Statistical Sensor Modelling for Autonomous Driving Using Autoregressive Input-Output HMMs

Edvin Listo Zec[1], Nasser Mohammadiha[2], Alexander Schliep[3]

Abstract— **Advanced driver assistance systems (ADAS) are standard features in many vehicles today and they have been proven to significantly increase the traffic safety. This paved way for development of autonomous driving (AD). To enable this, the vehicles are equipped with many sensors such as cameras and radars in order to scan the surrounding environment. The sensor outputs are used to implement decision and control modules. Verification of AD is a challenging task and requires collecting data from at least hundreds of millions of autonomously driven miles. We are therefore interested in virtual verification methods that simulate interesting and relevant situations, so that many scenarios can be tested in parallel. Realistic simulations require accurate sensor models, and in this paper we propose a probabilistic model based on the hidden Markov model (HMM) for modelling the sequential data produced by the sensors used in ADAS and AD. Moreover, we propose an efficient way to estimate parameters that scales well to big data sets. The results show that extending the HMM to use autoregression and input dependent transition probabilities is important in order to model the sensor characteristics and substantially improves the performance.**

I. Introduction

Advanced driver assistance systems (ADAS) and autonomous driving (AD) include a lot of different assistive technologies that increase car and road safety. Features such as emergency braking, lane keeping assist, collision warning, and blind spot detection are already being offered by many car manufactures and have been proven beneficial for safety of people inside and outside of the cars. A significant increase in safety and reduction of road injuries is foreseen when autonomous cars are brought into real traffic. Different types of sensors such as cameras, radars and lidars are used to monitor the 360 degree environment surrounding the self-driving vehicle. These sensors are used for example to detect surrounding moving objects (cars, pedestrians, cyclists), localise them and estimate their speed and heading together with stationary objects (lane markings, traffic signs, barriers), which are then used to plan safe driving to reach the destination.

Effective and efficient verification and performance analysis is crucial for successful realisation of ADAS and AD technologies. The verification of ADAS systems is usually done by collecting a lot of data from test tracks or real traffic and expeditions. The collected data is then analysed to compute statistics showing that the performance of the chosen functions fulfils the requirements. The estimated number of miles that are needed to be driven in order to ensure the safety of autonomous vehicles range from hundreds of millions of miles to hundreds of billions of miles, taking extremely long time to finish even when using large fleets of autonomous vehicles [1]. Therefore, the verification is usually based on a combination of virtual testing and testing in real traffic. Interesting scenarios with a lot of diverse realisations are simulated in parallel with virtual verification methods, resulting in significant amount of virtual-driven miles in short time. This is then combined with testing in real traffic in order to validate the safety of the end product. Realistic sensor models are very important for getting representative and useful simulations.

One class of sensor models is a statistical model that uses the ideal sensor data as input and adds noise and error to yield realistic sensor data. This could be done on raw sensor data level such as radar detections and camera images, or on a higher level such as object lists. While the first one is useful for perception, sensor fusion or whole system verification, the latter is more useful in sensor fusion and verification of functions.

Recently there have been some work on sensor modelling for the purpose of virtual testing [2], [3], [4], [5] which is mostly based on parametric methods. In [2] a non-parametric data-driven statistical model was developed from which one could draw samples in order to generate sensor position output. In [4] a radar model is proposed where first noise is added to the raw signals, and then filtering is performed to model sensor output. Further, [5] propose a variational autoencoder (VAE) approach in order to model the radar sensor output given some input vector, using object lists and spatial rasters. While these models are interesting, the time domain is often left out when modelling the sensor output. Further, a lot of domain knowledge is typically used for modelling the sensors, which makes the models very sensor dependent.

In this paper, we focus on sensor modelling on object level and propose sensor independent time series models based on a hidden Markov model (HMM) to effectively model sensor errors. We specifically consider our observations that sensor errors change slowly over time and that the errors are usually correlated with parameters describing the environment or ego-vehicle motion. To achieve a model that delivers these properties, we propose using an autoregressive input-output HMM (AIOHMM) to learn the sensor output Y_t at time t by conditioning on an input vector X_t and the previous

[1]Edvin Listo Zec is with Zenuity, Gothenburg, Sweden. edvin.listo-zec@zenuity.com

[2]Nasser Mohammadiha is with Zenuity, Gothenburg, Sweden, nasser.mohammadiha@zenuity.com

[3]Alexander Schliep is joint Faculty of the Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden. alexander@schlieplab.org

output Y_{t-1}. Large amounts of collected data from real traffic on country roads and highways in Europe is used and we perform simulations to show that the proposed methods can satisfactorily capture the sensor behaviours. Our main contributions in this paper are modelling temporal sensor characteristics with the AIOHMM. We can generate time series data describing the errors in the production sensors by sampling from the model. This synthetic data can be used in a virtual environment in order to describe realistic sensor outputs instead of using ideal sensors. Further, we enable 30 times faster training of the AIOHMM while still maintaining just as good results by using the Adam optimiser instead of line search.

II. Problem Overview and Data Set

Production sensors used in real driving may have errors expressed as noise and inaccuracies. Our main goal is to implement a generative model for these sensor characteristics in order to have an as realistic model as possible which can be used in virtual testing before the end product is released. In particular it is of interest to model the error of various sensor outputs over time of tracked objects from a host car, such as lateral and longitudinal position, velocity, acceleration and heading. The generated data can then be used in Computer Aided Engineering (CAE) tools to perform many parallel virtual tests. We use data collected in real traffics to train a generative model in order to learn the probability distribution of the sensor errors.

We consider an off-the-shelf sensor setup from Volvo Cars which is based on fusion of radar and camera sensors. The vehicle is also equipped with a Velodyne lidar HDL-64E that is used as the ground truth. The lidar detections are processed to detect objects and estimate their properties, and thus the goal is to model the difference between the production sensor and the ground truth readings for every detected object over time. For a given variable y that we want to model, the error over time t is defined as

$$y_{error}(t) = y_{sensor}(t) - y_{groundtruth}(t). \qquad (1)$$

Before the actual time series from the production sensor and the ground truth can be compared in order to calculate the error, we need to perform an association between all detected objects from both sensors. For this purpose, we have used an offline matching algorithm [6], where the detected objects are described by a dynamic state vector representing their position, speed, and the width and the length of the object. The output from the matching is a table consisting of object detections. In the table we find various different sensor outputs represented as time series, both from the ground truth and the production sensors. In total there are around 12,000 multivariate time series of different lengths in the training set, each describing a unique tracked object. 30 samples of the sensor error in estimating the the longitudinal position of an object are shown in Figure 1. Note that the vertical axis has been re-scaled due to sensitive information. For evaluation, a validation data set containing around 2,000 time series is used.

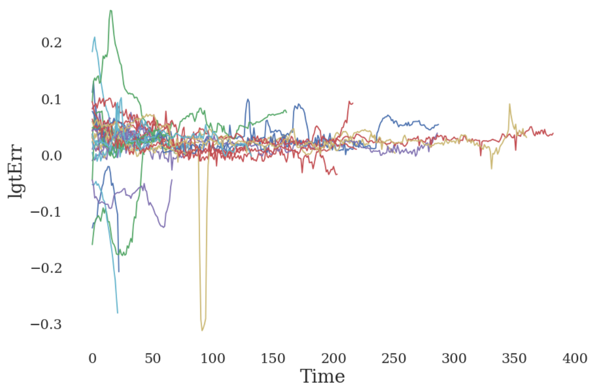


Fig. 1: 30 samples of longitudinal position error from the training data set. Each time series represent a unique tracked object. The vertical axis has been re-scaled.

III. Proposed Sensor Models

We focus on modelling the sensor error in estimating the longitudinal position of the objects error over time. The same method can be used to model other similar parameters such as error in estimating lateral position, heading and velocity of the objects, making the model sensor independent. For this purpose we propose to use hidden Markov models and extensions thereof to better capture the properties of the desired sensor error and noise signals. We start from a standard HMM and then implement and extend the autoregressive input/output hidden Markov model [7] which is inspired by the input/output HMM by [8].

HMMs have been extensively used in many different signal processing fields such as the fields of speech recognition [9], speech synthesis [10], in biology for the analysis of DNA and protein sequences [11], continuous-valued time courses [12] and inertial measurement unit (IMU) outputs [13]. The HMM can be summarised with the following parameters. The initial state probability vector $\boldsymbol{\pi}$, the transition matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and the probability density functions $\boldsymbol{f_i}$ over the observations for each state i. HMMs have proven to be very effective in modelling both discrete and continuous time series data. The standard HMM has however two main drawbacks which are described below.

Firstly, when modelling signals using an HMM it is assumed that the observations are conditionally independent given the hidden states, making it not suited for modelling highly autoregressive data. The AIOHMM extends the hidden Markov model by relaxing the conditional independence assumption between the outputs.

Secondly, the transition matrix $\boldsymbol{A}$ of the HMM is time homogeneous, making it harder to model data with long-term temporal dependencies. In order to solve this problem, the state transitions in the AIOHMM are parameterised and made time inhomogeneous with a log-linear function. This is done by conditioning the state transition probabilities on an input vector at each time step, using relevant information of the ego-vehicle and the surrounding environment that correlates with the output. Note that relaxing homogeneity of transitions has been previously explored in other domains and using other approaches [14]. Further, the output probabilities are conditioned on the same input vector.

Let X_t be the input at time step t, Z_t be the hidden state and Y_t be the output. Further, we define $S = \{1, 2, \ldots, N\}$ as the set containing all states. The AIOHMM is described as follows.

The transition probability between state i and j is parameterised with a log-linear function. We thus get a transition matrix $\boldsymbol{\psi}_{i,j}(t)$ with a time dependency modelled as

$$p(Z_t = j | Z_{t-1} = i, X_t; \boldsymbol{w_{ij}}) = \frac{\exp(\boldsymbol{w_{ij}} X_t)}{\sum_{\ell \in S} \exp(\boldsymbol{w_{i\ell}} X_t)}. \quad (2)$$

The output probabilities are parameterised with a normal distribution in each state i as

$$p(Y_t | Z_t = i, X_t, Z_{t-1}; \boldsymbol{\mu_{it}}, \boldsymbol{\Sigma_i}) \sim \mathcal{N}(Y_t | \boldsymbol{\mu_{it}}, \boldsymbol{\Sigma_i}). \quad (3)$$

In order to capture the autoregressiveness of the data and to model the temporal dependencies, the mean of the Gaussian is further modelled with a time dependency as

$$\boldsymbol{\mu_{it}} = \boldsymbol{\mu_i}(1 + \boldsymbol{a_i} X_t + \boldsymbol{b_i} Y_{t-1}), \quad (4)$$

where $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$ are learnt parameters for all states. Thus we can summarise the learning parameters of the AIOHMM as $\boldsymbol{\theta_i} = \{\boldsymbol{\mu_i}, \boldsymbol{a_i}, \boldsymbol{b_i}, \boldsymbol{\Sigma_i}, \boldsymbol{w_{ij}} | j \in S\}$.

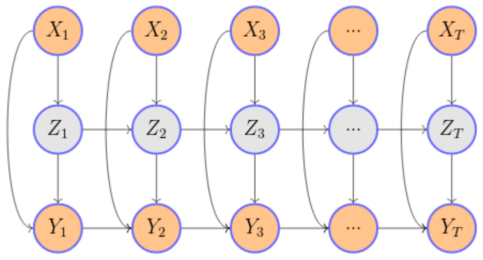


Fig. 2: A graphical representation of the AIOHMM.

We use the expectation-maximisation (EM) algorithm in order to learn the parameters in the AIOHMM, as described in [7]. The EM algorithm is an algorithm used for optimisation of the likelihood function for models that are described probabilistically in terms of observed and an unobserved components. This makes the algorithm suitable for training HMMs. The algorithm estimates a lower bound of the log-likelihood function and then finds a local optima to that estimate, ensuring monotonic increase in log-likelihood as well as convergence [15].

We implement the model in Python 3.5, and all parameters except for $\boldsymbol{w_{ij}}$ are optimised by deriving their closed form updates. In the original paper [7] the transition parameter $\boldsymbol{w_{ij}}$ is updated with line search. Although yielding good results, this slows down the training time heavily. Therefore, in this paper we propose to use the Adam algorithm [16] and implement a stochastic gradient descent based parameter estimation instead of the line search in order to learn $\boldsymbol{w_{ij}}$. This reduces the computational time needed in the M step. We parallelise the code over the states, performing gradient descent with Adam on independent cores.

A graphical representation of the model is shown in Figure 2. A time homogeneous AIOHMM (h-AIOHMM) is achieved when the link between X_t and Z_t is removed. For both the AIOHMM and the h-AIOHMM the same input features $X \in \mathbb{R}^5$ were chosen from a combination between feature importance analysis using random forests with the scikit-learn library [17] and intuition. All features come from the ground truth readings from the lidar and include for example length of the tracked object, host speed and the angle between the host car and the object. Since the models are only guaranteed to converge to a local optima, a number of different models of each type were randomly initialised and trained.

A. Model evaluation

Evaluating a generative model is not an easy task to do. Especially, quantifying the quality of the generated time series is difficult and is mostly an application dependent task. The problem is hard since usually we only have one single realisation of an underlying stochastic model, and only using this sample makes it hard to draw conclusions of the model quality. Our validation set include around 2,000 sequences, all of which correspond to unique traffic scenarios.

We will thus compare our generated samples using a validation set by using log-likelihood and root mean squared error (RMSE), which should be able to capture the temporal dependencies. The log-likelihood is one of the most used measures for evaluations of generative models and the RMSE will be used since it is suitable for time series comparison. To compare models from a different perspective, namely the closeness of the distributions between the generated samples with the real samples, one should use a metric comparing distributions. It is of interest to see if different parts of the signals, such as large errors, are represented with the right density in the generated samples. Comparing two probability distributions P and Q can be done in a lot of different ways and extensive work has been done in this field [18]. A common way to do this is using the Kullback-Leibler divergence defined as $K(P \| Q) = \sum_x \log\left(\frac{p(x)}{q(x)}\right) p(x)$ [19]. However, it is important to note that it is not a true metric as it is not symmetric, $K(P \| Q) \neq K(Q \| P)$, and as the triangle inequality does not hold. Further, the distribution P could generate samples that have zero probability for the Q distribution, which results in an infinite KL divergence. For this reason we will use the smoothed and symmetrised version of the KL divergence, namely the Jensen-Shannon divergence [20]. Let $M = \frac{1}{2}(P + Q)$. The Jensen-Shannon divergence is then defined as

$$J(P \| Q) = \frac{1}{2} K(P \| M) + \frac{1}{2} K(Q \| M). \quad (5)$$

It is symmetric and is guaranteed to be finite. Further, its square root is a true metric called the Jensen-Shannon distance (JSd).

The JSd together with the log-likelihood and RMSE will be used in order to evaluate models in a quantitative way. Note that in computing the JSd, the order of the samples is not important since a histogram is invariant under permutation. Therefore, both log-likelihood and JSd have to be taken into account and analysed in order to draw a conclusion. To further evaluate the temporal aspects of the model we will look at the JSd between the first difference distributions of the validation set and the generated samples, i.e. the distribution of $y_t - y_{t-1}$. We will further qualitatively evaluate the models by performing visual inspection of the generated data.

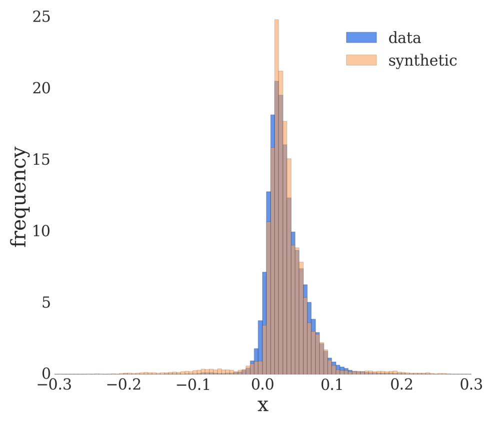
IV. Results and discussion

Three different types of models were trained and evaluated on the training and validation set. We compare standard HMMs with Gaussian mixture emissions (GMM-HMM) to AIOHMMs with and without a homogeneous transition matrix for modelling the longitudinal position error.

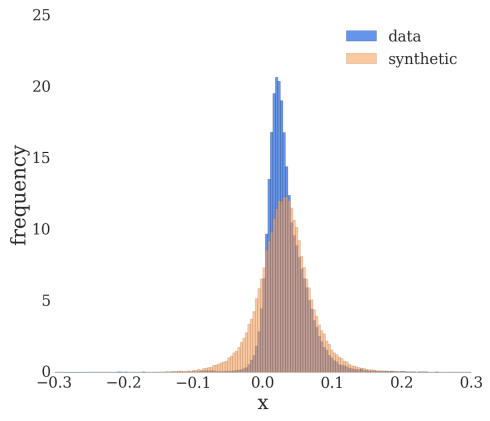
There is no simple way of optimally choosing the number of states in an HMM. Using the traditional model selection methods, as the Akaike information criterion or the Bayesian information criterion, has drawbacks when choosing the number of states for an HMM, since they often prefer too complex models [21]. In this paper the number of states were chosen by training on a different, smaller training set and evaluating the models with the log-likelihood and JSd as well as visual inspection. By empirical analysis, we in general observe that 3-5 states usually are sufficient for this particular problem, capturing the characteristics of the data while at the same time keeping the model complexity low. All models used in this paper are based on four states.

In Figures 4 and 5 we see the results from all the models. The black trajectories were chosen from the validation set to illustrate different behaviour seen in the data. The same input features X were used from these trajectories to generate 1,000 sample trajectories from each model for each validation trajectory. The mean of these samples are plotted in dashed purple. The blue area spans the interval between the 0.05th and 0.95th quantile of the simulated trajectories. Note that the vertical axis is re-scaled.

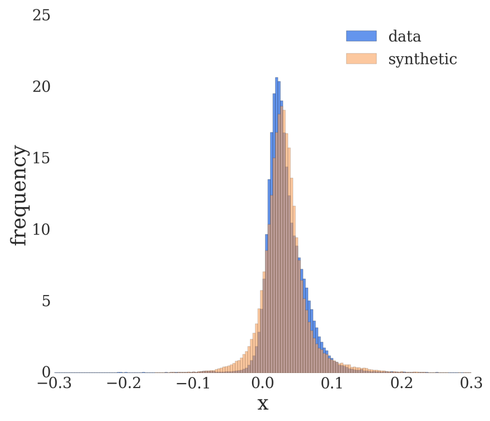
Figures 4a and 5a show sample trajectories from the GMM-HMM and Figure 3a shows that it is able to capture the underlying distribution as a whole, with a Jensen-Shannon distance of 0.14 as shown in Table I. However, a low log-likelihood and visual inspection shows that the GMM-HMM does not capture the temporal dependencies. For example, it does not capture the autoregressiveness of the data due to the independence assumptions between outputs. This is also reflected in Table I which shows that we have a relative high JSd of 0.27 for the first difference of the sample trajectories generated by the GMM-HMM as compared to the other models.



(a) GMM-HMM histogram

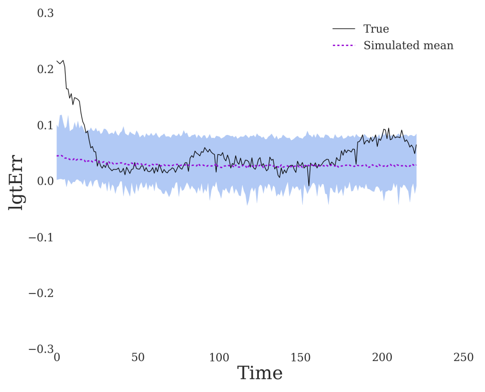


(b) h-AIOHMM histogram

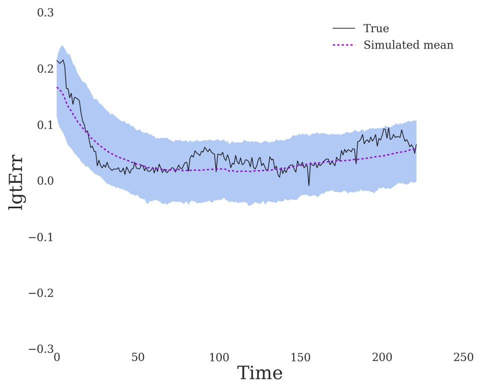


(c) AIOHMM histogram

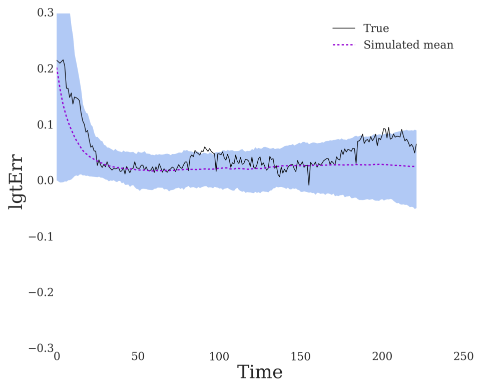
Fig. 3: Histogram of observations in all sequences from the validation set and samples from the models. Orange is generated data and blue is validation data. The horizontal axis has been re-scaled.



(a) GMM-HMM

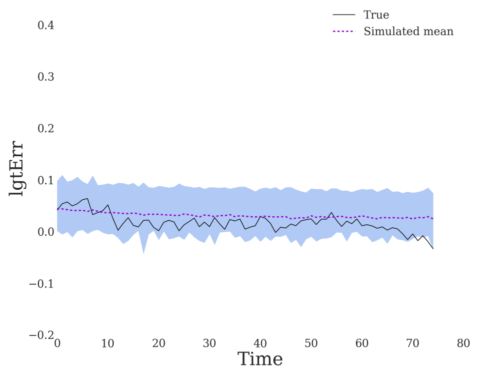


(b) h-AIOHMM

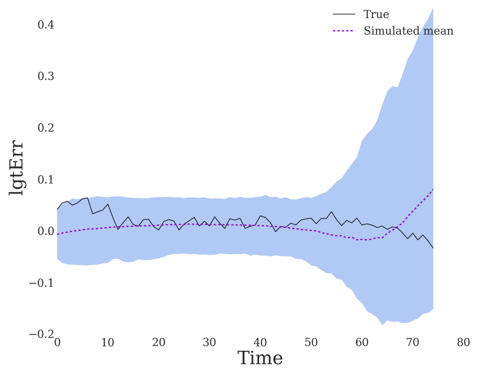


(c) AIOHMM

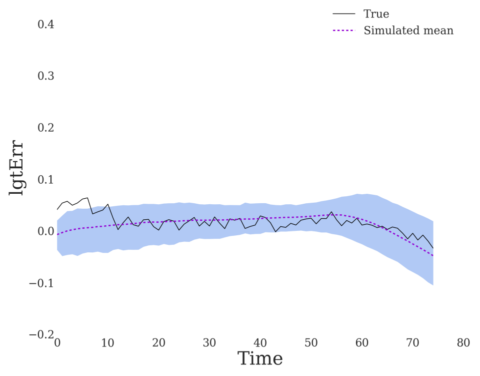
Fig. 4: Trajectory 1 of the longitudinal position error from the validation set (black) together with sampled trajectories from the three different models. The vertical axis has been re-scaled.



(a) GMM-HMM



(b) h-AIOHMM



(c) AIOHMM

Fig. 5: Trajectory 2 of the longitudinal position error from the validation set (black) together with sampled trajectories from the three different models. The vertical axis has been re-scaled.

TABLE I: A summary of all the models.

Model	JSd	1st diff. JSd	Loglik. (10^4)
GMM-HMM	0.14	0.27	5.8
h-AIOHMM	0.24	0.11	11.3
AIOHMM	0.13	0.15	10.3

TABLE II: Root mean squared error for the different models.

Model	Trajectory 1	Trajectory 2	Avg. of all trajectories
GMM-HMM	0.87	0.68	0.86
h-AIOHMM	0.53	0.93	0.71
AIOHMM	0.65	0.42	0.67

In Figures 4b and 5b we see that the temporal dependencies are captured more smoothly for the h-AIOHMM relative the GMM-HMM. Meanwhile, compared to the AIOHMM in Figures 4c and 5c it is noted that the uncertainty is higher for the h-AIOHMM. Figure 3b further shows that the overall distribution is not captured as good as for the AIOHMM, shown in Figure 3c. This suggests that conditioning the output on the previous output together with inputs, although very important, is not enough to capture the overall nature of the data. We notice that conditioning the transition probabilities on the inputs and enabling the model to have time inhomogeneous transitions is important in order to capture the temporal dependencies. Table I shows that both the AIOHMM and the h-AIOHMM manage to capture the distribution of the first difference quite well and both models have high log-likelihood relative the GMM-HMM.

Further, from the 1,000 sample trajectories root mean square errors between the real trajectory and the sample trajectories were calculated. In Table II the mean of these 1,000 RMSEs for the two chosen trajectories are shown, and in the fourth column we report the RMSE values where 100 sample trajectories were drawn for each one of the 2,000 real trajectories. We observe that the AIOHMM outperforms the other two models.

V. Conclusions

In this paper we developed statistical sensor models for autonomous driving, which are able to generate time series describing sensor outputs such as longitudinal and lateral position, velocity and acceleration. These models are sensor independent and can be used in automotive simulation environments in order to efficiently and effectively model realistic sensor outputs. The proposed data-driven models are evaluated and compared both qualitatively and quantitatively to collected data from real test drives. Our evaluations show that the AIOHMM is the best model, able to capture the temporal aspects of the sensor characteristics better than the other two models. The AIOHMM is through autoregression and the time inhomogeneous transition probabilities able to capture the temporal structures of the sequences and the dependency of the error to other parameters describing the scenario. Future work will include implementing these models in virtual environments, where the models can be tested on system level. This will enable testing of ADAS and AD functions and we will be able to evaluate the models further from the results of these tests.

Acknowledgement

The authors would like to thank Volvo Car Corporation for providing the data necessary to perform this work.

References

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, no. C, pp. 182–193, 2016. [Online]. Available: https://EconPapers.repec.org/RePEc:eee:transa:v:94:y:2016:i:c:p:182-193

[2] N. Hirsenkorn, T. Hanke, A. Rauch, B. Dehlink, R. Rasshofer, and E. Biebl, "Virtual sensor models for real-time applications," *Advances in Radio Science*, vol. 14, no. B., pp. 31–37, 2016.

[3] N. Hirsenkorn, H. Kolsi, M. Selmi, A. Schaermann, T. Hanke, A. Rauch, R. Rasshofer, and E. Biebl, "Learning sensor models for virtual test and development."

[4] S. Bernsteiner, Z. Magosi, D. Lindvai-Soos, and A. Eichberger, "Radar sensor model for the virtual development process," *ATZelektronik worldwide*, vol. 10, no. 2, pp. 46–52, 2015.

[5] T. A. Wheeler, M. Holder, H. Winner, and M. J. Kochenderfer, "Deep stochastic radar models," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 47–53.

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

[7] A. Jain, H. S. Koppula, B. Raghavan, and A. Saxena, "Know before you do: Anticipating maneuvers via learning temporal driving models," *CoRR*, vol. abs/1504.02789, 2015. [Online]. Available: http://arxiv.org/abs/1504.02789

[8] Y. Bengio and P. Frasconi, "Input-Output HMM's for Sequence Processing," *Annalen der Physik*, vol. 7, no. 5, pp. 1231–1249, 1996. [Online]. Available: http://www.dsi.unifi.it/ paolo/ps/tnn-96-IOHMMs.pdf

[9] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Foundations and trends in signal processing*, vol. 1, no. 3, pp. 195–304, 2008.

[10] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 1315–1318.

[11] B.-J. Yoon, "Hidden markov models and their applications in biological sequence analysis," *Current genomics*, vol. 10, no. 6, pp. 402–415, 2009.

[12] A. Schliep, A. Schönhuth, and C. Steinhoff, "Using hidden markov models to analyze gene expression time course data," *Bioinformatics*, vol. 19, no. suppl_1, pp. i255–i263, 2003.

[13] G. Panahandeh, N. Mohammadiha, A. Leijon, and P. Händel, "Continuous hidden markov model for pedestrian activity classification and gait analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 5, pp. 1073–1083, 2013.

[14] M. Seifert, M. Strickert, A. Schliep, and I. Grosse, "Exploiting prior knowledge and gene distances in the analysis of tumor expression profiles with extended hidden markov models," *Bioinformatics*, vol. 27, no. 12, pp. 1645–1652, 2011.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: http://www.jstor.org/stable/2984875

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City*, vol. 1, no. 2, p. 1, 2007.

[19] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," *arXiv preprint arXiv:1511.01844*, 2015.

[20] J. Briët and P. Harremoës, "Properties of classical and quantum jensen-shannon divergence," *Physical review A*, vol. 79, no. 5, p. 052311, 2009.

[21] 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," *arXiv preprint arXiv:1701.08673*, 2017.