PEM

Perception Error Model for Virtual Testing of Autonomous Vehicles

Piazzoni, Andrea; Cherian, Jim; Dauwels, Justin; Chau, Lap Pui

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

PEM: Perception Error Model for Virtual Testing of Autonomous Vehicles

Andrea Piazzoni, Jim Cherian, Justin Dauwels, *Senior Member, IEEE*, and Lap-Pui Chau, *Fellow, IEEE*

Abstract— **Even though virtual testing of Autonomous Vehicles (AVs) has been well recognized as essential for safety assessment, AV simulators are still undergoing active development. One particular challenge is the problem of including the Sensing and Perception (S&P) subsystem into the virtual simulation loop in an efficient and effective manner. In this article, we define Perception Error Models (PEM), a virtual simulation component that can enable the analysis of the impact of perception errors on AV safety, without the need to model the sensors themselves. We propose a generalized data-driven procedure towards parametric modeling and evaluate it using Apollo, an open-source driving software, and nuScenes, a public AV dataset. Additionally, we implement PEMs in SVL, an open-source vehicle simulator. Furthermore, we demonstrate the usefulness of PEM-based virtual tests, by evaluating camera, LiDAR, and camera-LiDAR setups. Our virtual tests highlight limitations in the current evaluation metrics, and the proposed approach can help study the impact of perception errors on AV safety.**

Index Terms— **Autonomous vehicles, computer vision, vehicle safety, simulation.**

I. INTRODUCTION

ONE of the major limitations preventing a public deployment of Autonomous Vehicles (AVs) is that their safety is not yet guaranteed or well-established in general. Furthermore, Shen et al. [1] emphasize how the explainability of Artificial Intelligence (AI) in AVs is necessary for trust and acceptance. Hence, the automotive industry, the academic community, and regulators are developing safety assessment procedures. By analyzing the AV Operational Design Domain (ODD) [2], we can identify the challenges that the AV will encounter. In particular, sensors are known to be very susceptible to weather conditions and the amount of sunlight [3].

The AV should handle such issues and still exhibit safe driving behavior. For example, if the perception uncertainty increases, the AV could reduce its speed and adopt a more defensive driving, thus maintaining an adequate level of safety. Nevertheless, failures in obstacle detection may still lead to undesirable behavior such as collisions, emergency maneuvers, or traffic rules violations. For instance, the leading cause of a 2018 AV fatal accident was determined to be a perception error that was not adequately handled [4]. Thus, a deeper understanding of how perception errors affect the AV response is necessary for safety assurance.

This connection between perception errors and AV response can be explored via a holistic testing approach, both on a test track and in virtual environments. In this paper, we concentrate on virtual tests. Virtual testing of AVs by simulations offers a safe and convenient way to validate safety [5]. However, how to effectively include perception modules in the simulation pipeline is an open question. A common approach in the industry is to employ high-fidelity models that represent the real world. In particular, physics-based sensor simulations may generate synthetic sensor signals, which the perception module can process. Besides being highly compute-intensive, this approach is not yet sufficiently developed since simulating sensors at high physical fidelity is challenging. Therefore, we aim to develop a less complex alternative by modeling the functionality of the S&P in terms of perception errors, which we can conveniently integrate into a simulation pipeline. In this article, we provide the following contributions:

- We summarize the state-of-the-art evaluation metrics for AI-based perception algorithms and discuss how they are inadequate for assessing the safety of AVs.
- We present a simulation pipeline that integrates PEMs in a virtual environment as a computationally efficient way of testing Automated Driving System (ADS) capabilities w.r.t. perception errors.
- We present a generalized data-driven approach towards building Perception Error Models (PEMs), using public datasets and Open-Source software.
- We demonstrate the application of PEMs by comparing three different sensor setups, i.e., camera, LiDAR, and fusion of camera and LiDAR. Moreover, we assess the safety provided by these setups in three representative urban scenarios.

Our previous work relied on hand-crafted PEMs, which helped demonstrate a few limitations of evaluation metrics [6]. This article extends that work by introducing a data-driven modeling step and generalizing the procedure. This way,
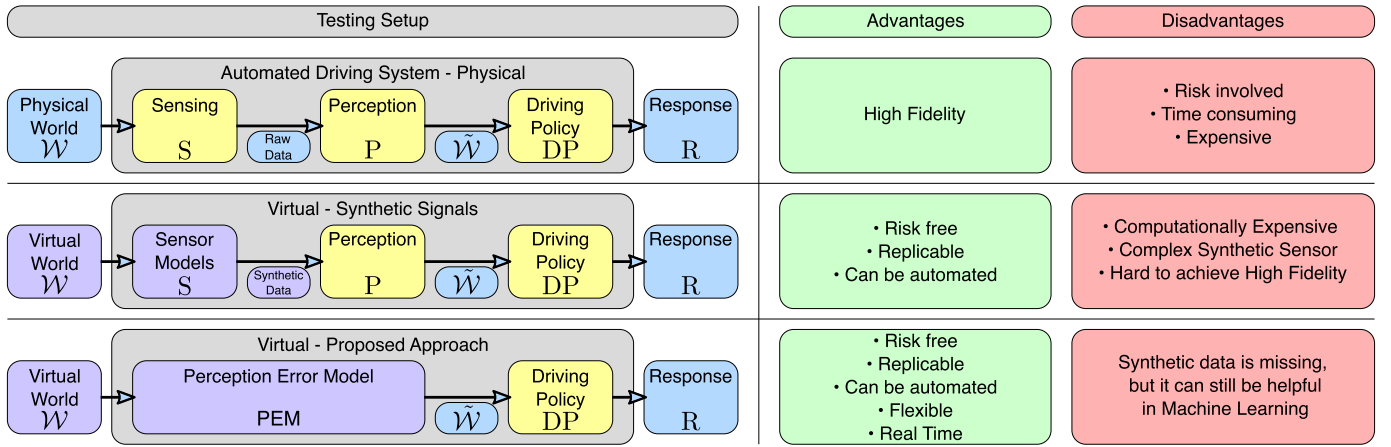
Fig. 1. Comparison of three different testing setups that integrate perception: physical testing (top), synthetic signals in a virtual environment (middle), and our proposed approach involving PEMs (bottom), models that approximate the error of a Sensing and Perception module.

TABLE I

ACRONYMS

Acronym	Meaning
ADS	Automated Driving System
AV	Autonomous Vehicle
CV	Computer Vision
DP	Driving Policy
FoV	Field of View
IoU	Intersection Over Union
mAP	Mean Average Precision
MLE	Maximum Likelihood Estimation
ODD	Operational Design Domain
PEM	Percepion Error Model
S&P	Sensing and Perception
$\mathcal{W}$	World, i.e. surrounding obstacles
$\tilde{\mathcal{W}}$	Perceived World
R	Response

PEMs deployed in simulation experiments provide meaningful results. Thus, we present a complete procedure to investigate the impact of perception errors on safety.

This article is organized as follows. In Sections II and III, respectively, we provide an overview of the literature on Virtual Testing and Perception Evaluation and Modeling. In Section IV, we introduce our proposed definition of PEM, while in Section V, we illustrate this definition with an example of a PEM. In Section VI, we demonstrate how a PEM can be integrated into a simulation pipeline. At last, in Section VII, we discuss our study, and in Section 8, we summarize the contributions and future directions. In Table I we report all the acronyms we adopt throughout the article.

II. BASIC CONCEPTS OF VIRTUAL TESTING

In this Section, we describe our abstraction of an AV. Then, we provide an overview of AV virtual simulations and testing.

A. AV Modules

As illustrated in Figure 1, we describe AVs through three components: Sensing (S), Perception (P), and Driving Policy (DP). The Sensing component S, or sensor setups, comprises all the AV's sensors to sense the surroundings. Standard sensors in the AV field are cameras, LiDAR, and RADAR [7]. The

Perception component P algorithm processes sensor signals to provide the perceived world $\tilde{\mathcal{W}}$ (or object list), which is usually a noisy version of the ground truth world $\mathcal{W}$. With the term Driving Policy DP [8], we refer to the process that determines the response R of the AV given the perceived state of the physical world, i.e., the AV behavior.

B. Virtual Environments and Simulations

Virtual simulations are useful for various reasons (see Figure 1). First, they provide a risk-free environment, which serves to avoid fatalities and damage to properties. Moreover, numerous simulations can be run, and thousands of scenarios can be tested, very often with high repeatability. Therefore, virtual testing is a practical and economical approach for developers and third-party evaluators alike.

Key ingredients for virtual testing are a set of test scenarios that replicate real-life situations that AVs may be exposed to [9], [10], and [11], and measurable safety metrics [12]. To assess AVs by virtual testing, evaluators compute safety metrics from the simulated AV behaviors, such as safety clearance distances (between AV and other traffic participants), speed limits, or specific traffic rules violations. Thresholds on those safety metrics demark safe from unsafe behavior. The major drawback of virtual testing is its potential lack of fidelity. Key constraints here include the modeling effort and computational costs in achieving a real-time integrated virtual environment simulation with an extremely high level of detail. These challenges mainly concern physical characteristics and physical phenomena, the behavior of other road users, and noise characteristics for sensors and vehicle motion. Hence, one of the significant challenges is replicating a realistic sensor input to the vehicle under test. Synthetic signals are commonly utilized in commercial [13], [14], [15], [16] and Open-Source simulators [17], [18]. Their fidelity is affected by both the quality of the sensor model and the quality of the virtual environment model. Additionally, sensor models are computationally expensive, as synthetic signals potentially increase in size and details. For example, a synthetic camera must generate images at the exact resolution and rate of the

modeled physical camera. Therefore, this approach does not scale well due to hardware requirements.

However, synthetic signals are also a promising approach to solve other problems, such as data generation for machine learning tasks. For example, Cortes et al. [19] have exploited synthetic signals to develop perception algorithms that perform better on real data. However, Talwar et al. [20] trained and tested perception algorithms on synthetic and real datasets. They have demonstrated that the trained models do not transfer well between real and synthetic datasets.

III. PERCEPTION EVALUATION AND MODELING

Since we aim to test S&P systems, we present state-of-the-art perception evaluation metrics in this Section. Next, we investigate related works on sensor modeling.

In the Computer Vision (CV) field, perception software is often evaluated as an independent component. We can divide perception errors into four categories:

- **Detections**: Detecting the surroundings can be affected by false positives and false negatives;
- **Misclassification**: Even if detected, any object can be associated with the wrong class.
- **Object parameters**: Any parameter can be potentially affected by measurement errors. Physical parameters with a continuous domain (e.g., size, position) inherently involve a degree of precision and accuracy.
- **Dynamics**: Each error can evolve. Detected objects can be lost (tracking error), a class label could be changed. Furthermore, object parameters can physically change (e.g., moving objects), making the error dynamic.

A. Evaluation Metrics

By comparing the ground truth and the CV algorithm output, we can measure the magnitude of all error types and summarize them into a single metric. CV algorithms developed for generic object detection tasks are often evaluated by Intersection over Union (IoU) and Mean Average Precision (mAP) metrics [21], [22]. Similarly, tracking algorithms are evaluated with Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) [23]. These metrics are developed without considering the deployment context of the algorithms so that they apply to any domain. Thus, these metrics are unaware of the scene's composition and hence are context-independent. However, for AV applications, the composition of the urban scene affects whether specific errors (e.g., misdetections) are critical.

We observe that the most desirable property of S&P is to provide *sufficient* information to DP to afford a safe reaction to the surroundings. A perception error is critical only if it leads to unsafe behavior and is negligible otherwise. For example, while undesirable, misclassifying an obstacle on the road is not critical if the AV stops or avoids it properly. On the other hand, the same error in a different context may lead to an unacceptable response or even a collision. Addressing this limitation, studies such as [24], [25], [26], and [27] propose metrics that integrate context-specific criteria in the evaluation.
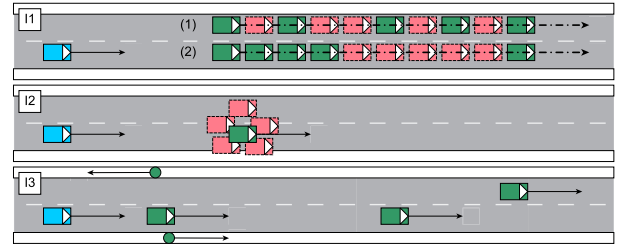

Fig. 2. Illustration of critical issues I1, I2, I3. We indicate the ego vehicles in blue, other objects in green, and detections in red. (I1) Temporal Relevance: The detection timeline of the vehicle in lane 1 is more sporadic and unstable, but non-detection intervals are shorter than the vehicle in lane 2. (I2): Overlap Sensitivity: A detection bounding box can be closer, farther, or on the side relative to the actual vehicle. (I3) Relevance of the objects: Many objects are present. In each of these examples, the context of the error affects its severity.

We highlight four aspects regarding the connection between perception metrics and overall AV safety (see Figure 2), which are out of the scope of metrics derived from IoU and mAP:

- **I1: Temporal Relevance**: AVs operate in a dynamic environment, where the state changes continuously and decisions are time-sensitive. The same error, e.g., misdetection, can have a different impact on safety based on the time interval it manifests on and how it evolves. Long intervals of misdetection may generate slightly higher safety concerns than sporadic episodes.
- **I2: Overlap Sensitivity**: When detected, the coordinates of an obstacle are analyzed to determine the course of action (e.g., avoid, follow at a distance). A smaller magnitude of spatial errors is desirable, but limiting the evaluation to the bounding box overlap (IoU) alone may not be informative in terms of safety.
- **I3: Relevance of the objects**: While driving on the road, AVs are surrounded by many objects, road users, traffic signals, and traffic lights. At any specific time, only a subset of these objects is relevant to DP. Furthermore, the relevance varies over time, as relative distances, directions, and the potential of a collision change continuously.
- **I4: Impact on Driving Policy**: Perception Modules deployed in AVs directly provide their output to the Driving Policy. Hence, the strengths and weaknesses of the two components are closely tied together, and evaluating the first without accounting for the second is shortsighted. For example, a DP may react poorly to misclassifications and robustly against false positives, while another may behave oppositely.

While novel, safety-aware perception metrics could address the first three considerations, no metric can be easily *generalized* to any arbitrary Driving Policy, as different driving styles may have different demands in perception quality. Thus, we approach this problem by employing virtual testing where we directly integrate the DP in the pipeline.

B. Modeling Errors

Sensor error (or noise) modeling is a valuable technique in signal processing. In fact, by applying integration algorithms, it is possible to drastically increase the sensors' viability. For example, Kalman filters assume Gaussian noise and linear

TABLE II
RELATED WORK ON SENSOR MODELING

Author	Method	Sensors	Dataset
Arnelid et al.[28]	RC-GAN	–	Private
Berkhahn et al. [29]	RODEs		
Hirsenkorn et al. [30]	Non-parametric	Radar	Private
Krajewski et al. [31]	NN	LiDAR	Private
Mitra et al. [32]	RNN	Camera	KITTI [33]
Innes et al. [34]	NN	–	KITTI [33]
Sadeghi et al. [35]	NN	–	Synthetic
Zec et al. [36]	AIOHMM	–	Private
Zhao et al. [37]	LiDAR equation	LiDAR	Private

dynamical models. In our context, we are interested in modeling the error *after* the perception layer, i.e., at the object level. Hoss et al. [38] provide an extensive review of the state-of-the-art of this specific task. The authors highlight the lack of shared procedures and datasets.

In Table II, we summarize recent works on sensor error modeling. Mitra et al. model 2D bounding boxes errors in a camera-based system by learning Recurrent Neural Networks [32]. Conversely, Hirsenkorn et al. model distance measuring error of a RADAR-based system employing a non-parametric approach [30]. We note that the different nature of the models makes quantitative comparisons not only unfeasible but also meaningless. Even if targeting the same sensor, Krajewski et al. [31], and Zhao et al. [37] follow two very different approaches to LiDAR modeling. Krajewski et al. [31] model the sensor error as the difference between the object states [31]. Notably, they employ drones to collect the Ground Truth, which is then compared to the object states perceived by a LiDAR-based system. Zhao et al. [37], instead, model the LiDAR physics to compute the position of each point in the point cloud. Nevertheless, the authors also approach the object level by providing a *target mode*. Instead of the whole point cloud, they generate a single point for each object in the scene. This solution provides a slightly faster implementation, as the size of the synthetic data each step generates is drastically reduced. However, this approach cannot easily include other sensors, as the model is based on LiDAR physics. Moreover, the model does not imply a perception layer with fusion and tracking in the pipeline. To address this aspect, Zec et al. [36] and Arnelid et al. [28], propose sensor-agnostic models. Both of their model, Autoregressive Input-Output Hidden Markov Model (AIO-HMM) [36] and Recurrent Conditional Generative Adversarial Networks (RC-GAN) [28], model object parameters such as location and size with time dependencies. These solutions are more suitable for AV virtual testing since AVs rely on multiple sensors of diverse natures. The authors can also compare their results, as they analyze the same private dataset and have access to both models. More recently, Sadeghi et al. [35] adopt PEMs proposed in our previous work [6]. Their implementation relies on a Neural Network trained on synthetic data and a LiDAR-based object detector. Moreover, they demonstrate the efficiency and effectiveness of using PEMs as surrogates for S&P systems using a collection of scenarios from the CARLA Autonomous Driving Challenge [39]. Innes and Ramamoorthy [34] also employ PEMs in a simulation study to estimate rare failure

probabilities in automated emergency braking scenarios. The authors argue that a photorealistic simulation is not necessary to study the AV behavior as long as the input to the DP is affected by the same error distribution as the real one. Berkhahn et al. [29] conducted a simulation study injecting errors, modeled with random ordinary differential equations (RODEs), in the simulation pipeline to study their impact on intersection conflicts. Interestingly, the perception errors modeled are intended for human drivers and not for an S&P of an AV.

In summary, various approaches for modeling perception errors have been proposed in the literature, such as probabilistic models, neural networks, non-parametric models, or physics-based models. However, we note that most of them rely on private datasets, preventing a direct model comparison. Furthermore, most of these studies are still closely tied to specific sensors (e.g., camera, LiDAR) and model them individually. However, Hanke et al. present a modular architecture where individual sensor models are combined and integrated into a simulation pipeline [40].

In contrast to all those existing approaches from the literature, we model the S&P as a whole, including the sensors and the AI processing of the sensor data. Our approach is not limited to specific sensors or AI algorithms, since we model errors in a generic way that applies to any particular perception system. Thus, our approach shifts the perspective from sensor models to perception error models (PEMs).

Moreover, existing studies have not yet introduced a comprehensive and flexible pipeline for simulation experiments.

IV. PERCEPTION ERROR MODELS

In this Section, we propose a generalized definition of Perception Error Models. We propose a sensor-agnostic definition, which abstracts from the modeled sensors and, thus, has a standard interface. This assumption is helpful for flexible integration in a simulation pipeline, which would greatly benefit the field if adopted. Additionally, we summarize some general considerations for error modeling.

A. Generalized Notation and Definition of PEM

We denote the collection of n surrounding Ground Truth (GT) objects (obstacles or road users) $\mathcal{W} = \{o_1, \ldots, o_n\}$. The S&P observes the "world" $\mathcal{W}$ and generates the perceived world $\tilde{\mathcal{W}} = \{\tilde{o}_1, \ldots, \tilde{o}_m\}$, i.e., the collection of the m perceived objects. Next, the Driving Policy DP generates a response R by analyzing $\tilde{\mathcal{W}}$. Real objects o and perceived objects $\tilde{o}$ are represented with an array of parameters. Depending on the S&P system, this array may include the class (e.g., vehicle, pedestrian) and the 3D bounding box of the objects, described by its location (x,y,z), orientation (yaw, pitch, roll), and dimensions (length, width, height). Moreover, an object may also include parameters such as velocity, brake and turning indicator status, or even an activity label (e.g., parking, running, crossing). Depending on the system, we can extend this notation to include other road elements or a more refined classification that discriminates between cars, bikes, trucks, or other classes. Generally, some S&P systems may assume a

flat surface, especially if the system cannot measure the depth or simplify vehicle dynamics. Such assumptions reduce the number of relevant parameters, e.g., assuming pitch and roll equal to 0.

In Figure 1, we note RawData = $S(\mathcal{W})$ and $\tilde{\mathcal{W}}$ = P(RawData). We can observe that:

$$\tilde{\mathcal{W}} = P(S(\mathcal{W})) = S\&P(\mathcal{W}) = \mathcal{W} + \mathcal{E}, \qquad (1)$$

where $\mathcal{E}$ is the error between $\tilde{\mathcal{W}}$ and $\mathcal{W}$. The response R determines the AV *behavior* and, therefore, the overall safety and performance of the autonomous system:

$$R = DP(\tilde{\mathcal{W}}) = DP(\mathcal{W} + \mathcal{E}). \qquad (2)$$

In this article, we define Perception Error Model (PEM) as an approximation of the combined function of sensing subsystem S and a perception subsystem P:

$$PEM(\mathcal{W}) \approx S\&P(\mathcal{W}) = \tilde{\mathcal{W}} = \mathcal{W} + \mathcal{E}. \qquad (3)$$

This definition outlines the interface and role of the model. A PEM receives the ground truth world $\mathcal{W}$ and returns the perceived world $\tilde{\mathcal{W}}$.

B. Error Modeling Considerations

We outline four key factors that alter the manifestation of perception errors $\mathcal{E}$ in AV, which should be considered while designing a PEM.

1) Positional Aspects: The relative position of any $o_j \in \mathcal{W}$ w.r.t. the ego-vehicle slightly affects the quality of the S&P response [7]. Sensors have limited Field of View (FoV), e.g., limited range, and blind spots. The relative distance also affects the signal strength and resolution. Furthermore, S&P systems usually perform better in areas where the FoV of multiple sensors overlap, as the signals from the sensors might be fused to generate the perceived world $\tilde{\mathcal{W}}$.

2) Parameter Inter-Dependencies: Object parameters **X**, **C** are clear players in $\mathcal{E}$ manifestation [41]. For example, larger objects are more likely to be detected, slow-moving vehicles are easier to track, or classification errors affect size estimation. Additionally, even parameters not relevant to DP may be crucial, such as the color or material of the object [7]. For example, LiDAR and radar have degraded performances on dark/non-reflective surfaces or metallic objects.

3) Occlusion: Objects on the road may influence the detection of other objects. For instance, large objects (e.g., trucks) may occlude smaller objects (e.g., cars or cyclists). Occlusion has much impact on the reliability of perception systems. Cumulative statistics computed over both occluded and non-occluded objects may yield a biased assessment of the actual performances of an S&P module. It is recommended to report performance statistics for different levels of occlusion.

4) Temporal Aspects: Objects often move in the scene. For example, a previously occluded object may become not occluded, and vice versa. Additionally, algorithm uncertainties and filtering techniques are affected by their previous state. Hence, the error $\mathcal{E}$ affecting an S&P also evolves.

C. Validation

The objective of PEMs is to inject perception errors into the simulation pipeline. Thus, their validity is achieved if and only if the AV behavior generated embedding a PEM is comparable to the behavior the same AV would generate employing an actual S&P. Sadeghi et al. [35] use synthetic data to tune their model. Thus, they can directly compare the resulting behaviors, as the sensing module (synthetic) and the vehicle dynamics (simulator) are exactly the same in both approaches. However, if the PEM is tuned on real data, a proper validation would require physical testing to validate the PEM and the vehicle dynamics embedded in the simulator. Thus, a more straightforward but arguably less effective validation ensures that the errors have the same magnitude and distribution as the S&P that the PEM is replacing.

V. DATA DRIVEN PEM

As mentioned in subsection III-B, most studies rely on private datasets. Our approach, instead, relies on a public dataset and open-source algorithms. We define a perception dataset as a collection S of scenes s. Each scene is a stream of pairs $(\mathcal{W}_t^s, \tilde{\mathcal{W}}_t^s)$, where $t = 1, 2, \ldots T$ is discrete time.

To generate this dataset, we have considered several CV datasets, including KITTI [33], ApolloScape [44], nuScenes [43], Lyft Level 5 dataset [45], and Waymo open dataset [46]. Such datasets are mainly used to design perception systems (P), which are assessed by comparing the generated $\tilde{\mathcal{W}}$ with the ground truth $\mathcal{W}$. By contrast, we consider an existing P, i.e., the open-source P system from Baidu Apollo [42], and design the PEM by modeling the error $\mathcal{E}$ as the difference between $\mathcal{W}$ and $\tilde{\mathcal{W}}$. We opted for nuScenes as the chosen example to illustrate our methodology due to the inclusion of RADAR data, rich annotations (used as ground truth $\mathcal{W}$), and data collected in Singapore and Boston. Only a few implications derive from our selection of a public dataset and a perception algorithm. To avoid any bias, we highlight such implications. The nuScenes choice guided us in the approach regarding the model of occlusion. This approach can be easily replicated on any dataset by computing the occlusion levels similarly. Nevertheless, a different model of occlusion can be explored. The choice of Apollo, instead, has a substantial implication on the implementation. While modeling S&P as a whole, the PEM's output must follow the original perception module P output format. With the described setup, the developed PEM models a hypothetical vehicle that employs the Apollo perception module P and the nuScenes sensors setup S, including a frontal camera, LiDAR, and RADAR.

A. Perception Dataset

Figure 3 illustrates the procedure to generate the perception dataset. The 1000 scenes composing nuScenes, 20 seconds streams on recorded sensor data, are annotated with ground truth $\mathcal{W}$ at 2Hz. We convert $\mathcal{W}$ in the same format and frame of reference of $\tilde{\mathcal{W}}$. Then, we adapt the raw sensor data and prepare the message streams for the Apollo perception module. To exchange data between modules, Apollo employs
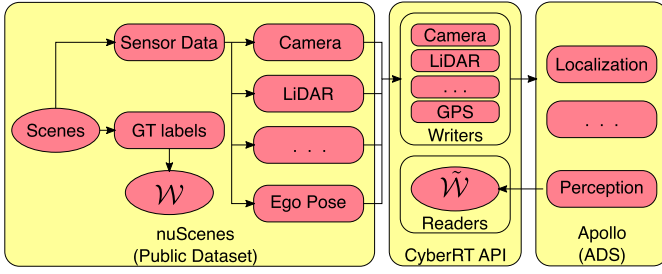
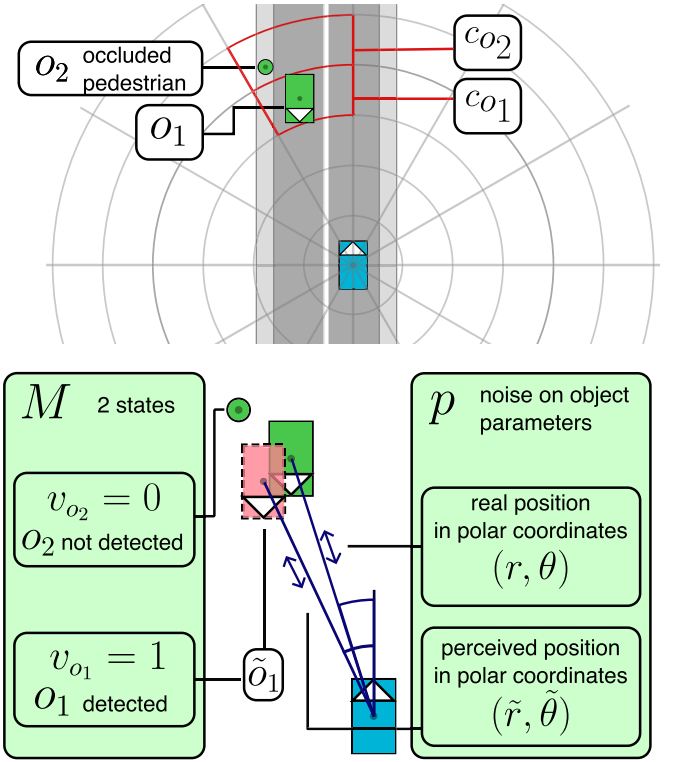Fig. 3. Software pipeline to generate $\tilde{\mathcal{W}}$ by means of Apollo [42] and nuScenes [43].

Fig. 4. Example of a frame and corresponding variables in the PEM. Above we depict the labeled ground truth. Objects are positioned on a polar grid. The different sensors may cover the different zones of the polar grid with a different degree of reliability. Below the detection state and the noise on object parameters are determined via M and p, respectively.

Cyber-RT [47], a ROS-like middleware. Hence, we can utilize Cyber-RT APIs to define writer and reader nodes. In particular, we prepared as many writers as per sensor type, each to synchronously replay raw data to replicate a nuScenes scene. While replaying each scene, the Apollo perception module is activated and processes the raw data, providing the output $\tilde{\mathcal{W}}$. A Cyber-RT reader logs to file the generated $\tilde{\mathcal{W}}$. Next, it is possible to compose the perception dataset using $\mathcal{W}$ from nuScenes annotations and the recorded $\tilde{\mathcal{W}}$ for each scene, obtaining a stream of pairs $(\mathcal{W}_t^s, \tilde{\mathcal{W}}_t^s)$ for each scene.

Lastly, any multi-object detection task requires solving a matching problem. In each frame, both $\mathcal{W}$ and $\tilde{\mathcal{W}}$ are composed of multiple objects. Establishing a connection between items from one set to the other is necessary to measure any error. Bernardin et al. provide a well-established solution [23] common in the CV field. Alternatively, Arnelid et al. [28] and Zec et al. [36] relied on the solution proposed by Florback et al. [48]. All these matching algorithms are based on similarity functions. Thus, it is essential to highlight that any parameter and procedure defined in this task will alter our error models. In particular, we imposed a threshold of 10m as the maximum distance between o_i and $\tilde{o}_j$ to allow a match.

B. Learning PEMs

In this section, we explain how to learn a PEM from data, using the dataset presented in subsection V-A. Given the generalized PEM definition (Equation 3), a PEM can be implemented in various ways, and the goal is to generate $\tilde{\mathcal{W}}$ given $\mathcal{W}$. For the scope of this article, we design our model to be easily relatable to basic evaluation metrics such as detection rates and IoU, as well as to provide a clear example of ways to address the considerations provided in subsection IV-B.

In particular, we model the detection of each object $o_i \in \mathcal{W}$ with a binary state variable v, with $v = 1$ when the object is detected (hence, a corresponding perceived object $\tilde{o}_i$ exists), and $v = 0$ otherwise. In addition, we model the error ε on the parameters of the detected objects. We parameterize the objects by their location, expressed in polar coordinates (radial r, angular θ), and their perceived values as $(\tilde{r}, \tilde{\theta})$. The proposed PEM operates in two steps when integrated into the simulation (see Figure 4). First, it processes each object $o_i \in \mathcal{W}$ and determines whether it is detected, generating the corresponding perceived object $\tilde{o}_i$. Second, it calculates the

magnitude of error ε on the parameters of $\tilde{o}_i$. The PEM then composes $\tilde{\mathcal{W}}$ as the set of all $\tilde{o}_i$, and returns it to the simulator.

We model the evolution of the state v by a Hidden Markov Model (HMM) with Gaussian emissions. In particular, a transition matrix $\mathbf{A}$ models the state variable v, and the emission probabilities model the error ε on the object's parameters $(\tilde{r}, \tilde{\theta})$. We highlight that the state variable v is observed in the dataset, which greatly simplifies the learning task. Moreover, we do not introduce emissions for non-detected objects, since we do not need to generate errors on their parameters. To account for the various factors that can affect the performance of the S&P performance (see subsection IV-B), we propose dividing the model into several partitions based on external conditions $c \in C$. Each partition is parameterized according to the relative data in the dataset. For example, a zone-based partitioning (as shown in Figure 4) generates partitions trained specifically on objects located in the corresponding zone.

We represent an input object o_i in the scene with three variables $o_i = (r, \theta, c)$, i.e., the polar coordinates (radial r, angular θ), and the corresponding values of $c \in C$. We describe perceived object $\tilde{o}_i$ with three state variables $\tilde{o}_i = (\tilde{r}, \tilde{\theta})$. With $\tilde{\rho}$ and $\tilde{\phi}$, we indicate the perceived polar coordinates, i.e., with error. We use the notation o_{it} to indicate the state of the object o_i at time t.

We implement a PEM as a set of $|C|$ pairs:

$$\text{PEM} = \{(M^c, p^c), c \in C\}, \tag{4}$$

where:

- C is the set of external conditions c that alter the perception quality. Following the considerations in positional aspects and occlusion, we define C as the cross-product of a grid cell partition of the ego-vehicle surroundings and the set of occlusion levels. We can easily follow nuScenes occlusion modeling. The nuScenes dataset defines the four levels of visibility, i.e., the fraction of the annotation (i.e., the ground truth) visible from the ego-vehicle. It is binned into 4 bins: $0 - 40\%$, $40 - 60\%$, $60 - 80\%$, and $80 - 100\%$. Hence, our set C is generated as follows:

$$C = \{\text{vis}_0, \text{vis}_1, \text{vis}_2, \text{vis}_3\} \times \{\text{grid cells}\}, \quad (5)$$

where the polar grid is obtained with 30 deg sectors and radius increments of $10m$.

- for each $c \in C$, M^c is the Markov Chain that models v, i.e., whether an object is detected. Given the considerations about temporal aspects, each M^c is implemented by the transition matrix $\mathbf{A}^c$ to determine v:

$$\mathbf{A}^c = \begin{bmatrix} a_{00}^c & a_{01}^c \\ a_{10}^c & a_{11}^c \end{bmatrix}, \quad (6)$$

where the transition matrix parameters vary with C, as c refers to o_{it} values.

The probability of an object $\tilde{o}_i$ at time t is obtained via Maximum Likelihood Estimation (MLE):

$$P(v_{it} = l | v_{i,t-1} = k, o_{it}) = \mathbf{A}_{kl}^c = \frac{w_{kl}^c}{\sum_{m \in D} w_{km}^c}, \quad (7)$$

where w_{kl}^c are the transitions from state $v = k$ to $v = l$ that occur in $c \in C$ in the dataset $(\mathcal{W}_t^s, \tilde{\mathcal{W}}_t^s)$ (see from subsection V-A).

- p^c describes the parameter error ε distribution, i.e., the difference between perceived and ground truth parameters:

$$\varepsilon = (\varepsilon_{\tilde{r}}, \varepsilon_{\tilde{\theta}}), \quad (8)$$

$$\varepsilon_{\tilde{r}} = \frac{\tilde{r}}{r}, \quad (9)$$

$$\varepsilon_{\tilde{\theta}} = \tilde{\theta} - \theta. \quad (10)$$

If object o_{it} is detected, the object $\tilde{o}_{it}$ parameters $\left(\tilde{r}, \tilde{\theta}\right)$ are generated with error distributed as a multivariate Gaussian distribution $\mathcal{N}^c$ over the parameters r, θ of o_i:

$$(\varepsilon_{\tilde{r}}, \varepsilon_{\tilde{\theta}}) \sim \mathcal{N}(\mu^c, \Sigma^c), \quad (11)$$

where $\mu^c = (\mu_r^c, \mu_\theta^c)$ and Σ^c represent means and covariances on the error for objects o_i with $c \in C$.

This model is entirely sensor agnostic, as no parameter is tied to any specific sensor or sensor output (e.g., point cloud, 2D bounding boxes).

Thus, we need to estimate seven parameters for each $c \in C$. First, each model M^c has 2 degrees of freedom. We can estimate a_{11}^c and a_{01}^c, and since each row sums to 1, we can derive a_{10} and a_{00} accordingly. Second, each distribution p^c
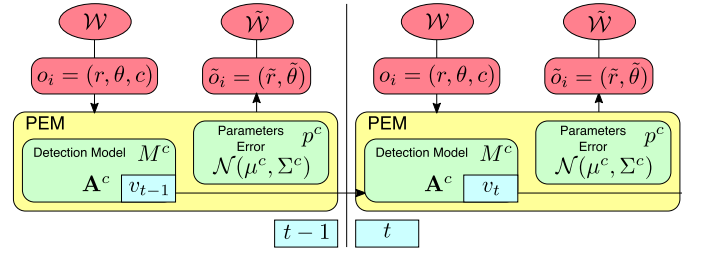

Fig. 5. Illustration of 2 steps (frames) of PEM we deploy in our experiments.

has 5 parameters: the means (μ_r^c, μ_θ^c), and the parameters $(\sigma_r^c, \sigma_\theta^c, \sigma_{r,\theta}^c)$ in Σ^c:

$$\Sigma^c = \begin{bmatrix} \sigma_r^c & 0 \\ 0 & \sigma_\theta^c \end{bmatrix} \begin{bmatrix} 1 & \sigma_{r,\theta}^c \\ \sigma_{r,\theta}^c & 1 \end{bmatrix} \begin{bmatrix} \sigma_r^c & 0 \\ 0 & \sigma_\theta^c \end{bmatrix}, \quad (12)$$

where $\sigma_{r,\theta}^c$ is the correlation value.

Similarly to evaluation metrics, these parameters can be obtained analytically via MLE for each $c \in C$ independently. However, a partitioned model approach introduces two main drawbacks. First, independently estimating parameters in each partition reduces the amount of data considered. Second, zone-based partitioning discounts prior knowledge of zone adjacency, which is important because objects are rarely located in only one zone. This problem is typical in spatial data analysis, which aims to identify regional patterns (e.g., pollution levels). To this end, hierarchical models such as Conditional Autoregressive (CAR) models are often employed [49]. CARs embed such zone adjacency knowledge in the parameter estimation. In particular, they assume that the parameter y_c we want to estimate (e.g., $\mathbf{A}_{kl}^c$, μ^c) is conditioned to neighboring values of $y_{n,n \neq c}$. The hierarchical nature of the model allows us to define the conditioning in a probabilistic manner:

$$y_c | y_n, n \neq c \sim \mathcal{N}\left(\alpha \sum_{n=1}^{|C|} b_{cn} y_n, \tau_n^{-1}\right), \quad (13)$$

where τ_n^{-1} is the precision at each partition, with prior $\tau_n \sim \Gamma(1, 1)$. We describe the adjacency structure in the matrix b. In particular, we consider a partition adjacent to the other partitions that share an edge in the zone-based partitioning and have the same occlusion level.

A priori, we assume that each partition's most probable value is the average of the neighbors' values. However, this approach also admits slightly different values, but only if the data is sufficient to support such a hypothesis. Intuitively, the CAR model acts as a smoothing technique that considers the amount of data to determine the smoothing factor. For example, partitions with no samples will converge to the average of the neighbors' values. In contrast, partitions with a high number of samples will converge the value supported by the samples.

The adoption of CAR models makes the parameters no longer obtainable directly via MLE, since the partitions are no longer independent. Therefore, we employ the probabilistic programming framework PyMC3 [50], and define a CAR model for each of the seven parameters of the PEM. Next,
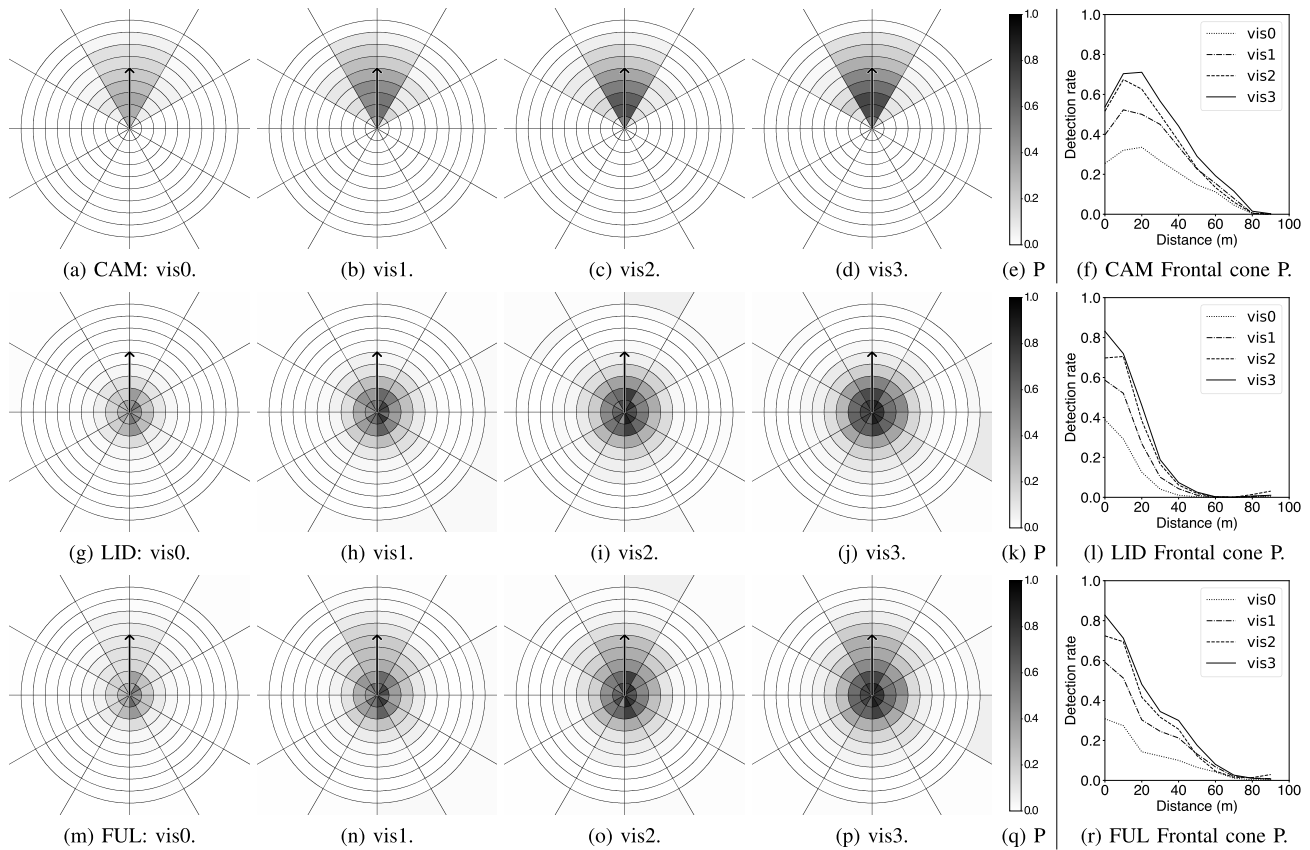
Fig. 6. Probability P of detection for nuScenes ground truth object processed by Apollo perception module. Objects are located around the ego vehicle positioned in the center of the polar grid (radius increments of 10m and angle increments of 30 degrees). The ego vehicle faces upward, as indicated by the arrows. We can observe how P values outline the effective FoV of each sensor setup CAM, LID, and FUL. Moreover, nuScenes occlusion levels have a clear impact on P. As expected, less occluded object (vis3) lead to generally higher detection rates across all grid cells. Figures 6f, 6l, and 6r depict the decay of P in the frontal cone for each visibility level based on the sensor setup.

TABLE III
LIST OF PEMs OBTAINED USING THREE DIFFERENT SENSOR SETUPS

ID	Sensor configuration
CAM	Only frontal camera data.
LID	Only Lidar data, i.e., only the point cloud.
FUL	Full setup, combining camera, LiDAR, and RADAR data.

we apply the PyMC3 inference framework to fit the parameters employing the dataset $(\mathcal{W}_t^s, \tilde{\mathcal{W}}_t^s)$. In particular, we apply the Maximum a Posteriori method to obtain point estimates of the PEM parameters.

C. Examples

During the generation of a perception dataset illustrated in subsection V-A, we can easily decide which data, e.g., images or point clouds, we feed to the Apollo perception module (see Figure 3). For the experimental scope of this paper, we selected three different sensor setups, summarized in Table III. We select these three configurations as they often occur in practice. Moreover, in this way, we can test several PEMs. In Figure 6, we show the most direct metric, i.e., detection rate, in the context of PEM under each of the chosen sensor setups. We present a diagram for each element in C. Thus, we depict a polar grid for each occlusion value. Within each of the delimited zones, we can show how a single parameter or a metric varies. Detection rates are the most informative and intuitive to visualize as they depict

the effective Field of View of the S&P. In Figures 6f, 6l, 6r, we observe how the CAM detection rate has a slower decay in the frontal cone, followed by FUL and LID. The fusion algorithm present in FUL improves on LID beyond the 30m range but only partially exploits camera data at more considerable distances.

VI. SIMULATION EXPERIMENTS

In this section, we present the simulation testing setup aimed at investigating different PEMs, not in raw perception metrics but in terms of their impact on safety. To this end, we adopted SVL simulator [18] and Apollo 5.0 [42] as ADS under test. Our choice is driven by the open-source nature of the tools, which facilitates our customization, and the reliable co-simulation offered by the SVL bridge towards the CyberRT middleware employed by Apollo. We employed ViSTA framework for testing automation [10]. The framework comprises Python scripts that implement the defined scenarios, control SVL actors in a deterministic manner, and log the results. We extended ViSTA and SVL simulator by integrating PEMs in the pipeline, as illustrated in Figure 7 and Figure 8. We prepared a custom sensor in the SVL simulator that connects via socket to a local Python server to send $\mathcal{W}$. The server implements the PEM and processes $\mathcal{W}$ to generate $\tilde{\mathcal{W}}$ (see Equation 3) with the determined frequency.
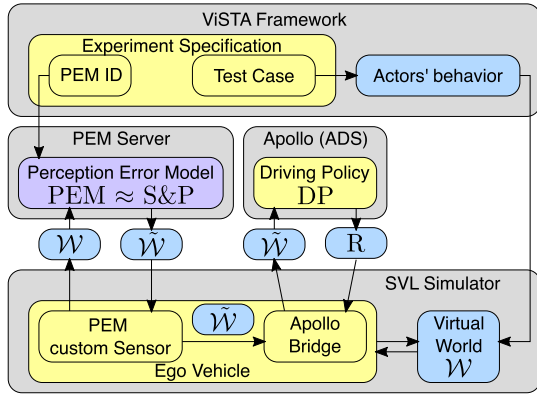
Fig. 7. Software architecture for virtual testing with PEM integration. The ViSTA framework automates testing by configuring the other components.

This solution provides access to many Python statistical and machine-learning tools.

A. List of Scenario-Based Experiments

We have defined a set of experiments, as illustrated in Figure 7. Each experiment combines a specific Test Case (a Scenario with instantiated parameters) and a PEM. In this article, we compare the 3 PEMs previously introduced (see Table III) against three representative urban driving scenarios (see Figure 9).

We executed each of the nine resulting experiments 500 times to account for the randomness introduced by the PEMs. Moreover, we ran each scenario 250 times with error-free perception (baseline), resulting in the ground-truth object list.

1) Test case TC1: The ego vehicle drives on a straight road. After 400m, a pedestrian jaywalks in front of the vehicle. The ego vehicle has to slow down, stop, or perform an evasive maneuver to avoid the collision. We configured the pedestrian in the simulations so that it causes a collision if the ego vehicle does not react.

2) Test case TC2: The ego vehicle drives on a straight road and approaches a leading traffic vehicle driving at 7m/s. The ego and traffic vehicles proceed for 500m until reaching the traffic light. The ego vehicle has the option to overtake the leading vehicle.

3) Test case TC3: A combination of both previous scenarios. This scenario may also involve occlusion, as the leading vehicle could hide the potential pedestrian detection. While the leading vehicle follows the same script as TC2, the pedestrian starts walking only when this leads to a collision with the ego vehicle.

These scenarios have been selected as they represent some of the most basic interactions between the ego vehicle and a pedestrian (TC1) or another vehicle (TC2) while involving a potential risk of a collision. Moreover, these scenarios require a similar response to avoid collision.

B. Experimental Results

In this section, we report the outcome of our experiments, demonstrating the scope of analysis afforded by introducing

Fig. 8. Screenshot of the SVL-Apollo co-simulation, including PEM. Bounding boxes in $\tilde{\mathcal{W}}$ are consistent between Apollo (left) and SVL (right), even undetected objects.
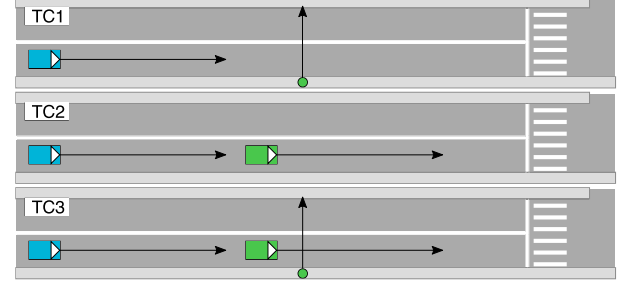

Fig. 9. Illustration of the three scenarios in our experiments. TC1: jaywalking pedestrian, TC2: following a vehicle, TC3: the combination of both.

TABLE IV

SUCCESS RATE BY COMBINATION OF TEST CASE AND SENSOR SETUP

Sensor Setup	TC1		TC2		TC3	
	<1m	>1m	<1m	>1m	<1m	>1m
CAM	99.2%	0.8%	61.6%	38.4%	59.0%	41.0%
LID	0.0%	100%	0.0%	100%	0.0%	100%
FUL	0.0%	100%	0.0%	100%	0.0%	100%

PEMs in the simulation pipeline. We compare the effect of deploying the three PEMs shown in subsection V-C, learned by varying the sensor setup. Hence, we can observe the safety achieved by each different S&P. For ease of understanding, we limit the analysis to one of the most straightforward safety metrics: the minimum distance between the ego vehicle and the obstacle (vehicle or pedestrian) during a complete test case run. Intuitively, we can consider a distance < 1m as severe as a collision. Table IV summarize the success rate of each experiment in three different levels of safety based on the minimum spatial distance achieved in each run.

Additionally, Figure 10 depicts the whole set of simulated runs. The vertical axes represent the chosen safety metric (minimum spatial distance) in all plots, while the horizontal refers to a perception metric. We highlight two perception metrics, *Relative Detection Frequency*, i.e., the rate of detection of the obstacle, and *Max non-Detection Interval*, i.e., the length of the most extended time interval without a detection. We computed the relative detection frequency of the obstacles only when they were within 100m of the ego vehicle. This consideration prevents skewed results caused by the amount of time the ego-vehicle spends in the obstacle's proximity compared to the simulation's overall duration.

1) TC1: This scenario is the most challenging, and CAM setup cannot perform safely. Upon further inspection, we noticed a high positional error in the CAM model that prevents the detection of the potential collision, despite the
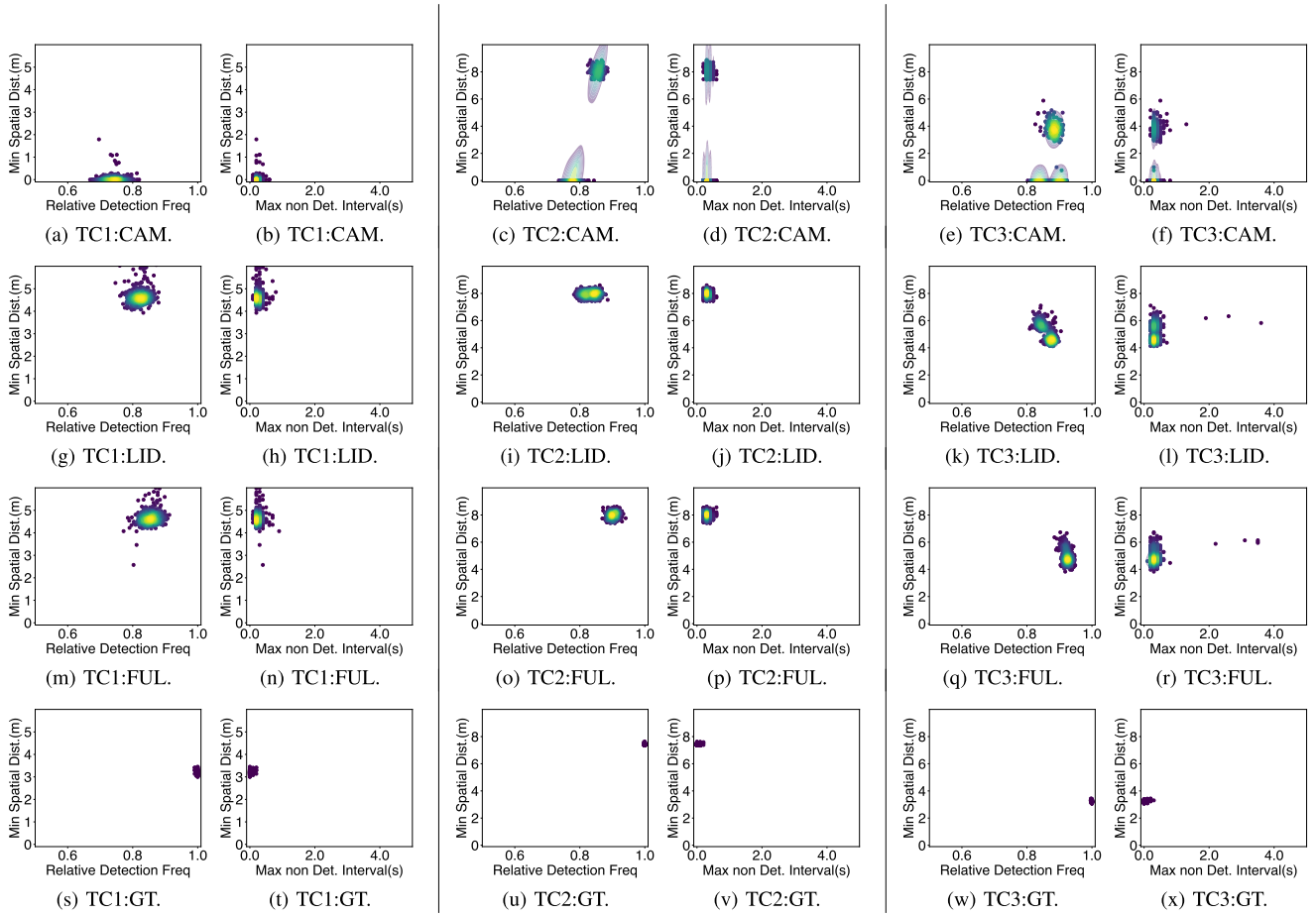
Fig. 10. Density scatter plots outlining results for each experiment, i.e., a combination of Test Case and PEM. Columns 1-2: TC1, columns 3-4: TC2, columns 5-6: TC3. Moreover, each row is associated with a particular sensor setup: CAM, LID, FUL, and GT. Each couple of plots (e.g., columns 1-2, in the first row) depicts the 500 simulation runs executed under that combination of Test Case and PEM. The density of samples is represented on a color scale from blue (low density) to yellow (high density).

pedestrian being detected at a similar rate compared to LID and FUL.

2) TC2: In this scenario, the AV employing CAM setup presents two distinct behaviors. It can perform safely in 38.4% of the runs.

3) TC3: Interestingly, the scenario combination of the previous allows for a safer outcome for all setups. In particular, the presence of 2 potential obstacles promotes a slower driving style, reducing the amount of collisions.

C. Discussion

We bypass all the considerations presented in section III by employing virtual testing instead of perception metrics to evaluate perception. We can evaluate the overall S&P by associating it with its safety rather than cumulative metrics with no contextual awareness. Moreover, we are testing a specific Decision Policy implemented by the Decision Module employed in Apollo, which directly addresses I4. This result is not readily achievable without PEMs, as simulations based on synthetic data are in most cases too computationally expensive and do not provide any direct insight into perception performance (e.g., Figure 6).

In subsection III-B, we presented some state-of-the-art modeling techniques for perception errors. However, given the relatively under-explored nature of the field, those studies do not follow a standard and structured procedure. Designing a PEM remains a complex task as it requires formulating various assumptions, such as determining which parameters to consider and which kind of error to model. This makes each study unique and rarely compatible with the others. However, most of these studies could be easily adapted and integrated into our proposed simulation framework by replacing the PEM in the PEM server illustrated in Figure 7.

In our previous work [6], PEMs were hand-crafted to explore specific *variations*, e.g., varying the detection probability. In this study, PEMs are learned from the nuScenes dataset [43] processed by Apollo perception module [42]. In particular, we selected three sensor setups (CAM, LID, and FUL), subsets of the sensors in the nuScenes dataset. A first step in this direction is explored in [32], where we introduced the idea of injecting perception errors in a virtual environment. However, that study was limited to 2D bounding boxes generated by a camera sensor, and the error modeling only considered temporal dependency. In the current work we abstract the sensor type, and our example PEMs address all four considerations presented in subsection IV-B. We implement temporal aspects in M and parameter inter-dependencies in p. We address positional aspects with the grid partitioning

and occlusion with C, which is not considered in any other related study in the literature.

The main limitation of the proposed approach is the need to define and model each type of error explicitly. For example, our model does not generate false positives, since errors are only computed on existing objects in the scene. Simulations based on synthetic data do not suffer from this problem, since the actual perception module introduces the errors. Nevertheless, determining if the false positives generated by the synthetic data are realistic requires modeling error properties such as its distributions and causes. Thus, modeling the errors is also needed for validation purposes.

Another limitation is that to effectively replicate environmental conditions such as rain, fog, or day/night time, these conditions need to be present in the dataset. However, this limitation affects sensor models and synthetic data in the same measure, as the sensor model requires the same kind of data to be tuned, as demonstrated by Zhao et al. [37]. Lastly, we observe that deploying PEMs instead of sensor models provides configurable properties on a different level. Sensor models allow us to tune *sensor* properties such as resolution or frequency. With PEMs, we can instead configure *perception* properties such as detection probability or tracking loss probability, as demonstrated in [6].

VII. Conclusion

In this article, we have proposed a generalized data-driven approach to test and study perception errors in a virtual environment. Our approach involves the introduction of PEM as a computationally efficient way of injecting perception errors into a simulation pipeline. We have provided guidelines for a data-driven modeling for PEM and its integration in a simulation pipeline using open-source software Apollo [42], SVL [18], and public datasets nuScenes [43]. This approach leads to benefits in many areas. First, it affords a deeper understanding of the perception module, identifying shortcomings and weaknesses. Secondly, simulation fidelity increases by not relying on perfect perception, but is not as computationally expensive as relying on synthetic signals. Additionally, we conducted simulation experiments involving three PEMs, each based on different sensor setups (camera, LiDAR, and a combination of both). Our experiments highlight the connection between S&P performance and AV safety. We identify two directions for future developments: PEM fidelity and PEM application. The former involves developing more sophisticated PEMs, considering more object parameters (e.g., size, class) and external conditions (e.g., weather). For the latter, we proposed a standardized interface notation for PEM that facilitates comparisons and reusability. These directions can promote the effectiveness of simulation platforms toward a safer deployment of AVs on public roads.

References

[1] Y. Shen et al., "To explain or not to explain: A study on the necessity of explanations for autonomous vehicles," 2020, *arXiv:2006.11684*.

[2] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for on-Road Motor Vehicles*, SAE Standard J3016_201806, 2018.

[3] E. Marti, M. A. de Miguel, F. Garcia, and J. Perez, "A review of sensor technologies for perception in automated driving," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 4, pp. 94–108, Winter 2019.

[4] *Collison Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian*, NTSB, Washington, DC, USA, 2019.

[5] W. Young, A. Sobhani, M. G. Lenné, and M. Sarvi, "Simulation of safety: A review of the state of the art in road safety simulation modelling," *Accident Anal. Prevention*, vol. 66, pp. 89–103, May 2014.

[6] A. Piazzoni, J. Cherian, M. Slavik, and J. Dauwels, "Modeling perception errors towards robust decision making in autonomous vehicles," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3494–3500.

[7] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, Feb. 2019.

[8] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," 2017, *arXiv:1708.06374*.

[9] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, Part I: Methodology," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1573–1582, Mar. 2021.

[10] A. Piazzoni, J. Cherian, M. Azhar, J. Y. Yap, J. L. W. Shung, and R. Vijay, "ViSTA: A framework for virtual scenario-based testing of autonomous vehicles," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Aug. 2021, pp. 143–150.

[11] C. Gómez-Huélamo et al., "Train here, drive there: ROS based end-to-end autonomous-driving pipeline validation in CARLA simulator using the NHTSA typology," *Multimedia Tools Appl.*, vol. 81, no. 3, pp. 4213–4240, Jan. 2022.

[12] J. Guo, U. Kurup, and M. Shah, "Is it safe to drive? An overview of factors, metrics, and datasets for driveability assessment in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3135–3151, Aug. 2020.

[13] IPG. (2022). *IPG CarMaker*. [Online]. Available: http://www.ipg-automotive.com/en/products-solutions/software/carmaker/

[14] MSC. (2022). *Vires VTD*. [Online]. Available: http://www.vires.mscsoftware.com

[15] Siemens. (2022). *Siemens Simcenter Prescan*. [Online]. Available: http://www.plm.automation.siemens.com/global/en/products/simcenter/prescan.html

[16] NVIDIA. (2022). *NVIDIA Drive Sim and Omniverse*. [Online]. Available: http://www.nvidia.com/en-sg/self-driving-cars/simulation/

[17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

[18] G. Rong et al., "LGSVL simulator: A high fidelity simulator for autonomous driving," 2020, *arXiv:2005.03778*.

[19] A. Cortés, C. Rodríguez, G. Vélez, J. Barandiarán, and M. Nieto, "Analysis of classifier training on synthetic data for cross-domain datasets," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 190–199, Jan. 2022.

[20] D. Talwar, S. Guruswamy, N. Ravipati, and M. Eirinaki, "Evaluating validity of synthetic data in perception tasks for autonomous vehicles," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Aug. 2020, pp. 73–80.

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[22] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[23] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *EURASIP J. Image Video Process.*, vol. 2008, pp. 1–10, Jan. 2008.

[24] C.-H. Cheng, G. Nährenberg, C.-H. Huang, H. Ruess, and H. Yasuoka, "Towards dependability metrics for neural networks," in *Proc. 16th ACM/IEEE Int. Conf. Formal Methods Models Syst. Design*, Oct. 2018, pp. 1–4.

[25] M. Lyssenko, C. Gladisch, C. Heinzemann, M. Woehrle, and R. Triebel, "From evaluation to verification: Towards task-oriented relevance metrics for pedestrian detection in safety-critical domains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 38–45.

[26] G. Volk, J. Gamerdinger, A. v. Bernuth, and O. Bringmann, "A comprehensive safety metric to evaluate perception in autonomous systems," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–8.

[27] C.-H. Cheng, A. Knoll, and H.-C. Liao, "Safety metrics for semantic segmentation in autonomous driving," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Aug. 2021, pp. 57–64.

[28] H. Arnelid, E. L. Zec, and N. Mohammadiha, "Recurrent conditional generative adversarial networks for autonomous driving sensor modelling," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1613–1618.

[29] V. Berkhahn, M. Kleiber, J. Langner, C. Timmermann, and S. Weber, "Traffic dynamics at intersections subject to random misperception," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 5, pp. 4501–4511, May 2022.

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

[31] R. Krajewski, M. Hoss, A. Meister, F. Thomsen, J. Bock, and L. Eckstein, "Using drones as reference sensors for neural-networks-based modeling of automotive perception errors," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 708–715.

[32] P. Mitra, A. Choudhury, V. R. Aparow, G. Kulandaivelu, and J. Dauwels, "Towards modeling of perception errors in autonomous vehicles," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 3024–3029.

[33] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[34] C. Innes and S. Ramamoorthy, "Testing rare downstream safety violations via upstream adaptive sampling of perception error models," 2022, *arXiv:2209.09674*.

[35] J. Sadeghi et al., "A step towards efficient evaluation of complex perception tasks in simulation," 2021, *arXiv:2110.02739*.

[36] E. L. Zec, N. Mohammadiha, and A. Schliep, "Statistical sensor modelling for autonomous driving using autoregressive input–output HMMs," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1331–1336.

[37] J. Zhao, Y. Li, B. Zhu, W. Deng, and B. Sun, "Method and applications of LiDAR modeling for virtual testing of intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2990–3000, May 2021.

[38] M. Hoss, M. Scholtes, and L. Eckstein, "A review of testing object-based environment perception for safe automated driving," 2021, *arXiv:2102.08460*.

[39] G. Ros, V. Koltun, F. Codevilla, and A. Lopez. (2019). *The Carla Autonomous Driving Challenge*. [Online]. Available: https://carlachallenge.org/

[40] T. Hanke, N. Hirsenkorn, B. Dehlink, A. Rauch, R. Rasshofer, and E. Biebl, "Classification of sensor errors for the statistical simulation of environmental perception in automated driving systems," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 643–648.

[41] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2012, pp. 340–353.

[42] H. Fan et al., "Baidu Apollo EM motion planner," 2018, *arXiv:1807.08048*.

[43] H. Caesar et al., "NuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11618–11628.

[44] X. Huang et al., "The ApolloScape dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 954–960.

[45] R. Kesten et al. (2019). *Level 5 Perception Dataset 2020*. [Online]. Available: https://level-5.global/level5/data/

[46] (2019). *Waymo Open Dataset: An Autonomous Driving Dataset*. [Online]. Available: https://www.waymo.com/open

[47] (2019). *CyberRT*. [Online]. Available: https://cyber-rt.readthedocs.io/

[48] J. Florbäck, L. Tornberg, and N. Mohammadiha, "Offline object matching and evaluation process for verification of autonomous driving," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 107–112.

[49] N. Cressie, *Statistics for Spatial Data*. Hoboken, NJ, USA: Wiley, 2015.

[50] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in Python using PyMC3," *PeerJ Comput. Sci.*, vol. 2, p. e55, Apr. 2016.

Andrea Piazzoni received the M.Sc. degree in computer science from the University of Milano-Bicocca, Italy, in 2016. He is currently pursuing the Ph.D. degree with IGP-ERIAN, Nanyang Technological University, Singapore. He has worked as an academic researcher on robotics and autonomous vehicles. His research interests include virtual simulation, robotic perception, decision-making, and probabilistic graphical models.

Jim Cherian received the B.Tech. degree in electronics engineering from the Cochin University of Science and Technology, India, in 2002, and the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore, in 2019, in collaboration with the BMW Group, Germany. He has worked extensively as a software architect in the automotive embedded systems software industry and a Senior Research Fellow in future mobility and autonomous vehicle enablers with CETRAN, NTU. He is currently a Principal Scientist in advanced robotic systems and applications with ARTC, A*STAR, Singapore. His research interests include optimality, interoperability, and the behavioral safety of autonomous systems for manufacturing, logistics and mobility applications, intelligent sensing and perception quality, virtual simulations, mobile computing, crowdsensing systems, and mobility data mining.

Justin Dauwels (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the Swiss Polytechnical Institute of Technology (ETH), Zürich, in December 2005. He is currently an Associate Professor in circuits and systems with the Department of Microelectronics, TU Delft. He was an Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, until 2020. His research interests include data analytics with applications to intelligent transportation systems, autonomous systems, and the analysis of human behavior and physiology. He served as the Chairperson of the IEEE CIS Chapter in Singapore from 2018 to 2020. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING since 2018, an Associate Editor for the journal *Signal Processing* (Elsevier) since 2021, and an organizer of IEEE conferences and special sessions. He has also been an elected member of the IEEE Signal Processing Theory and Methods Technical Committee and IEEE Biomedical Signal Processing Technical Committee since 2018.

Lap-Pui Chau (Fellow, IEEE) received the Ph.D. degree from The Hong Kong Polytechnic University in 1997. He was with the School of Electrical and Electronic Engineering, Nanyang Technological University, from 1997 to 2022. He is currently a Professor with the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University. His current research interests include computer vision, video analytics for intelligent transportation systems, human motion analysis, and metaverse. He was the Chair of the Technical Committee on Circuits & Systems for Communications of IEEE Circuits and Systems Society from 2010 to 2012. He was the general chair and the program chair for some international conferences. Besides, he served as an associate editor for several IEEE journals and a Distinguished Lecturer for IEEE BTS.