Procedure for Generating Weather-Specific Errors Models Toward Modeling Perception Errors in Autonomous Vehicles*

Shao-Hsuan Hung†

***Abstract*—This paper implements a procedure that generates perception error models (PEMs) under various weather conditions for the future study of autonomous vehicles (AVs) behaviors in various conditions. The machine learning-based PEM is capable of enhancing the efficiency of virtual testing for AVs without the need for computationally expensive sensor and perception models. However, the previous work [1] has two limitations: First, it does not consider the impact of various weather conditions on the PEM. Second, the proposed procedure for preparing the perception dataset is time-consuming for collecting the dataset. In this work, the implemented procedure is formed by two pipeline: (1) a perception dataset processing pipeline that allows us to gather the perception dataset w.r.t. weather conditions and sensor modalities with just one command. (2) We design a hidden Markov model (HMM)-based PEM and corresponding training pipeline to generate weather-specific PEMs. This pipeline is intended to integrate with nuScenes, a public AV dataset, and Baidu-Apollo, an open-source driving software. We then demonstrate the procedure by generating weather-specific PEMs using data from 60 scenes in the nuScenes dataset. Our results highlight that weather-specific PEMs generated by the procedure are necessity for virtual testing to simulate more realistic environmental conditions.**

***Index Terms*—Autonomous vehicles, virtual testing, hidden Markov model, sensor error modeling, perception dataset**

I. Introduction

One of the limitations that hinders the deployment of Level 4 and Level 5 autonomous vehicles (AVs) [2] to the general public is the lack of fully guaranteed safety for AVs. Specifically, sensors are known to be unreliable under certain weather conditions [3]. The sensor setup on typical AVs for environmental perception includes multiple vision cameras, radar sensors, and LiDAR sensors [4]. Each sensor modality has its own strengths and weaknesses. Table I compares various sensor modalities from both sensing and operational perspectives. Although cameras is unreliable in certain weather, visibility, and lighting conditions, they are capable of capturing color and contour information under non-extreme conditions. LiDAR sensors provide accurate distance and angle measurements, but they are susceptible to adverse weather conditions like snow and rain. This is because the laser pulses are deflected and dispersed when passing through raindrops or fog particles [5]. Radar sensors demonstrate robustness under various weather and lighting conditions. However, the current radar technology for automotive applications is constrained by its angular resolution [6]. A reliable AV should be capable of addressing the perception issues mentioned above in various conditions. A deeper understanding of how perception errors affect the AVs under various road and weather conditions is crucial.

TABLE I
Comparison of sensors for environment perception [6]

		Visible Light Camera	Radar	Lidar
Sensing features	Distance measurement	Medium	Very High	High
	Velocity measurement	Low	High	Low
	Angle measurement	Low	Medium	High
	Object features	Color contour	Intensity	Intensity
	Sampling rate	High	Medium	Low
Operational features	Weather (rain, snow, fog)	Vulnerable	Robust	Vulnerable
	Visibility (dust, smoke)	Vulnerable	Robust	Vulnerable
	Illumination (low-light, glare)	Vulnerable	Robust	Robust

To evaluate the behavior of AVs during the design of the automated driving system (ADS), various testing methods have been proposed. Some studies use high-fidelity simulation software to conduct virtual testing. Some studies have conduct in a real-world environment [7]. Virtual testing offers a safe and automated approach to validate the behavior of AVs. The typical method for simulating the operation of AVs in a virtual environment is to use physics-based models for sensors and objects. This generates synthetic signals for the ADS. However, the high physical fidelity model is computationally intensive and not yet sufficiently developed, as achieving a high level of fidelity is challenging. Therefore, an alternative, less computationally intensive way to model the functionality of the sensing and perception module (S&P) with perception errors is to incorporate a data-driven machine learning model called perception error models (PEMs) into the simulation pipeline [1]. Since the PEMs do not generate high-fidelity synthetic data, they can provide real-time output and integrate with the open-source ADS software.

In this research, our goal is to construct a procedure for future study of AVs driving behavior under various weather conditions. We use nuScenes [8], a public multimodal autonomous driving dataset, and Baidu-Apollo [9], an open-source autonomous driving platform, to validate the efficacy of

* Research internship report in Justin Dauwels's Lab, EEMCS depart., Delft University of Technology, Delft, the Netherlands. The work is under supervision of Dr. Justin Dauwels (J.H.G.Dauwels@tudelft.nl), Dr. Zhiyong Sun (z.sun@tue.nl), and Dr. Zenjie Zhang (z.zhang3@tue.nl).

† Shao Hsuan Hung is with Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, the Netherlands. Email:s.hung@student.tue.nl.

TABLE II
ACRONYMS

Abbreviation	Meaning
AV	Autonomous Vehicle
ADS	Automated Driving System
HMM	Hidden Markov Model
PEM	Perception Error Model
S&P	Sensing and Perception module
$\mathcal{W}$	Virtual world, obstacles in the test environment
$\hat{\mathcal{W}}$	Perceived world, obstacles that sense by vehicle

the entire procedure. The following contributions would like to be highlighted.

- We implement a dataset preparation pipeline that utilizes the nuScenes dataset as ground truth and collects perception error from Baidu Apollo platform.
- We propose a HMM-based PEM that models object-level perception errors. This model considers both sensor measurement errors and missing detection errors.
- We implement a training pipeline for weather-specific PEMs. The future study can use the procedure to generate weather-specific PEMs.

The article is organized as follows: In Section II, we introduce preliminary knowledge on virtual testing of AV, and provide literature reviews on sensor modeling. We then state the problem statement. In Section III, we introduce our proposed dataset preparation pipeline. In Section IV we present our designed of HMM-based PEM and the training pipeline. In Section V, we utilize designed procedure to generate the weather-specific PEMs to validate the efficacy of the entire procedure. In Section VI, we summarize the contributions and discuss future work. In Table II, we present a comprehensive list of frequently mentioned acronyms in this article.

II. Related Work and Problem Statement

In this section, we first introduce preliminary knowledge on virtual testing of AV, and HMM. Then we review the related work on sensor modeling to highlight the differences between existing work and our work. Finally, we formulate the problem statement.

A. Virtual Testing of AVs

The virtual testing environment provides a risk-free and repeatable way to test the ADS without crashing testing equipment. Moreover, many driving scenarios can be simulated simultaneously with a high level of reproducibility. Fig. 1 compares the differences of ADS in virtual testing and in the real world. The subplot (a) in Fig. 1 shows that a real-world AV is described by three components: Sensing $\mathcal{S}$, Perception $\mathcal{P}$, and Driving Policy $\mathcal{DP}$. The sensing component consists of sensors that detect obstacles in the physical world $\mathcal{W}$. The perception component is corresponds algorithms that process signals from sensors and generate the perceived world $\hat{\mathcal{W}}$. Since sensors have noise measurements and perception

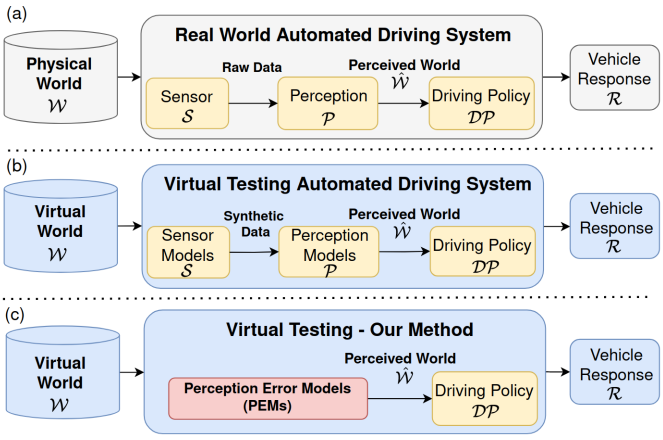

Fig. 1. Comparison between three different AV testing methods that integrate sensing, perception, and driving policy in ADS (a) testing AV in the physical world,(b) sensor models to generate synthetic data from the virtual world, and (c) our proposed approach using PEMs to perceive objects in the virtual world.

algorithms do not perfectly process signals, the signal value of the perceived world $\hat{\mathcal{W}}$ is not always the same as the physical world $\mathcal{W}$. Thus, we can view the perceived world as the result of sensing and perception components: $\hat{\mathcal{W}} = \mathcal{P}(\mathcal{S}(\mathcal{W})) = \mathcal{W} + \varepsilon$, where ε represents the error between $\hat{\mathcal{W}}$ and $\mathcal{W}$. The driving policy component $\mathcal{DP}$ is a set of processes that determine vehicle's response $\mathcal{R}$ based on the perceived world. The subplot (b) in Fig. 1 shows that the architecture of ADS in the simulation is similar to that of the real world. The only difference is that the sense data (synthetic data) is generated by the sensor models that sense the surroundings in the virtual world. For our PEM method, we define PEM as an approximation of the combination of the sensing $\mathcal{S}$ and perception $\mathcal{P}$ components:

$$\text{PEM}(\mathcal{W}) \approx \mathcal{P}(\mathcal{S}(\mathcal{W})). \tag{1}$$

Without generating computationally expensive synthetic sensor data, the PEM is capable of generating real-time data of the perceived world $\hat{\mathcal{W}}$ based on the virtual world $\mathcal{W}$, thereby improving the efficiency of the virtual testing pipeline.

B. Perception Error Modeling

Modeling sensor error enables sensor model to have realistic performance in simulation. In this work, our focus lies in the modeling of error that occurs after the perception module of the ADS. Specifically, we aim to understand and analyze the object level error that is contributed by sensor and perception module. Hoss et al. [10] provide an review on the current advancements in object-level, data-driven sensor modeling. The author highlight the absence of a standardized benchmarking dataset and sharing procedure in the sensor modeling field. To tackle this problem, Piazzoni et al. [1] propos a generalized data-driven procedure and evaluation metrics using an open-source driving software and a public AV dataset. However, the dataset preparation procedure in their work requires manually typing in the commands from ADS to extract the data. This

would be time-consuming when processing larger scale of public dataset.

For sensor error modeling, Piazzoni et al. [1] present a generalized data-driven approach to constructing perception error models (PEM) using public datasets and open-source software. They additionally implement the PEM that consist of several partitions based on a polar grid and the degree of occlusion. For each partition in the polar grid, a specific HMM with binary hidden states is trained. They consider the state of being detected or missed as the hidden state. Given that the state of detection (either detected or missed) is annotated in their data, the parameters of the PEM can be analytically obtained through maximum likelihood estimation. They then study driving scenarios using the PEM by detection frequency of object, in regardless of the measurement noise errors of sensors. Furthermore, the authors did not acknowledge the potential impact of perception errors resulting from changes in environmental conditions. These errors can vary significantly in low-light, foggy, or snowy weather [5], which contributes to the observed changes in error distribution.

In contrast to existing approaches in the literature, we develop an automatic perception dataset processing pipeline. This pipeline can be executed with a single command and is compatible with public datasets for testing on the open-source ADS platform, Apollo. Furthermore, for sensor modeling, we consider both object-level measurement error and the state of missed and detected as observations of unknown hidden Markov processes.

C. Problem Statement

In this work, our objective is to construct a procedure that generates weather-specific PEMs and validate that the procedure can be used for future study of AVs behaviors under various weather conditions. Specifically, the procedure is divided into two parts. First, for the perception dataset preparation. We implement a processing pipeline that generates the perception dataset for different conditions $\mathcal{D}$ for training PEM in terms of weather conditions and sensor setups:

$$\mathcal{D} = \{(\mathcal{W}_i, \hat{\mathcal{W}}_i)| \quad i \in C \times M\}, \tag{2}$$

where C is the set of weather conditions and M is the set of sensor setups. In our work, $C = \{\text{sun}, \text{rain}, \text{night}\}$, and $M = \{\text{fusion of LiDAR and radar}, \text{LiDAR only}, \text{radar only}\}$. The second procedure is the training pipeline of weather-specific PEM. We designed a hidden Markov models (HMM)-based PEM that models perception errors with by weather-specific perception data:

$$\hat{\mathcal{W}}_{C\times M} \approx \text{PEM}_{C\times M}(\mathcal{W}_{C\times M}). \tag{3}$$

The overall procedure is shown in Fig. 2. In Section III, we present a procedure that prepares the perception dataset. In Section IV, we formulate the HMM-based PEM and present the training pipeline.

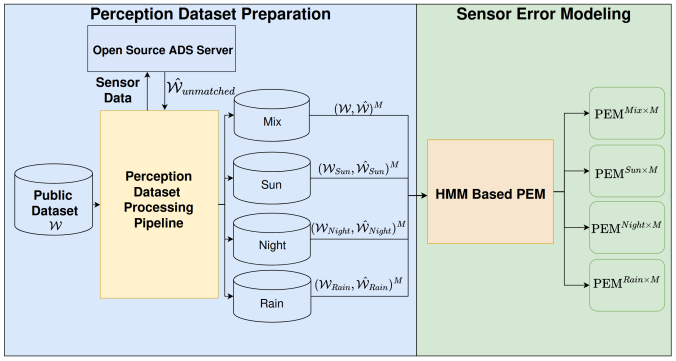

Fig. 2. The overview of the method that preparing perception dataset and modeling sensor error.

III. Perception Dataset Preparation

A. Steps of perception dataset processing pipeline

To prepare the perception dataset for training the PEM, the perception dataset processing pipeline is proposed and implemented. In general, the pipeline read and convert format of the public dataset, then send the raw sensor data streams to an open source ADS server, then retrieved the perceived world data streams from ADS to get the perception dataset. Furthermore, we provide the interface that allows users to split the perception dataset into subset in terms of weather conditions and sensor modalities. For demonstration, we select the nuScenes dataset as public dataset and Baidu-Apollo as ADS, but this pipeline is also compatible with other public dataset if ones implement the format converter. The processing pipeline is implemented by several Python scripts and Linux shell scripts. This can be fully automated and also flexible to extend to other public datasets and ADS.

The procedure of the processing pipeline is shown in Fig. 3, the pipeline consists of three steps: (1) Parse sensor information from the public dataset. (2) Send virtual world $\mathcal{W}$ and record perceived world $\hat{\mathcal{W}}$ in the ADS platform. (3) Spatial-temporary matching the objects in virtual world $\mathcal{W}$ (from public dataset) and objects in perceived world $\hat{\mathcal{W}}$ (from ADS).

For the first step, there are three information that need to be converted and sent from the nuScenes dataset to the Apollo ADS platform: (1) Sensor specifications and coordinate transformation relations among sensors. The above configurations are automatically generated and deployed to the Apollo ADS platform by the processing pipeline. (2) Raw sensor data in the nuScenes dataset. (3) Ground truth annotations of objects, including the positions and orientations of surrounding objects, which we assume to be the ground truth that consist of the virtual world $\mathcal{W}$.

For the second step, the pipeline sends the raw sensor message streams to Apollo perception scene by scene. Each scene is a 20-seconds-long record file that contains multimodal sensor data. To send and receive sensor message streams between the nuScenes dataset and sensing and the perception module of Apollo, Apollo uses Cyber-RT [11], a ROS like

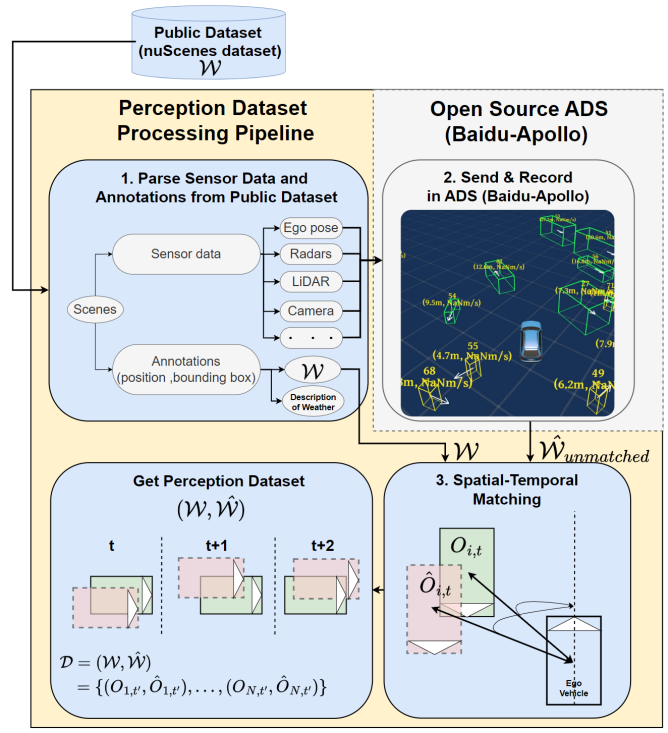

Fig. 3. Overview of procedures of the perception dataset processing pipeline.

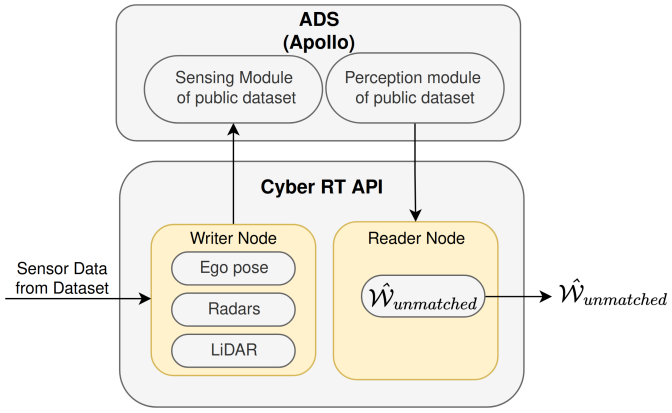

Fig. 4. Software pipeline that playing nuScenes dataset and recording perceived objects in the Baidu-Apollo ADS platform.

middleware. We use Cyber-RT to construct writer nodes that receive sensor message streams from the nuScenes dataset and transmit them to Apollo. Also, we defined a reader node to receive message streams from Apollo perception module. Specifically, as shown in Fig. 4, we build several writer nodes for ego vehicle pose, radars, and LiDAR to synchronously play raw data for nuScenes scenes. While playing the scenes, the perception module in Apollo is launched and processes the raw data, generating the perceived world $\hat{\mathcal{W}}$.

Then, in the last step, we match the perceived objects $\hat{\mathcal{W}}$ (from perception module from ADS) and annotations $\mathcal{W}$ (from public dataset) considering both spatial and temporal consistency. We adopt a well-established matching algorithm proposed by Bernardin et al. [12]. In each time frame, the algorithm attempts to find a minimum cost assignment between objects from virtual world $\mathcal{W}$ and objects from perceived world $\hat{\mathcal{W}}$ on a local pervious frame basis. One thing to highlight is that the setting of the thresholds can alter matching results. Specifically, We set up a threshold of 10 meters as the maximum distance error and a range of $-45°$ to $45°$ as the maximum angular error between a ground truth object and a perceived object to determine a match. Eventually, the perception dataset processing pipeline gives out a sequence of pairs $(\mathcal{W}, \hat{\mathcal{W}})$ for each scene in the dataset.

Since the ADS has the potential to miss detecting for tracking an object in consecutive discrete time sequences. We define two mutually exclusive sets of the missing detection time t_{miss} and detected time t' of a consecutive discrete time set t such that $t = t' \cup t_{\text{miss}} = \{t_1, 2, \cdots, t_n\}$ and $t' \cap t_{\text{miss}} = \oslash$, where t_1 and t_n is denoted as the start time frame and the end time frame of an arbitrary ground truth object trajectory. The generated perception dataset $\mathcal{D}$ is defined as a combination of pairs of sequences of the virtual world $\mathcal{W}$ and the perceived world $\hat{\mathcal{W}}$ in non-consecutive discrete time t':

$$\mathcal{D} = (\mathcal{W}_{t'}, \hat{\mathcal{W}}_{t'}) = \{(O_{1,t'}, \hat{O}_{1,t'}), ..., (O_{N,t'}, \hat{O}_{N,t'})\}, \quad (4)$$

where we denote the collection of N pairs of matched surrounding ground truth object trajectories (positional information of obstacles and road users in non-consecutive discrete time sequence) in the virtual world $\mathcal{W}'_t = \{O_{1,t'}, \cdots, O_{N,t'}\}$ and surrounding perceived objects in the perceived world $\hat{\mathcal{W}} = \{O_{1,t'}, \cdots, O_{N,t'}\}$ (generated by the S&P module of ADS). For each ground truth object trajectory O and perceived object trajectory $\hat{O}$ are described as the local polar coordinate (radial r and angular θ) of objects w.r.t. the ego vehicle in time sequences t': $O_{i,t'} = (\{r_{i,t'=t_1}, \cdots, r_{i,t'=t_n}, \{\theta_{i,t'=t_1}, \cdots, \theta_{i,t'=t_n}\})$ and $\hat{O}_{i,t'} = (\{\hat{r}_{i,t'=t_1}, \cdots, \hat{r}_{i,t'=t_n}\}, \{\hat{\theta}_{i,t'=t_1}, \cdots, \hat{\theta}_{i,t'=t_n}\})$.

B. Interface and Extensibility

We provide an interface for users to split the perception dataset $\mathcal{D} = (\mathcal{W}_{t'}, \hat{\mathcal{W}}_{t'})$ into several subsets $(\mathcal{W}_{t'}, \hat{\mathcal{W}}_{t'})^{C \times M}$ based on the weather conditions C (sun, night, and rain) and types of sensor modalities M (fusion of LiDAR and radar, radar only, and LiDAR only). For extensibility, the pipeline can be extended to extract perception data from other public dataset such as [13]. The researchers only have to implement the dataset format converter in Python. The rest of the scripts of in the pipeline can be used to extract weather-specific perception data.

IV. Hidden Markov Model-Based Perception Error Model

In this section, we provide the model structure, assumptions and training procedure of HMM based PEM. In Section IV-A, we outline the structure of the HMM-based PEM. In Section IV-B the assumptions of the HMM-based PEM are stated. Then in Section IV-C, we describe the training procedure of the model and explain how to convert the perception dataset

TABLE III
NUMBER OF PARAMETERS IN PEM

	Transition matrix	Initial matrix	Emission matrix
HMM1	n_1^2	n_1	$5n_1$
HMM2	n_2^2	n_2	$2n_2$

into observations during the data pre-processing step. Last, in Section IV-D, we demonstrate how the PEM performs inference.

A. Model Structure

The reason that we use the HMM to model the perception error is because it has been proved to have good performance on modeling the sequential data. [14]. Fig. 5 illustrates the structure of the HMM-based PEM. A PEM is consist of two sets of HMMs: HMM1 (HMM with Gaussian emissions) and HMM2 (HMM with binomial emissions). The perception errors are treated as observations in the PEM. A PEM can be described as follows:

$$\text{PEM} = \{\lambda_{\text{HMM1}}, \lambda_{\text{HMM2}}\}, \tag{5}$$

where λ_{HMM1} and λ_{HMM2} are the parameters of the HMMs, including the initial matrix $\boldsymbol{\pi}$, transition matrix $\mathbf{A}$, and emission matrix $\mathbf{B}$: $\lambda_{\text{HMM1}} = (\boldsymbol{\pi}_1, \mathbf{A}_1, \mathbf{B}_1)$, $\lambda_{\text{HMM2}} = (\boldsymbol{\pi}_2, \mathbf{A}_2, \mathbf{B}_2)$. If we have n_1 number of hidden states in HMM1, n_2 number of hidden states in HMM2, the total number of parameters for the PEM is shown in Table IV-A. Since the number of hidden states in our case is non-directly observable, we will provide more details on the criteria for selecting the number of states in the model selection step in Section IV-C.

The reason that we consider two HMMs as one PEM is because the positional perception errors consist of two types of errors: (1) sensor measurement error $\boldsymbol{\varepsilon} = (\varepsilon_r, \varepsilon_\theta)$ in local polar coordinates with respect to the vehicle. (2) the binary state variable of missed or detected $\mathbf{v}$ for sensor that observe an existing object. Consequently, for HMM1, we choose the sensor measurement errors of single object trajectory with consecutive discrete time as observations of unknown hidden state's evolution, and each hidden state have corresponding Gaussian emission probability for the observed measurement errors. For HMM2, we select the state of "missed" or "detected" in the corresponding single object trajectory in the consecutive discrete time as observations for another evolution of unknown hidden state. Each hidden state of HMM2 have corresponding binomial emission probability for the observed measurement state. The HMM2 models take a binary state variable v as output observations. The binary state variable represents the detection of each object $O_i \in \mathcal{W}$. If $v_{i,t} = 1$, means that the object O_i has been detected at time stamp t, indicating the existence of a perceived object $\hat{O}_{i,t}$. For $v_{i,t} = 0$, means that the object O_i does not have a corresponding $\hat{O}_{i,t}$.

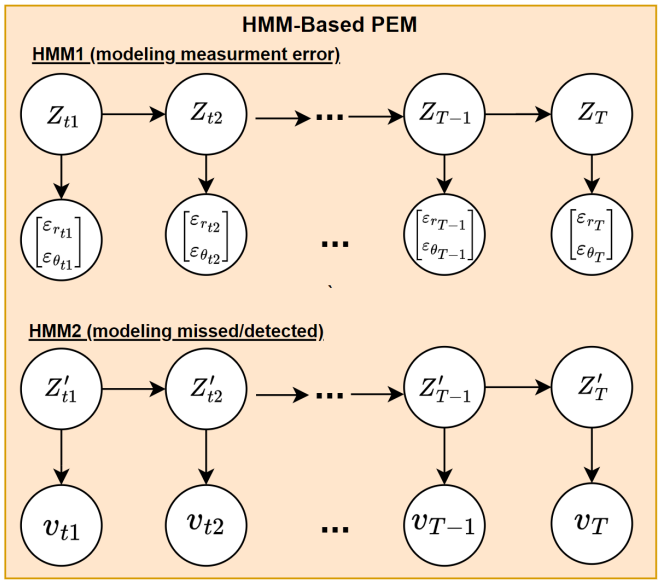

Fig. 5. The graphical reorientation of the HMM-based PEM. A PEM is consist of HMM1 and HMM2, which the HMM1 models the sensor measurement error, the HMM2 models the state of measurement is missed or detected.

B. Assumptions of HMM-based PEM

We assume that two HMMs (HMM1 and HMM2) in the HMM-based PEM satisfy the following assumptions:

- First-order hidden Markov model assumptions: (1) The fist-order Markov assumption, (2) the output independence assumption, and (3) the stationary assumption [15].
- We assume that the internal hidden states processes of HMM1 (Z) and HMM2 (Z') are independent. Also the observation of HMM1 ($\boldsymbol{\varepsilon}$) is independent to the hidden states processes of HMM2 (Z'), the observation of HMM2 ($\mathbf{V}$) is independent to the hidden states processes of HMM1 (Z). Otherwise, the PEM would be viewed as factorial HMM, which takes exponential time to train [16]. The joint probability of PEM is the product of joint probability of HMM1 and HMM2 respectively: $\mathbf{P}(\boldsymbol{\varepsilon}, Z, v, Z' | \lambda_{\text{HMM1}}, \lambda_{\text{HMM2}}) = \mathbf{P}(\boldsymbol{\varepsilon}, Z | \lambda_{\text{HMM1}}) \cdot \mathbf{P}(v, Z' | \lambda_{\text{HMM2}})$. Similarly, the maximum likelihood estimation of PEM is product of maximum likelihood estimation of HMM1 and HMM2 respectively:

$$\begin{aligned} \lambda^*_{\text{PEM}} &= \underset{\lambda_{\text{PEM}}}{\text{argmax}} \mathbf{P}((\boldsymbol{\varepsilon}, v) | \lambda_{\text{PEM}}) \\ &= \underset{\lambda_{\text{HMM1}}, \lambda_{\text{HMM2}}}{\text{argmax}} \sum_{Z} \sum_{Z'} \mathbf{P}(\boldsymbol{\varepsilon}, v, Z, Z' | \lambda_{\text{HMM1}}, \lambda_{\text{HMM2}}). \end{aligned} \tag{6}$$

This assumption guarantee the λ^*_{PEM} can be obtained by estimating the HMM1, and HMM2 respectively.

C. Training pipeline of the HMM-based PEM

Since we model perception errors as observations of an unobservable data generating process which the number of hidden states is unknown. The parameters of PEM are learned

using an unsupervised learning method [17]. We implement the model using the probabilistic programming framework, PyHHMM [18]. The procedure for PEM unsupervised learning is described below:

1) Data pre-processing: The training data for HMM1 and HMM2 consist of multiple object trajectories and binary states with varying length. We extract observations of PEM from the perception dataset that is described in Section III. However, the perception dataset only contains pairs of object trajectories for non-consecutive time sequences $(O_{i,t'}, O_{i,t'})$. Therefore, we impute the missing value with the mean error of the object trajectories, converting a non-consecutive time sequence into a consecutive time sequence. Specifically, for two mutually exclusive sets of the missed detection time set t_{miss} and detected time t' of a consecutive discrete time set t, ($t = t' \cup t_{\text{miss}} = \{t_1, \cdots, t_n\}$ and $t' \cup t_{\text{miss}} = \oslash$, where t_1 and t_n is start time frame and end time frame of an arbitrary ground truth object trajectory.) We define consecutive observations as a set of multiple sequences $\boldsymbol{\varepsilon} = \{\varepsilon_{i,t} \mid \forall i, t \in \mathbb{Z}^+\}$. Each sequence are describe as error of local polar coordinate of consecutive discrete time:$\varepsilon_{i,t} = \begin{bmatrix}\varepsilon_{r_t} & \varepsilon_{\theta_t}\end{bmatrix}^\top = (\varepsilon_{i,t'} \cap \varepsilon_{i,t_{\text{miss}}})$. For sequences in the detect time set t', they are the difference of ground truth objects and perceived objects: $\varepsilon_{i,t'} = \hat{O}_{i,t'} - O_{i,t'}$. For sequences in the miss time set t_{miss}, the missing values are imputed by the mean value of the detect time set: $\varepsilon_{i,t_{\text{miss}}} = \hat{O}_{i,t_{\text{miss}}} - O_{i,t_{\text{miss}}} = \sum_{t'}(\hat{O}_{i,t'} - O_{i,t'})/n$.

As for the training data $\mathbf{V}$ for HMM2, we define $\mathbf{V}$ as multiple sequences of binary state variables: $\mathbf{V} = \{\{v_{1,t_n}, \cdots, v_{1,t_{n+l}}\}, \{v_{k,t_{n'}}, \cdots, v_{k,t_{n'+l'}}\}\}$, where k is the indices of object trajectories, (t_n, t_{n+l}) and $(t_{n'}, t_{n'+l'})$ are the start and end frame of trajectory 1 and trajectory n respectively. We label the binary state variable $v_{i,t'} = 1$ for a detected time stamp and label the binary state variable $v_{i,t_{\text{miss}}} = 0$ for a missed time stamp. In case of a object trajectory with a high missing rate that may affect the training result, we filter out object trajectories with a missing rate higher than 50%. For training and evaluation, we divide the perception dataset into 70% of training set and 30% testing set.

2) Learning parameters of PEM and model selection: We adopt the Baum-Welch algorithm (EM algorithm) to find the maximum likelihood estimation of the PEM parameters given the sequences of observations mentioned in Section IV-C1. Since it would takes exponential time to train the PEM directly. [16], we make the assumptions mentioned in Section IV-B to estimate the PEM parameters by maximizing the likelihood of HMM1 and the likelihood of HMM2 separately. For HMM1 and HMM2, the Baum-Welch algorithm finds the local maximum likelihood:

$$\lambda^*_{\text{HMM1}} = \underset{\lambda_{\text{HMM1}}}{\text{argmax}} \mathbf{P}(\boldsymbol{\varepsilon}|\lambda_{\text{HMM1}}). \tag{7}$$

$$\lambda^*_{\text{HMM2}} = \underset{\lambda_{\text{HMM2}}}{\text{argmax}} \mathbf{P}(\mathbf{V}|\lambda_{\text{HMM2}}). \tag{8}$$

Therefore, the PEM obtain local optimum parameters $\lambda^*_{\text{PEM}} = (\lambda^*_{\text{HMM1}}, \lambda^*_{\text{HMM2}})$.

For model selection, we adopt the Akaike information criterion (AIC) [19] to determine the number of hidden states. The AIC score for HMM [20] is described as :AIC $= -2\ln L + 2k$, where L is the likelihood function of the HMM model, k is the number of estimated parameters in the HMM. The AIC attempts to balance the fit to the training data and model complexity. For higher model complexity, the AIC score tends to be higher. For the model selection, we choose the HMM with the lowest AIC score.

3) Evaluation: Quantifying the quality of the generated time series is a difficult task, and it mostly depends on the application. For the PEM, we evaluate the HMM1 and HMM2 separately. Since HMM1 is used for modeling the measurement error, we use Jensen-Shannon divergence (JS Div.) [21] to evaluated the quality of error distribution generated by HMM1 with the test dataset. One thing to notice is that we filter out samples that HMM2 predicted as missed to ensure that the evaluation only depends on samples that HMM2 classified as detected. The JS Div. is defined as:

$$J(P||Q) = \frac{1}{2}\text{KL}(P||Q) + \frac{1}{2}\text{KL}(Q||P), \tag{9}$$

where P is the distribution of the error $\boldsymbol{\varepsilon}$ that output by HMM1, and Q is the distribution of the testing set extracted from the perception dataset. $\text{KL}(\cdot)$ is the Kullback-Leibler divergence, defined as $\text{KL}(P||Q) = \sum_x P(x)\log(P(x)/Q(x))$ [22].

On the other hand, since the HMM2 models the sensor state of missed or detected, we calculate the macro average accuracy to evaluate the HMM2 with the test dataset. To fairly evaluate the accuracy rate of missed and detected, we use the macro average of accuracy. The accuracy of missed object means the ratio of state of missed predicted by PEM to that in the test set for same object trajectory. Similarly, the accuracy of detected object means the ratio of state of detected predicted by PEM to that in the test set for same object trajectory. The macro average of accuracy $\text{Acc.}_{\text{Macro}}$ is defined as: $\text{Acc.}_{\text{Macro}} = (\text{Acc.}_{\text{Missed}} + \text{Acc.}_{\text{Detected}})/2$.

D. Inference of the HMM-based PEM

After training the PEM by Baum-Welch algorithm, we use the resulting model for inference. In general, the PEM serves as a noise generator, as shown in Equation 1. Given the ground truth object, the PEM generates the perception error ε, and outputs the noisy measurements, constructing the perceived world $\hat{\mathcal{W}}$. For the HMM-based PEM, the inference can be described as:

$$\hat{O}_t = \text{PEM}(O_t) = \begin{bmatrix} v_t(r_t + \varepsilon_{r_t}) \\ v_t(\theta_t + \varepsilon_{\theta_t}) \end{bmatrix}, \tag{10}$$

where O_t are local polar coordinate (w.r.t. ego vehicle) of ground truth object trajectories in virtual world $\mathcal{W}$, the $\hat{O}_t$ are local polar coordinate (w.r.t. ego vehicle) of perceived object trajectories in perceived world $\hat{\mathcal{W}}$. When PEM predicts a detected noisy measurement, the $v_t = 1$, output would be $(r_t + \varepsilon_{r_t}, \theta_t + \varepsilon_{\theta_t})$. When the PEM predicts a missed measurement, the $v_t = 0$, making the output of the PEM becomes 0.

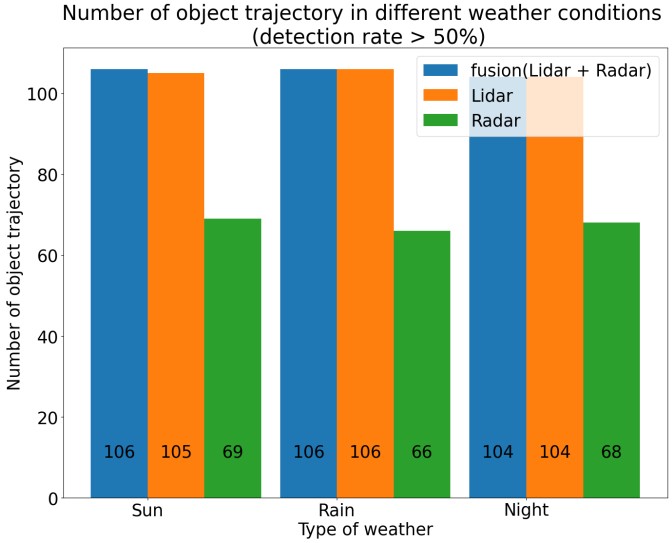

Fig. 6. Number of object trajectories in perception dataset w.r.t. weather conditions and sensor modalities that collected from 20 scenes for each weather.

V. Experiments

In goal of the experiments is to validate the implemented procedure (both dataset preparation procedure in Section III and weather-specific PEM training procedure in Section IV) can generate weather-specific PEMs that predict different distributions of perception errors. For dataset preparation phase, we collect the virtual world $\mathcal{W}$ and perceived world $\hat{\mathcal{W}}$ from nuScenes dataset and Baibu-Apollo ADS respectively. For weather-specific PEM, we use the resulted perception dataset to trained the PEM w.r.t. weather conditions (sun, night, rain) and sensor modalities (fusion of radar and LiDAR, LiDAR, and radar). The experiment study is run on laptop with an Intel i9-11980HK CPU, a NVIDIA GeForce RTX 3080 Laptop GPU, and the Ubuntu 20.04 operating system.

A. Dataset Preparation

For dataset preparation, we selected 20 scenes for each weather condition from the nuScenes dataset (60 scenes in total). Then, using the proposed processing pipeline to play each 20-seconds scene and record the perceived object from the Baidu-Apollo perception module. For the matching algorithm, we establish a threshold of 10 meters as the maximum distance error and a range of $-45°$ to $45°$ as the maximum angular error between a ground truth objects and a perceived objects to determine a match. Then, we collect trajectories with detection rate higher than 50% in case of the high missing rate affect the training result of HMM2. The resulted perception dataset, a set that consist of multiple object trajectories, is shown as Fig. 6. The reason that the radar sensors have relative smaller number of object trajectories is because the nuScenes dataset only has sparse point cloud.

TABLE IV
Evaluate JS Div. for PEM with various weather conditions and sensor modalities

	Sun	Night	Rain
Fusion (Lidar + Radar)	ε_r JS Div. : 0.158 ε_θ JS Div.: 0.156	ε_r JS Div. : 0.177 ε_θ JS Div.: 0.185	ε_r JS Div. : 0.134 ε_θ JS Div.: 0.129
Lidar only	ε_r JS Div. : 0.141 ε_θ JS Div.: 0.143	ε_r JS Div. : 0.153 ε_θ JS Div.: 0.154	ε_r JS Div. : 0.133 ε_θ JS Div.: 0.136
Radar only	ε_r JS Div. : 0.174 ε_θ JS Div.: 0.350	ε_r JS Div. : 0.257 ε_θ JS Div.: 0.240	ε_r JS Div. : 0.158 ε_θ JS Div.: 0.170

TABLE V
Evaluate macro average accuracy for HMM2 with various weather conditions and sensor modalities

	Sun	Night	Rain
Fusion (Lidar + Radar)	Macro Avg.: 0.50	Macro Avg.: 0.49	Macro Avg.: 0.52
Lidar only	Macro Avg.: 0.51	Macro Avg.: 0.51	Macro Avg.: 0.50
Radar only	Macro Avg.: 0.52	Macro Avg.: 0.54	Macro Avg.: 0.48

B. Training PEM

We use the extracted data to train the weather-specific PEM, and split the entire data into 70% training set and 30% of test set. For the evaluation step, we evaluate the HMM1 using the JS Div. The JS Div. measures the difference between the error distributions generated by PEM and the error distributions from test set. The result of PEM for different sensor setups and weather conditions are shown in Table V-B. We notice that for radar sensor, the value of JS Div. is the highest over others cases, which means the radar-based PEMs do not match the distribution of the test set very well. This is because the number of object trajectories for radar is fewer than in other cases. Then we evaluate the HMM2 by marco average of accuracy, the results are shown in Table V-B. For the inference of the PEM, HMM1 and HMM2 would generate two outputs. One is the error sampled from the Gaussian emission of the corresponding hidden state, the other is the prediction of the states that were missed or detected. An example of PEM inference is shown in Fig. 7. The red dash dot line is the sample data from the test set, the blue dot line and blue shade represent the mean and confidence intervals of the error at the corresponding hidden state.

C. Test Results

In this subsection, we validate the implemented procedure by comparing the error distributions generated by weather-specific PEMs and mixed-weather PEMs. To support the claim that the procedure can bring out different error distributions of PEMs for different weather conditions, PEMs are trained regardless of the weather type. Here, we compare the weather conditions using the same sensor setup because there is a significant difference in the number of object trajectories between radar and other sensor setups. The following analysis is based on the number of trajectories, as shown in Fig. 6. Since a PEM has two output observations: the predicted measurement error and the states of missed and detected for

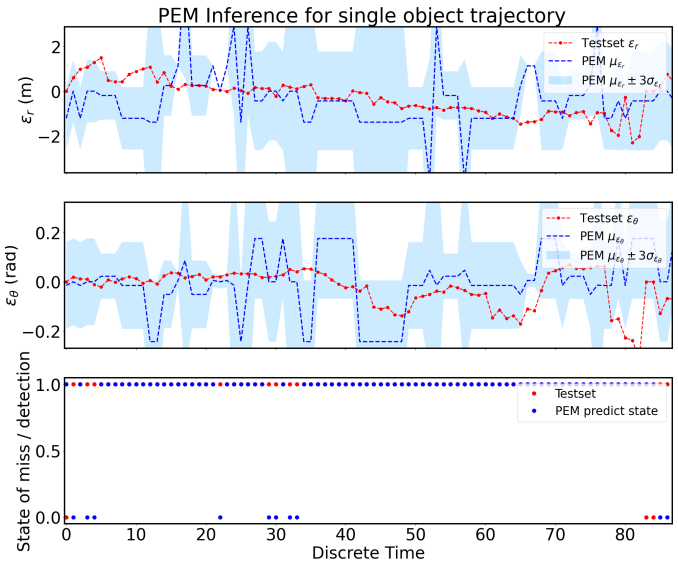

Fig. 7. Example of PEM inference compare with the single trajectory from test set.

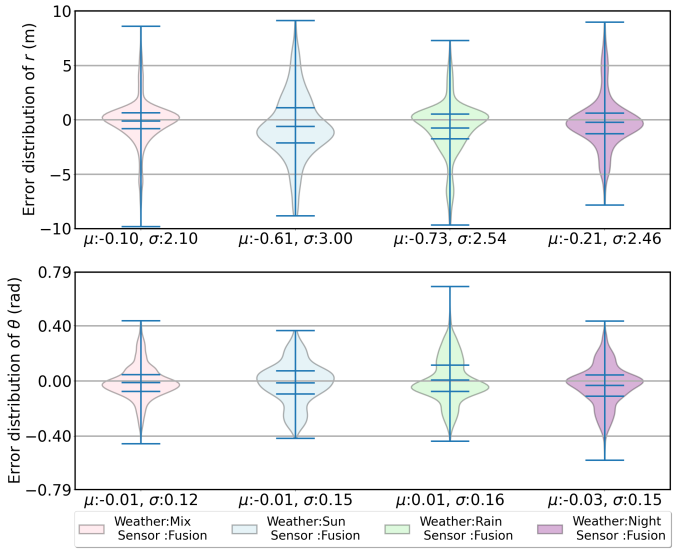

Fig. 8. Comparison of error distribution of polar coordinate r and θ for different weather, fusion of LiDAR and radar.

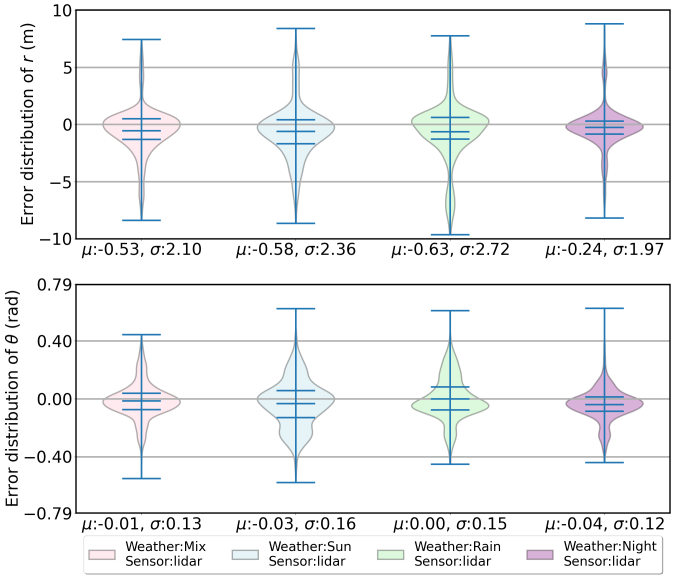

Fig. 9. Comparison of error distribution of polar coordinate r and θ for different weather, LiDAR.

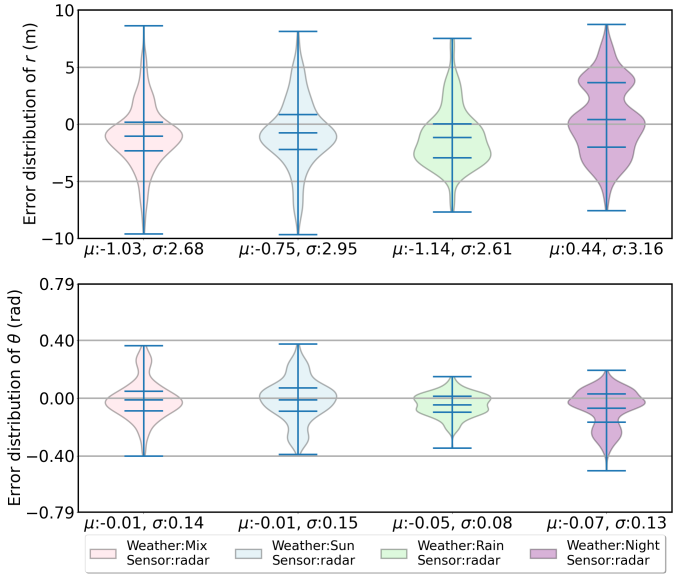

Fig. 10. Comparison of error distribution of polar coordinate r and θ for different weather, radar.

the perceived object, we plot the errors distribution that PEM classifies as 'detected' noise measurements. Fig. 8, Fig. 9, and Fig. 10 show the error distributions generated by different weather-specific PEMs with fusion, LiDAR, and radar sensor setups, respectively. We can observe that by using the implemented procedure, the weather-specific PEMs generate different error distributions w.r.t. different weather and sensor setups. The experimental results indicate that the procedure can generate weather-specific PEMs which can simulate more realistic environmental conditions.

D. Limitations

Here, we highlight two main limitations of the experiments. First, our experimental are limited to only 60 scenes (20 scenes for each weather condition) because processing all 850 scenes from the nuScenes dataset would require a massive amount of storage space. In particular, it requires 720 GB of storage space to store the record files of the 60 scenes. Secondly, the number of samples from radar sensors is much lower compared to LiDAR because the nuScenes dataset only provides sparse radar point clouds. To draw a definitive conclusion on how weather affects the PEM, it is essential to gather additional

data from the entire nuScenes dataset and obtain more samples from radar. A possible solution is to include other public datasets, such as the RADIATE dataset [13] to have more samples in diverse weather conditions (such as snow and fog) and dense radar data.

VI. Conclusion

In this work, we implement a procedure that can generate weather-specific PEMs from public dataset. To achieve this goal, we implement two pipeline: (1) an automatic perception dataset processing pipeline using the nuScenes dataset and Baidu-Apollo ADS. This pipeline has the benefit of automatically extracting the dataset for PEM training and deploying the customized sensor setup in the ADS with just one command in the terminal, which saves time. (2) We design the HMM-based PEM and training pipeline that considers both measurement errors and the state of the sensor being missed or detected as observations from non-directly observed hidden Markov processes. The procedure can generate weather-specific PEM that have different error distributions as we demonstrate in the experiments.

For future work, it is necessary to extract more data and more type of weather from multiple public datasets to train the underlying weather-specific PEMs. Our implemented procedure can generate PEM that simulates realistic road conditions once more public datasets are included. This will help people efficiently simulate the AVs' behaviors under various weather conditions in virtual testing.

References

[1] A. Piazzoni, J. Cherian, J. Dauwels, and L.-P. Chau, "Pem: Perception error model for virtual testing of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2023.

[2] SAE, "J3016: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," 2108.

[3] E. Martí, M. A. de Miguel, F. Garcia, and J. Pérez, "A review of sensor technologies for perception in automated driving," *IEEE Intelligent Transportation Systems Magazine*, vol. PP, pp. 1–1, 09 2019.

[4] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2140

[5] J. Vargas, S. Alsweiss, O. Toker, R. Razdan, and J. Santos, "An overview of autonomous vehicles sensors and their vulnerability to weather conditions," *Sensors*, vol. 21, no. 16, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5397

[6] A. Pandharipande, C.-H. Cheng, J. Dauwels, S. Z. Gurbuz, J. Ibanez-Guzman, G. Li, A. Piazzoni, P. Wang, and A. Santra, "Sensing and machine learning for automotive perception: A review," *IEEE Sensors Journal*, vol. 23, no. 11, pp. 11 097–11 115, 2023.

[7] W. Huang, K. Wang, Y. Lv, and F. Zhu, "Autonomous vehicles testing methods review," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 163–168.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[9] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo EM motion planner," *CoRR*, vol. abs/1807.08048, 2018. [Online]. Available: http://arxiv.org/abs/1807.08048

[10] M. Hoss, M. Scholtes, and L. Eckstein, "A review of testing object-based environment perception for safe automated driving," 02 2021.

[11] "Cyber-rt," https://cyber-rt.readthedocs.io/, accessed: 2023-11-13.

[12] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, 01 2008.

[13] M. Sheeny, E. D. Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, and A. Wallace, "Radiate: A radar dataset for automotive perception in bad weather," 2021.

[14] B. Mor, S. Garhwal, and A. Kumar, "A systematic review of hidden markov models and their applications," *Archives of Computational Methods in Engineering*, vol. 28, pp. 1429–1448, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:219411037

[15] A. v. d. Bosch, *Hidden Markov Models*. Boston, MA: Springer US, 2010, pp. 493–495. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_362

[16] Z. Ghahramani and M. Jordan, "Factorial hidden markov models," vol. 29, 01 1995.

[17] J. Pohle, R. Langrock, F. van Beest, and N. M. Schmidt, "Selecting the number of states in hidden markov models - pitfalls, practical challenges and pragmatic solutions," 2017.

[18] F. Moreno-Pino, E. Sükei, P. M. Olmos, and A. Artés-Rodríguez, "Pyhhmm: A python library for heterogeneous hidden markov models," 2022.

[19] H. Bozdogan, "Model selection and akaike's information criterion (aic): The general theory and its analytical extensions," *Psychometrika*, vol. 52, pp. 345–370, 02 1987.

[20] N. Dridi and M. Hadzagic, "Akaike and bayesian information criteria for hidden markov models," *IEEE Signal Processing Letters*, vol. PP, pp. 1–1, 12 2018.

[21] J. Briët and P. Harremoës, "Properties of classical and quantum jensen-shannon divergence," *Phys. Rev. A*, vol. 79, p. 052311, May 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.79.052311

[22] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," 2016.