1 we further expand this argument.

The work conducted as part of this thesis investigates falsification using

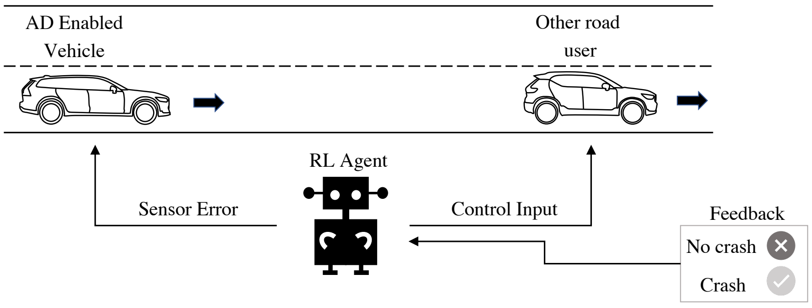

Figure 2.5: Example falsification setting.

reinforcement learning (RL). In Article B we compare falsification using reinforcement learning with an exact counterpart in the form of reachability analysis (RA) for a system that can be analyzed using both methods. This section gives an introduction to RL and the techniques used to falsify the system in Article B. We also give a brief description of RA to assist the reader of Article B. However, it should be noted that the work conducted within the scope of this thesis primarily targets falsification using reinforcement learning and not RA.

In Figure 2.5, an example falsification problem is illustrated. The leading vehicle is under the control of the RL-agent and the car behind drives autonomously. By generating control inputs to the leading vehicle and sensor errors for the AD car, the RL-agent attempts to cause a crash. If this succeeds within a given time horizon, a reward is given. Otherwise, the agent is either penalized or given no reward. Using experience collected from repeated iteration of this process, the agent learns how to generate more dangerous sequences, with a higher probability of leading to a crash.

2.2.1 Falsification and sensor error models

There are a wide range of methods available for system falsification. Some require knowledge about the inner workings of the system, such as closed-form expressions of the system dynamics, while others treat the system as a black-box. For the latter case, [25] provides a comprehensive survey. One of the benefits of using RL for falsification is connected to the simplicity of introducing likelihood based penalties for generated system disturbances. As an example, we can consider the previously mentioned work in [24]. Here, an RL agent generates motion trajectories for pedestrians with the goal of causing an accident in the simulation environment. The generated motions are penalized according to the Mahalanobis distance (which is closely connected to the likelihood of a multivariate Gaussian distribution) to a preferred action. In the same manner, we can penalize the generation of improbable sensor errors with a trained probabilistic sensor error model. In particular, this is possible when having access to a sensor error model in the form of the AR-RMDN (see Article A) which enables direct computation of error likelihoods.

2.2.2 Reinforcement learning

Reinforcement learning deals with sequential decision making under uncertainty. The problem setting is usually explained with an agent that acts in an unknown environment. By receiving feedback based on the actions made, the agent can iteratively improve its policy, i.e., which action to choose given the current environment state. The fact that it is the RL agent that generates the data used for updating its policy marks a clear distinction between RL and the typical machine learning paradigms: supervised and unsupervised learning. In the following sections, we describe some fundamental concepts of reinforcement learning and introduce the proximal policy optimization algorithm [26] used for the falsification implementation in Article B.

2.2.2.1 Markov Decision Processes and policies

It is common to model the environment, as well as the interactions the agent has with environment, using a Markov decision process (MDP). An MDP can be defined by (see [27, Chapter 2] for a detailed description) a set of decision epochs $\mathcal{T}$, a set of states $\mathcal{S}$, a set of (possibly state-dependent) actions $\mathcal{A}_s$, a state transition probability $p_t(\cdot|s, a)$ and a reward function $r_t(s, a)$. The system evolves in the following manner: at each decision epoch $t \in \mathcal{T}$ the agent observes a state $s \in S$ and makes a decision on which action $a \in \mathcal{A}_s$ to choose. Next, the agent transitions from s to a new state $s' \in \mathcal{S}$ according to $p_t(s'|s, a)$ and receives a reward $r_t(a, s)$. The policy $\pi(s)$ represents a rule for choosing actions at the decision epochs. In the following sections, we also write $\pi(a|s)$ to denote the probability of choosing action a when in state s under policy π. A typical goal of the agent is to find a policy that maximizes the total expected reward over all decision epochs $\mathcal{T}$ for a given MDP.

2.2.2.2 The value of a policy

Since different policies will result in different rewards, it is natural that some policies are considered more valuable than others. This is quantified by the value function of a policy. The state-value function of a policy is defined as follows:

Definition 1 (From [28], Chapter 3.5)**:** *The value function of a state s under a policy π, denoted $v_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, v_π can formally be defined as*

$$v_\pi(s) := \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right],$$

where $\gamma \in [0,1)$ is a discount factor, R_t the (random) reward at time t, and G_t is the return defined as

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

Another useful quantity is the action-value function:

Definition 2 (From [28], Chapter 3.5)**:** *The value of taking an action a in state s under policy π, denoted $q_\pi(s,a)$, is defined as the expected return when starting from s, taking action a, and thereafter following policy π*

$$q_\pi(s,a) := \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a],$$

where G_t is defined as in Definition 1.

In the next section we also refer to the advantage function $\mathrm{adv}(s,a)$, which is defined as

$$\mathrm{adv}_\pi(s,a) = q_\pi(s,a) - v_\pi(s). \tag{2.3}$$

The advantage function provides a notion of how good it is to choose action a in state s, compared to how good it is to be in state s in general. Often, the advantage function is denoted by $A(s,a)$. Here we used $\mathrm{adv}_\pi(s,a)$ to avoid potential confusion with the actions.

2.2.2.3 Proximal policy optimization

There are a wide range of ways in which to search for an optimal policy for a given MDP. When dealing with very large state and action spaces, e.g., continuous ones, a common approach is to utilize so-called policy gradient (PG) methods [28, Chapter 13]. The main idea behind PG methods is to parametrize the policy π using a set of parameters θ such that π_θ is differentiable with respect to θ. Since the policy is differentiable, we can use gradient based optimization methods to improve its value. A PG method that has been shown to balance ease of implementation with high performance, especially on continuous control tasks, is proximal policy optimization (PPO) [26].

Before going into the details of PPO, we should have an understanding of some of the challenges connected to PG methods. The cornerstone of PG methods is the policy gradient theorem, which relates the gradient of a performance measure $J(\theta)$ to the policy parameters in the following way

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[q_{\pi_\theta}(s,a)\nabla \log \pi_\theta(a|s)], \tag{2.4}$$

where q_{π_θ} is the action-value function for policy π_θ. The exact definition of $J(\theta)$ depends on the problem statement and is not important for this discussion (see [28] for details). By direct application of the PG theorem we end up with REINFORCE [29], a surprisingly simple algorithm for policy optimization, which also forms the basis of many subsequent algorithms. In practice, learning with REINFORCE is often slow since the gradient estimates are of high variance. This is mainly due to the fact that the return G_t is used as an unbiased estimate of the action-value function $q_{\pi_\theta}(s,a)$ in equation (2.4), which can vary significantly simply due to the stochastic nature of the environment. To reduce variance, and hence increase learning speed, we can utilize so-called actor-critic methods [28]. The idea is to use a separate function approximator, e.g., a neural network, to estimate the state-value function $v_\pi(s)$. This can then be used to compute an advantage estimate $\hat{A}$ by, e.g, generalized advantage estimation (GAE) [30]. Similar to the way G_t is used as an unbiased estimate of $q_{\pi_\theta}(s,a)$ in the REINFORCE algorithm, we can use $\hat{A}$ as a slightly biased estimate, but often with significantly reduced variance.

Another issue with PG methods is that changing the policy too much may result in a substantial change of the state and action distribution between optimization rounds. Hence, it may be necessary to somehow limit the change of the policy to ensure that the parameter inference remains stable.

Proximal policy optimization effectively combines actor-critic methods with a limitation of policy updates, in an algorithm that is both relatively simple to implement and retains many of the benefits of more complicated approaches (e.g., [31]). PPO makes use of the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ between the policy with updated parameters π_θ and the policy before parameter updates $\pi_{\theta_{\text{old}}}$, to limit the change of the policy. It considers the objective function

$$J(\theta) = \hat{\mathbb{E}}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right], \tag{2.5}$$

where ϵ is a hyperparameter controlling how far from the previous policy to step, $\hat{A}_t$ is an advantage estimate, and the expectation $\hat{\mathbb{E}}[\cdot]$ is the empirical average over a batch of samples. The function $\text{clip}(x, a, b)$ can equivalently be stated as

$$\text{clip}(x, a, b) = \max(\min(x, b), a).$$

Finding the best θ is then a matter of optimizing (2.5) using a stochastic gradient ascent algorithm.

2.2.3 Reachability analysis

Article B (see Section 3.2) is a collaborative work where my part concerned falsification using reinforcement learning. The other subject of the article is reachability analysis (RA). The RA part was contributed by the second author and is not within the scope of this thesis. We end this background with a short section meant to provide the interested reader with a notion of what RA actually is, and give some references for further reading.

Reachability analysis consists of a set of methods that can be used to formally verify system safety by analyzing possible system transitions in time with respect to a target set of states. A typical application of RA is to compute the states from which a system cannot avoid entering a predefined set of failure states, despite applying the best possible control actions. In the general non-linear continuous time case, Hamilton-Jacobi reachability analysis is often used to investigate the possible disturbances (e.g control actions) that can lead to a system entering or leaving a target state. For the problems treated in this thesis, where we consider linear discrete-time systems, it is possible to use set-theoretic methods for analyzing reachable sets. RA is an active research field with a rich history. However, this thesis places focus on other methods for validation and verification. Hence, we refer the interested reader to [32] for an excellent overview of Hamilton-Jacobi reachability, and to [33] for set-theoretic methods.

CHAPTER 3

Summary of included articles

This chapter provides a summary of the included articles.

3.1 Article A

Tobias Johansson, Anders Ödblom, Alexander Schliep.
”Autoregressive Mixture Density Networks for Sensor Error Generation in Autonomous Driving”
Submitted for publication.

In this paper, we propose a new sensor error model based on autoregressive mixture density networks (AR-RMDN). The model takes reference measurements of tracked vehicle objects, as well as state information connected to the ego-vehicle, as input and outputs an estimate of the longitudinal positional sensor error of the tracked object. A data set consisting of over 9000 object tracks was used to train and evaluate the model.

We looked into two state-of-the-art sensor errors models for benchmarks: a recurrent conditional generative network (RCGAN) [19], and an autoregressive input-output hidden Markov model (AIOHMM) [18], trained on the same data set. As the implementations of those models were not available to us, they had to be implemented from scratch. We used this opportunity to explore improvements for the models and by doing so, managed to significantly improve the computational efficiency of the AIOHMM using just-in-time compilation features for Python.

A range of different evaluation metrics were considered, including Jensen Shannon distance (JSd) and empirical likelihood. We found that our model outperformed the two others on all considered metrics. In particular, the JSd of the global error increment (first-difference) distribution was significantly improved. We also observed that the AR-RMDN seemed to combine desirable aspects from the other models, such as the representational power of deep learning connected to the RCGAN, and the training stability and more reasonable error variability of the AIOHMM.

I was responsible for conducting and designing the experiments, coming up with the idea of the model, and writing the paper.

3.2 Article B

Tobias Johansson, Angel Molina, Alexander Schliep, Paolo Falcone.
”Reinforcement Learning as an Alternative to Reachability Analysis for Falsification of AD Functions”
Machine Learning for Autonomous Driving Workshop at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

In this paper, we present reinforcement learning (RL) as a possible alternative to reachability analysis (RA) for validation of autonomous driving functions. As background, RA is powerful tool for system verification but suffers an exponential increase in computational requirements when the system dimension grows. The idea of the article was to investigate if RL could be used as an alternative to RA for high dimensional system verification and validation. For this purpose, we investigate falsification of a linear adaptive cruise controller (ACC) using both RA and RL. While the ACC controller is low-dimensional system, it allows for exact computations of both RL and RA falsification results, necessary for comparing the two methods.

We found that RL can serve as an alternative to RA in falsifying the ACC system, given that some approximation error is acceptable. Moreover, the RL approach was able to handle falsification of a larger region of the state-disturbance space compared to RA. This indicates that even for low-dimensional systems, despite the relatively poor sample complexity in terms of calls to the simulator, RL may be useful for system validation. More interestingly, we presented preliminary results for a higher dimensional lateral control example, for which RA could not obtain a solution within reasonable time. In this example RL was able to falsify the system. While the results are preliminary, they serve as an indication of that RL may be used instead of RA for falsification of higher-dimensional systems.

This paper was a collaborative work conducted together with Angel Molina. Equal contribution by the first two authors. I designed and conducted the experiments on reinforcement learning, co-wrote the paper, and jointly came up with the idea of the paper together with Angel Molina. I did not contribute to the reachability analysis.

CHAPTER 4

Concluding remarks and future work

In this thesis we have explored virtual validation methods for autonomous driving based on machine learning. The problem of making virtual simulations more realistic was addressed using sensor error modeling and a new probabilistic model was developed for this purpose - the Autoregressive Recurrent Mixture Density Network (AR-RMDN). The new model uses mixture density networks as its main component, which enables it to capture highly complex error distributions, and hence, generate realistic errors. When evaluated against other state-of-the-art models we found that the AR-RMDN performed better on all metrics considered.

We also investigated falsification of AD functions from the perspective of reinforcement learning and compared the results with exact solutions obtained from reachability analysis. The RL falsification approach used PPO to optimize parameters of a policy with Beta-distributed actions. We found that RL can potentially be used as an alternative to reachability analysis for the verification of high-dimensional systems, following fair correspondence on low-dimensional tasks (falsifying an Adaptive Cruise Controller) and promising results on preliminary data. However, to draw further conclusions, a more thorough investigation is needed.

During this work, we have identified several questions which would be interesting to pursue in future research. For sensor error modeling we ask

1. How can probabilistic sensor error models be confidently deployed in simulation environments?
2. How can the generative performance of probabilistic error models be evaluated on a per-time-series basis?
3. How can models be adapted to new sensor software and hardware without the need of extensive data collection?

The first and second points are closely related as they treat the generative performance of error models. During our experiments, we have found that there is a risk that 'strange' error series are generated by the model. Specifically, the model occasionally generates errors that are quite far from the typical errors under a given scenario. From an evaluation point of view these error series are difficult. Do they occur due to model irregularities or are they realistic but very infrequent? The third point has not been treated in this thesis but is certainly interesting to consider. It would be useful to somehow transfer information from models trained with similar perception setups to new systems, and hence avoid having to collect a lot of data for each system change.

The investigations of falsification methods using reinforcement learning revealed several aspects that deserve deeper investigation. We noticed that scalability with respect to the state-disturbance dimension was not necessarily as bad as for reachability analysis (according to preliminary results). However, solving even simple problems, such as the falsification of an adaptive cruise controller, is relatively expensive in terms of computational requirements. If the simulation environment is very complex, using this falsification approach may not be feasible. Hence, we need more sample efficient learning algorithms to be able to apply RL-based falsification as a virtual validation tool.

As discussed in Section 2.2.1, by using RL for falsification we enable a straightforward integration of sensor error models in the problem. This would be interesting to explore further, especially in the context of identifying probable disturbance sequences leading to specification violations. Previous work has been conducted in this direction [24], [34], [35] but without explicitly considering realistic sensor error models.

References

[1] R. A. Ferlis, “The dream of an automated highway,” Federal Highway Administration, Tech. Rep. FHWA-HRT-07-005, Jul. 2007.

[2] W. H. Organisation, “Road traffic injuries: Fact sheet,” Tech. Rep., 2021.

[3] S. D. Pendleton, H. Andersen, X. Du, *et al.*, “Perception, planning, control, and coordination for autonomous vehicles,” *Machines*, vol. 5, no. 1, p. 6, 2017.

[4] On-Road Automated Driving (ORAD) committee, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, Available: `https://doi.org/10.4271/J3016_202104`, Apr. 2021.

[5] Y. Håland, “The evolution of the three point seat belt from yesterday to tomorrow,” in *IRCOBI Conference*, 2006.

[6] D. E. Struble, “Airbag technology: What it is and how it came to be,” SAE Technical Paper, Tech. Rep., 1998.

[7] ISO, “26262:2018 road vehicles — functional safety,” International Organization for Standardization, Geneva, Switzerland, Standard, Dec. 2018.

[8] ISO/PAS, “21448:2019 road vehicles — safety of the intended functionality,” International Organization for Standardization, Geneva, Switzerland, Standard, Jan. 2019.

[9] United Nations Economic Commission for Europe (UNECE), *Proposal for a New UN Regulation on Uniform Provisions Concerning the Approval of Vehicles with Regards to Automated Lane Keeping System*, Available: `https://undocs.org/ECE/TRANS/WP.29/2020/81`, 2020.

[10] N. Kalra and S. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[11] S. Vitebskiy, L. Carin, M. Ressler, and F. Le, "Ultra-wideband, short-pulse ground-penetrating radar: Simulation and measurement," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 35, no. 3, pp. 762–772, 1997.

[12] N. Hirsenkorn, P. Subkowski, T. Hanke, *et al.*, "A ray launching approach for modeling an fmcw radar system," in *2017 18th International Radar Symposium (IRS)*, 2017, pp. 1–10.

[13] L. Winiwarter, A. M. E. Pena, H. Weiser, *et al.*, "Virtual laser scanning with helios++: A novel take on ray tracing-based simulation of topographic 3d laser scanning," *arXiv preprint arXiv:2101.09154*, 2021.

[14] Z. Yang, Y. Chai, D. Anguelov, *et al.*, "Surfelgan: Synthesizing realistic sensor data for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 118–11 127.

[15] Y. Chen, F. Rong, S. Duggal, *et al.*, "Geosim: Realistic video simulation via geometry-aware composition for self-driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7230–7240.

[16] T. Hanke, N. Hirsenkorn, B. Dehlink, A. Rauch, R. Rasshofer, and E. Biebl, "Generic architecture for simulation of ADAS sensors," in *2015 16th International Radar Symposium (IRS)*, Jun. 2015, pp. 125–130.

[17] N. Hirsenkorn, T. Hanke, A. Rauch, B. Dehlink, R. Rasshofer, and E. Biebl, "Virtual sensor models for real-time applications," *Advances in Radio Science*, vol. 14, pp. 31–37, 2016.

[18] E. L. Zec, N. Mohammadiha, and A. Schliep, “Statistical Sensor Modelling for Autonomous Driving Using Autoregressive Input-Output HMMs,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, pp. 1331–1336, 2018.

[19] H. Arnelid, E. L. Zec, and N. Mohammadiha, “Recurrent Conditional Generative Adversarial Networks for Autonomous Driving Sensor Modelling,” *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 1613–1618, 2019.

[20] J. Florbäck, L. Tornberg, and N. Mohammadiha, “Offline object matching and evaluation process for verification of autonomous driving,” in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2016, pp. 107–112.

[21] T. Johansson, A. Ödblom, and A. Schliep, “Autoregressive recurrent mixture density networks for sensor error generation in autonomous driving,” *Unpublished Manuscript*, 2021.

[22] C. M. Bishop, *Mixture density networks*, 1994.

[23] G. B. Folland, *Real analysis: modern techniques and their applications.* John Wiley & Sons, 1999, vol. Second edition.

[24] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, “Adaptive stress testing for autonomous vehicles,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1–7.

[25] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer, “A Survey of Algorithms for Black-Box Safety Validation,” *Retrieved from http://arxiv.org/abs/2005.02979*, 2020.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[27] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2005.

[28] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[29] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[30] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” *arXiv preprint arXiv:1506.02438*, 2015.

[31] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

[32] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, “Hamilton-jacobi reachability: A brief overview and recent advances,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, IEEE, 2017, pp. 2242–2253.

[33] F. Blanchini and S. Miani, *Set-theoretic methods in control.* Springer, 2008.

[34] M. Koren and M. J. Kochenderfer, “Adaptive stress testing without domain heuristics using go-explore,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[35] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, “Adaptive stress testing with reward augmentation for autonomous vehicle validation,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 163–168.