2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Continuous Behavioral Prediction in Lane-Change for Autonomous Driving Cars in Dynamic Environments

Chiyu Dong[1] and John M. Dolan[2]

***Abstract*— It is essential for autonomous driving cars to understand and predict other surrounding cars' behaviors, especially in urban environments, due to the high traffic volumes and complex interactions. Modeling the interaction among cars and their behaviors is challenging. The behavior estimation of a surrounding car serves as prior knowledge which helps the trajectory planner generate a path to perform properly with the other vehicles. It closes the gap between the high-level decision making and path planning. A new data-driven method is proposed to extend our previous behavior estimation. The new method predicts the continuous lane-change trajectory of a target car by modeling the interaction of all its surrounding vehicles' trajectories, without over-the-air communication between vehicles. The advantages of this approach are: 1. Learning the interactive model from real data; 2. Giving long-horizon estimation of the continuous trajectory of a target vehicle. The method is trained and evaluated on a public dataset. The experimental results show that the proposed method successfully predicts trajectories considering mutual interactions among cars, with low error based on the ground-truth.**

I. INTRODUCTION

As the autonomous driving industry continues to grow, more cars equipped with ADAS are running on public roads. Major car makers have already released ADAS features for their commercial vehicles. Those cars can perform Level 3 autonomy according to NHTSA's "Levels of automation" [1]. Google and Uber have been testing their autonomous cars in urban neighborhoods of cities such as Mountain View, Pittsburgh and Phoenix. However, complex scenarios such as intersections, ramp-merging and lane changes, which involve cooperation, intention understanding and social behaviors among traffic participants, still present challenges.

In addition to robust perception and control in the autonomous car, these scenarios require the car to interact suitably and socially appropriately with surrounding vehicles. There are two aspects of social behavior: 1) Correctly understanding human drivers' behaviors or intentions; 2) Reacting properly, similarly to humans.

Fig. 1 depicts a lane-change scenario, where the lane-changing trajectory of the target vehicle (Veh-s) highly relies on the surrounding vehicles' behaviors and recent movements. Therefore, in order to accurately predict the lane-change trajectory of one target vehicle, all surrounding trajectories should be taken into account. Once knowing other vehicles' intentions or estimated trajectories, current trajectory planners can generate proper reactions and paths to cooperative with surrounding cars. For example, in Fig.1, Veh-st is the autonomous driving vehicle, and Veh-s is a surrounding vehicle with its left-turn signal on, indicating the intent to perform a left lane change. The host vehicle (Veh-st) should estimate when and how the target car (Veh-s) will make the lane change, and then plan a proper trajectory to react. If there is a predictive engine which gives surrounding vehicles' future movement (the red dashed lines), trajectory planners can generate the corresponding path to perform safely. The difficulties of the trajectory estimation are: 1) modeling the mutual interactions among surrounding cars; and 2) predicting the continuous trajectories based on the interactions.

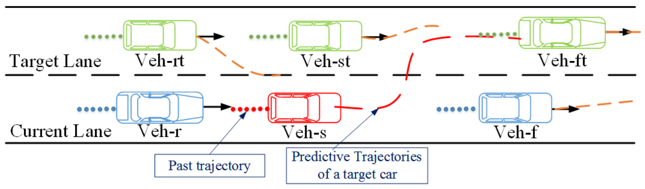


Fig. 1: A left lane change scenario. The red car (Veh-s) is the target car, which may perform a left-lane change into the target lane. Veh-st is an autonomous driving car, which needs to predict the lane-change trajectory of Veh-s in order to achieve cooperative behavior. The prediction considers the target car's leading car (Veh-f) and the following car (Veh-r); as well as the vehicles in the target lane: the immediate left car (Veh-st), its leading car (Veh-ft) and its following car (Veh-rt). The red dashed line indicates the predicted path for the target car (Veh-s).

Our previous method [2] considers the interaction of surrounding vehicles as a non-parametric regression and just outputs discrete lane-changing start/end points. Based on the previous method, the current paper extends the model to take continuous trajectories as input and then output a continuous trajectory as the prediction. The predicted trajectory of a target car is the result of the mutual interaction among its surrounding vehicles. The analysis of the interaction and the future movement is based on a short period of observations of all surrounding cars' movements. The method works as a regression that projects the current observations to produce the estimations. No assumption is made about how these cars interact with each other. The interactions among cars are captured inside the regression, which is learned from training data.

II. Related Work

In the last decade, researchers have proposed numerous cooperative planning algorithms for autonomous driving. Those

[1]Chiyu Dong is with the Department of Electrical and Computer Engineering Carnegie Mellon University, Pittsburgh, PA 15213 USA. chiyud@andrew.cmu.edu
[2]John M. Dolan is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA. jmd@cs.cmu.edu

algorithms can be categorized into three major categories of methods to address the social cooperation problem among cars:

A **Rule/Control-based** methods, represented by earlier slot-based lane-change decision making. In addition, control-based algorithms are also considered.
B **Optimization-based** approaches, which optimize specific cost functions to guarantee proper behaviors.
C **Probabilistic** approaches, most of which use the Markov Decision Process (MDP) and its extensions.

A. Rule-based methods

The rule-based methods are the most understandable and straightforward approaches. They have been applied on test vehicles since the 2007 DARPA Urban Challenge. For example, a slot-based approach was implemented for the CMU Boss vehicle merge behavior planner [3] . In the planner, kinematic information is used to check merge-in feasibility of each slot. Then the target slot is selected from the set of feasible slots according to the context of the maneuver, and predictions of others. The slot-based approach is straightforward to implement and robust in obvious or simple scenarios. However, the lack of prior knowledge of surrounding vehicles' intentions makes it hard to estimate or predict their movements and corresponding behaviors. Naranjo et al. [4] use fuzzy logic to make lane-change decisions. The method is also straightforward and simple to implement. However, it also does not consider prior knowledge, reaction of other cars, or prediction. Lu and Hedrick [5] formulated the cooperative behavior as a platoon control problem. But the algorithm still requires a "coordination layer" to select a pair of main-road cars to interact.

B. Optimization-based methods

Liu et al. [6] and Nilsson et al. [7] applied Model Predictive Control(MPC) to solve the cooperative planning problem. In order to improve computational efficiency, the MPC is converted to a convex optimization over its manifold. The method is theoretically sound for performing cooperative trajectory planning for robots. However, it still needs a predictive engine to provide an initial estimate of the other agents' possible future trajectories and the associated uncertainty. The Intelligent Driver Model (IDM) [8] based algorithms optimized the ego-vehicle's reactions in interaction-required scenarios such as freeway entrances and turns in intersections. The algorithms assumed that all other vehicles applied the identical behavioral model (IDM). Given a proposed intention for the ego-vehicle, the optimization algorithms converge to a proper cooperative trajectory regarding the reactions from the IDM-driven agents. This approach has several problems. Firstly, the assumption of IDM does not necessarily fit all vehicles, especially in interactive scenarios. Secondly, the IDM relies on several parameters which require hand-tuning. Thirdly, IDM is one of the most robust distance-keeping models, but interactive scenarios involve various interactions other than car-following, such as yield or not-yield reactions.

C. Probabilistic methods

Probabilistic methods form the largest percentage of solutions to lane changing or cooperative driving. Montemerlo et al. [9] integrated lane-changing behavior into Stanford Junior's global path planner, which is an instance of dynamic programming (DP). In fact, the problem is formulated as optimizing a variant Bellman equation, which implicitly follows the MDP framework and value iteration. Each action is assigned a penalty cost. The lane changing behavior is a penalty term in the cumulative cost function which is optimized by the DP. However, the algorithm does not consider other traffic participants. Galceran et al. [10] make decisions depending on the probability of past trajectories of all traffic participants. Both of them report discrete actions such as left-lane-change, right-lane-change etc., which can be used as an upper-level module in our method. Dong et al. [11] detect whether the other car will merge in by using PGM. However, this method only provides binary output of whether a car is likely to yield or not yield.

Ulbrich et al. [12] and Wei et al. [13] proposed an online Partial Obervable Markov Decision Process (POMDP) for lane-change using real-time belief space search [14]. However, to achieve real-time performance and use a simple POMDP framework, they discretized state and action spaces. To avoid discrete states, Bai et al. [15] proposed a continuous-state POMDP using a belief tree, and the model was applied to navigating intersections. However its actions are discrete and represented by a generalized policy graph (GPG). Seir et al. [16] proposed an online and approximate solver for a continuous-action POMDP, but only tested on toy problems. The POMDP solutions above still need manually designed probabilistic transition models and reward functions. Kuefler et al. [17] used Generative Adversarial Networks to mainly imitate and estimate single-lane behaviors. Qiao et al. [18] used reinforcement learning to model the interaction behaviors among vehicles and decision making for autonomous driving cars in intersections. Sadigh et al. [19] establish the transition models by (inverse) reinforcement learning, but their solutions are limited to the specific scenario. Chen et al. [20] solve the sequential prediction of the trajectory by using Long Short-Term Memory (LSTM).

Our method serves as a predictive engine which provides trajectory estimations. The estimated trajectory then helps the motion planner to make a desired path in dynamic environments. The trajectory estimation is based on an interactive model that captures mutual influences among all surrounding cars. The model is optimized from real training data.

III. Method

A. Model the trajectory prediction as a function

In our proposed method, the interactive trajectory predictor is formulated as a function of the related surrounding cars. In Fig. 1, the trajectories of all related surrounding cars and the target car are taken as the input. Considering Veh-s is the target car, whose trajectory needs to be estimated, the related surrounding cars of the target car include the leading car (Veh-f) and the following car (Veh-r) in the current lane, and the immediate neighboring car (Veh-st) in the target lane and its leading (Veh-ft) and following cars (Veh-rt). The method outputs the predicted lane-change trajectory, which is represented by a continuous expression of the trajectory. Intuitively, the predictor acts as a mapping from past trajectory to the future trajectory. More precisely, F: X $\rightarrow \mathcal{B}$ maps a collection of surrounding trajectories ($\mathbf{X} \in \mathcal{X}$) to a future trajectory ($b \in \mathcal{B}$). ($\mathcal{X}$ and $\mathcal{B}$ are respectively the input and output trajectory spaces, and $\mathbf{X}, b$ are corresponding elements in the two spaces. Details will be discussed in the next paragraph.) However, the model of the regression is hard to define; moreover, the input and output are of high dimension or even continuous. Mathematical tools in the Reproducing Kernel Hilbert Space (RKHS) provide feasible and theoretically sound ways to construct the regression without explicitly designing the model. The regression is considered as a mapping in RKHS and the following sections introduce a way of constructing the regression from data.

B. Construct the trajectory predictor model in RKHS.

Marinho et al. [21] introduced the reproducing Kernel Hilbert Space (RKHS) representation for trajectory planning. In this work, Gaussian Radial Basis Functions (RBF) are used to represent a trajectory, followed by a functional gradient to optimize a cost function to traverse through static obstacles or manipulate a high-dimensional arm. The idea has the potential to solve prediction and planning problems in dynamic and cooperative scenarios. We follow the framework to extend our previous work from estimating discrete values to the prediction of the continuous trajectory. Instead of using functional gradient to find the optimal solution, our method is based on RKHS non-parametric regression and a prior dataset. The result is a continuous representation of the predicted trajectory. In addition, our method has a closed-form solution, without using gradient descent.

A function f in a Reproducing Kernel Hilbert Space $\mathbf{H}$ can be represented by a linear combination of the kernel: $f(\cdot) = \sum \alpha_i K_{x_i}(\cdot)$, and this kernel has the reproducing property: $f(x) =< f, K_x >$, which is essential to RKHS. The kernel makes it possible to evaluate the function without an explicit definition of the function in the high-dimensional functional space [22]. Recalling the discussion in Section III-A, a trajectory predictor is then a function $\mathbf{F} : \mathcal{X} \rightarrow \mathcal{B}$, which maps a vector of trajectories ($\mathbf{X} \in \mathcal{X}$) to a future trajectory ($b \in \mathcal{B}$). $\mathcal{X}$ is a coordinate space which contains vectors of N surrounding vehicles' past trajectories $\mathbf{X} \overset{\text{def}}{=} \{x_i\}_1^N$, $x_i \in \mathbb{R}^{\mathrm{T}}$. T is the length of the relevant historical poses. The input contributes to all elements in the output vector. Using $\mathbf{\Gamma} = \{X\}_i^N$ as the training set and X_i as a training sample, then $\mathbf{F(\Gamma)} = \{[f_1(X_1), ..., f_D(X_1)], ..., [f_1(X_N), ..., f_D(X_N)]\}$. The output range $\mathcal{B} \subseteq \mathbb{R}^{\mathrm{D}}$ contains the representation of the predicted trajectory. The dimension of the range (the output domain) is $D > 1$, and this function is a vector-valued function. The kernel which is mentioned in the paragraph above is no longer a scalar-valued function but a matrix-valued one, i.e., $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\mathrm{D \times D}}$,

$$K(X,U) = \begin{bmatrix} k(X,U)_{1,1} & \cdots & k(X,U)_{1,D} \\ k(X,U)_{2,1} & \cdots & k(X,U)_{2,D} \\ \vdots & \cdots & \vdots \\ k(X,U)_{D,1} & \cdots & k(X,U)_{D,D} \end{bmatrix} \tag{1}$$

The Representer Theorem [23], [24] preserves the reproducing property for the matrix-valued kernel,

$$F(X) = \sum_{i=1}^{N} K(X_i, X) \cdot \boldsymbol{\alpha}_j, \quad \boldsymbol{\alpha}_j \in \mathbb{R}^{\mathrm{D}} \tag{2}$$

The $\cdot$ operator is the normal inner product in Euclidean Space, and $\boldsymbol{\alpha}$ is a ND-dimensional coefficient. Note that because the prediction function $\mathbf{F}$ is constructed into the RKHS, $\mathbf{F}$ is continuous and can be represented by a linear combination of a set of basis functions. The functions are unknown to us, and the exact form of the behavior generator function is also less interesting; instead, its evaluation given input trajectories is more important. Since we do not explicitly know the form of the function and in order to approximate the evaluation, we use non-parametric regression from the data in the RKHS, which is defined by the kernel above.

C. Non-parametric regression for trajectory prediction in RKHS

In order to construct the function for realistic prediction, it should be estimated from data and properly evaluated for given input. Note that we do not explicitly define the form of the function; instead, we use a linear combination of kernels, as mentioned in Equation 2. Once the kernel is determined, the only parameter to be optimized is the coefficient $\boldsymbol{\alpha}$. Therefore, the approximation becomes minimizing the regularized empirical error:

$$\hat{f} = \underset{f \in H}{\arg\min} \sum_{i=1}^{N} (b_i - f(X_i))^2 + \lambda J(f) \tag{3}$$

where (X_i, b_i) is training input and desired output, and $J(f)$ is the penalty term to prevent over-fitting. The regularization factor λ leverages the smoothness and accuracy of the regression function. Here $||f||_H$ is used as the penalty term (or the regulation term). According to [23], [24], there is a closed-form solution to estimate the coefficients:

$$\boldsymbol{\alpha} = (K(\mathbf{X}, \mathbf{X}) + \lambda N \mathbf{I})^{-1} \mathbf{b} \tag{4}$$

Substituting the evaluation from Equation 4 into Equation 2 yields the prediction function $\hat{f}$. With a new input X' observed, the prediction $\hat{b}$ becomes

$$\hat{b} = K^*(K + \lambda I)^{-1} \mathbf{b} \tag{5}$$

where $\mathbf{b} \overset{\text{def}}{=} \{b_i\}_1^N$ is the collection of the training trajectory samples. K^* is the new kernel result given incoming input. The $(K+\lambda I)^{-1}\mathbf{b}$ part can be pre-calculated offline given the training samples. Only the K^* will be re-evaluated with new observations, and matrix multiplication is performed with the pre-calculated $(K + \lambda I)^{-1}\mathbf{b}$.

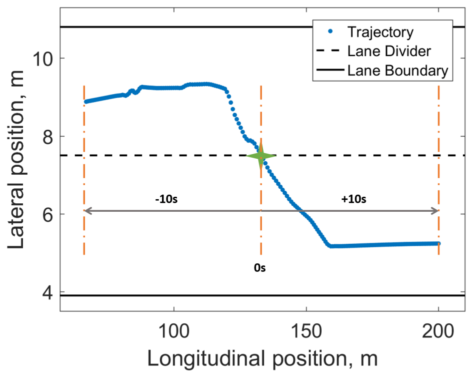


Fig. 2: A Trajectory of a lane-change target car in the dataset. The green starred point is the cross-lane point of the trajectory, and considered as the zero moment. The trajectory segments which are 10 seconds before and after the zero moment will be recorded as the full lane-change trajectory of the target car.

D. *Represent trajectories*

B-splines are widely used in computer graphics to represent continuous curves or planes. In the RKHS context, the B-spline representation can also be considered as a finite kernel [25]. Unlike other infinite kernels, the feature map of the B-spline kernel is easier to obtain, i.e., the parameters of a B-spline, including its knots and coefficients. In addition, a B-spline representation also preserves C2 (curvature) continuity regardless of the interpolation of noisy raw trajectory data points. The B-spline results in a smooth and controllable trajectory. The smoothness and feasibility of a trajectory is essential for the autonomous driving car. Therefore, the elements of the gram matrix in Equation (1) can then be calculated from the inner product of the feature map in the RKHS,

$$k(X,U)_{i,j} =< \phi(X),\ \psi(U) >_H \tag{6}$$

where $\phi(\cdot)$ $\psi(\cdot)$ are the feature map of the B-spline representation.

IV. Experimental Results

Real trajectory data from NGSIM[26] (US-101 and I-80 subsets) are used for training and testing in the experiments. Lane-change scenarios are extracted from the real data, and organized into groups. As shown in Fig. 1, each group contains one host car (Veh-s) and five surrounding cars, i.e., Veh-f, Veh-r, Veh-rt, Veh-ft, Veh-st. As shown in Fig. 2, the trajectory of each car in the group is recorded from 10 seconds before to 10 seconds after the host car (Veh-s) crossing the lane-marking.

TABLE I shows the total number of trajectories for each car type in the lane-change scenarios in the dataset. Segments of trajectories from all participants before the host car starts turning towards the target lane are transformed to B-spline form β. Two seconds of historical trajectories are used for input. The prediction is the future trajectory in the next 2s after obtaining the historical observations.

TABLE I: Number of the trajectories in the dataset for each vehicle.

Vehicle	s	f	r	rt	ft	st
No.	870	705	764	214	233	43

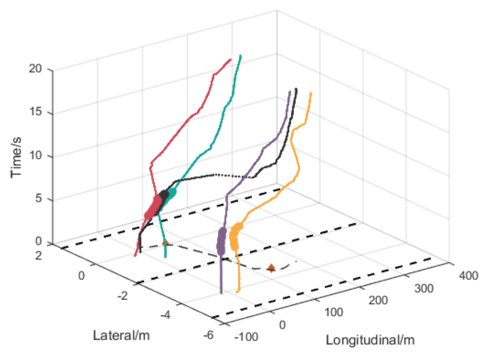


Fig. 3: Examples of the prediction procedure. The black trajectory is for the target vehicle; other colors indicate surrounding vehicles' trajectories. It shows an exmaple of right lane change. The X-axis is the longitudinal direction; the Y-axis is the lateral direction; the Z-axis is the time horizon. Dashed straight lines in the X-Y plane are lane dividers. The dashed curved line with orange dots in the X-Y plane are the 2-D projected paths for the target vehicles.

The timestep is 0.1s, and in total the horizon of the predicted trajectories is 2s.

Fig. 3 shows an example to illustrate the prediction procedure, as well as the input/output information. The method will predict a future trajectory of a target vehicle which is surrounded by at most 5 nearby vehicles. The target vehicle's trajectory is represented by the black line in each sub-figure, and the surrounding cars' are labeled by other colors. The highlighted segments (thicker parts along each trajectory) are used for the input of the method, which is the past 2 seconds of observations of all historical trajectories. To better illustrate the motion of the target vehicle, 2D paths (without temporal information) are also show as the dashed curves in the X-Y planes in Fig. 3. The segments between two orange dots are the 2D projection of the expected outputs. The expected outputs are the predicted trajectories of the target vehicles in the next 2 seconds. Instead of directly using the noisy raw data, trajectories are smoothed piecewise and fitted into B-splines.

A. *Statistical Results*

For a clear discussion and illustration, the results are separately visualized in longitudinal-time (Y-t) and lateral-time (X-t) domains. This separation helps the analysis of the spatial-temporal features of the predicted trajectories. Fig. 4 visualizes the tendency of the mean errors and standard deviations along the future time horizon. The X-axis starts from 2 seconds since the trajectory of 0 to 2 second is the input, so that the prediction starts at the 2-second mark. TABLEs II and III show the exact errors and standard deviations at selected time steps. Fig. 4a shows the mean errors and standard deviations of the predicted trajectory

TABLE II: Mean error (m) at given timestep for a predicted trajectory of a target car comparing with the groundtruth.

	2.3 s	2.6 s	2.9 s	3.2s	3.5s
TA, longitudinal	0.409	0.998	1.595	2.204	2.089
RKHS, longitudinal	1.425	3.551	5.672	7.951	9.925
RKHS, lateral	0.052	0.103	0.147	0.196	0.251

TABLE III: Standard deviation (m) at given timestep for a predicted trajectory of a target car comparing with the groundtruth.

	2.3 s	2.6 s	2.9 s	3.2s	3.5s
TA, logitudinal	0.504	3.207	8.850	18.11	30.92
RKHS, longitudinal	0.871	2.141	3.381	4.597	5.794
RKHS, lateral	0.051	0.101	0.145	0.195	0.250

comparing with the ground-truth and one baseline algorithm: Timed Automata(TA) [27]. TA is a learning-based algorithm to understand interactions between vehicles and then output a short-horizon prediction of future movement of a target vehicle. The TA model here is also trained from the same dataset. The red line with circles is the mean error curve from the proposed method; the red area depicts its standard deviation of the prediction along the time horizon. The green line with stars is the mean error curve, which is generated by the TA algorithm; the green area depicts the standard deviation. Both TA and the proposed method have increasing mean errors. But the mean error of TA is closer to zero. As shown in the first and the second row of TABLE II, the mean error of TA is almost 1 meter and 7.9 meters less than that of the proposed method at 0.3s and 1.5s, respectively. One possible reason that the proposed method has relatively large non-zero mean error is that the coefficients of the b-spline are treated equally. But the small error of the coefficient for the cubic term of the spline can heavily affect the shape, therefore results in a larger drift. However, the standard deviation of TA grows dramatically as the time horizon increase, which highly affects the stability and robustness of the prediction. As shown at the first and second row of TABLE III, the standard deviation of TA is about 1 meter and 25 meters larger than that of the proposed method at 0.6s and 1.5s, respectively. TA does not consider the long-horizon interaction, so it results in larger deviation than that of the proposed method.

Fig. 4b shows the mean error and standard deviation of the proposed method in the lateral direction comparing with the ground-truth trajectory when the target car makes the lane change. The solid blue line with triangles is the curve for mean errors along the time dimension, and the blue area shows the standard deviation. The TA algorithm does not explicitly consider the lateral direction. Thus the trajectory prediction in the lateral direction is only compared with the ground-truth in the dataset. The mean error and standard deviation grow as the time horizon increases. According to the third row in TABLEs II and III, the mean error and the standard deviation at the 1.5-second mark of the prediction are 0.251 and 0.250 meter, respectively.

B. Individual Examples

Fig. 5 shows an individual example scenario for the trajectory prediction in the lateral direction. The red line with circles is the prediction result. The green line with triangles is the ground-truth trajectory from real data. The blue line with stars is the error between the estimation and the ground-truth trajectories at each timestamp. Since the algorithm takes the first 2 seconds to collect historical observations, i.e., keeps track of all surrounding vehicles' trajectories, the prediction of the target car's trajectory starts from the the 2-second mark. The predicted trajectory and the ground-truth are close before 3s, and then they begin diverging. The error does not exceed 0.05m until the 3.5s timestamp. The largest error appears at the last timestamp, with the value of 0.23m. The result is consistent with the statistical results which are shown in Fig. 4b and TABLE II.

V. CONCLUSIONS

In this paper, we enhanced our previous non-parametric data-driven method to model interactions among vehicles and estimate their future trajectories, especially in a lane-change scenario. At most five surrounding vehicles are considered having mutual interactions with the target car in the scenario. Surrounding vehicles' past trajectories are considered in the prediction model. The predicted trajectory is continuous. The experimental results show that the proposed method provides continuous trajectory estimates with lower error than other methods, in terms of mean error and standard deviation at given future timestamps. The lack of training data for specific cases results in failure cases. In the future, more dynamic environments (inconsistent numbers of surrounding vehicles, in different scenarios) and trajectory representations will be studied to perform accurate behavioral understanding and trajectory prediction.

REFERENCES

[1] N. H. T. S. Administration *et al.*, "Preliminary statement of policy concerning automated vehicles," *Washington, DC*, pp. 1–14, 2013.

[2] C. Dong, Y. Zhang, and J. M. Dolan, "Lane-change social behavior generator for autonomous driving car by non-parametric regression in reproducing kernel hilbert space," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4489–4494.

[3] C. R. Baker and J. M. Dolan, "Traffic interaction in the urban challenge: Putting boss on its best behavior," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on.* IEEE, 2008, pp. 1752–1758.

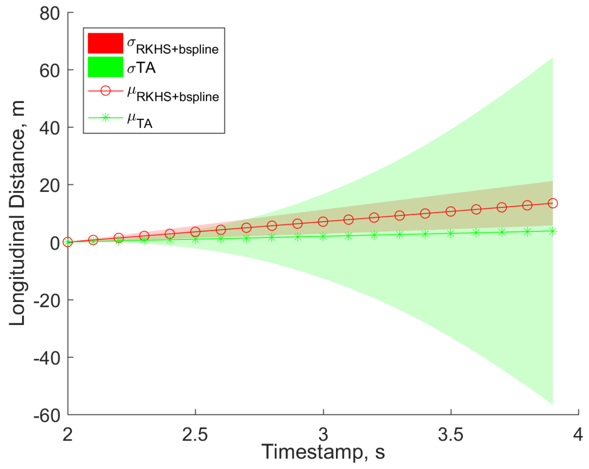
[4] J. E. Naranjo, C. Gonzalez, R. Garcia, and T. De Pedro, "Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 438–450, 2008.

[5] X.-Y. Lu and K. Hedrick, "Longitudinal control algorithm for automated vehicle merging," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 1. IEEE, 2000, pp. 450–455.

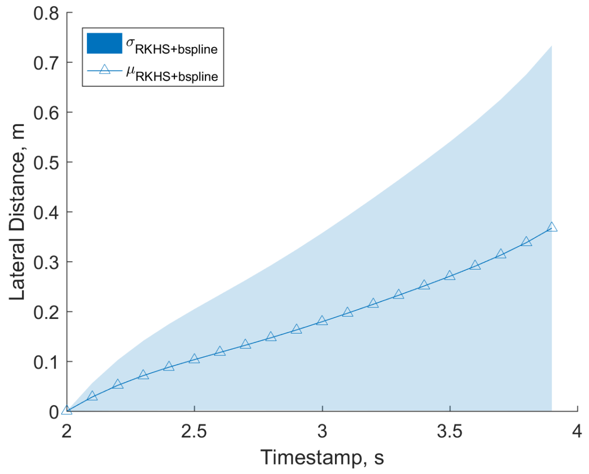
[6] C. Liu, C.-Y. Lin, Y. Wang, and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," in *American Control Conference (ACC), 2017.* IEEE, 2017, pp. 4177–4182.

[7] J. Nilsson, M. Brannstrom, E. Coelingh, and J. Fredriksson, "Lane Change Maneuvers for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2016.

[8] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.



(a) Errors and standard deviations in the longitudinal direction.



(b) Errors and standard deviations in the lateral direction.

Fig. 4: Statistical results of the mean errors and standard deviations for the proposed method and Timed Automata comparing with ground-truth, at each future timestamp (from 2.0 to 4.0 second).

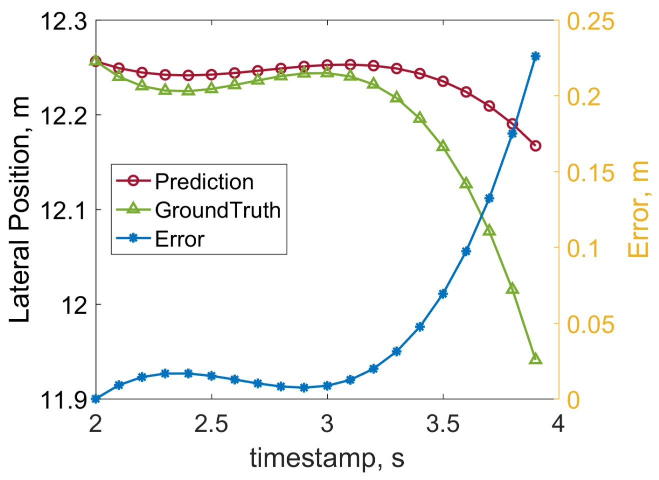


Fig. 5: An individual example for the trajectory estimation in the lateral direction. The X-axis is the timestamp. The left y-axis is the lateral position (corresponding to the red and green lines). The right orange y-axis is for the error (corresponding to the blue line).

[9] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[10] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, 2017, in Press.

[11] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving," in *2017 IEEE 28th Intelligent Vehicles Symposium (IV'17)*, Jun. 2017, pp. 1584 – 1589.

[12] S. Ulbrich and M. Maurer, "Probabilistic online POMDP decision making for lane changes in fully automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, oct 2013, pp. 2063–2067.

[13] J. Wei, J. M. Dolan, J. M. Snider, and B. Litkouhi, "A point-based mdp for robust single-lane autonomous driving behavior under uncertainties," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2586–2592.

[14] S. Paquet, L. Tobin, and B. Chaib-draa, "Real-time decision making for large pomdps," in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2005, pp. 450–455.

[15] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A POMDP approach," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.

[16] K. M. Seiler, H. Kurniawati, and S. P. N. Singh, "An online and approximate solver for pomdps with continuous action space," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2290–2297.

[17] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 204–211.

[18] Z. Qiao, K. Muelling, J. M. Dolan, P. Palanisamy, and P. Mudalige, "Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment," in *Intelligent Vehicles Symposium (IV), 2018 IEEE*. IEEE, 2018, pp. 1233–1238.

[19] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 66–73.

[20] Y. Chen, "Learning-based lane following and changing behaviors for autonomous vehicle," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, May 2018.

[21] Z. Marinho, A. Dragan, A. Byravan, B. Boots, S. Srinivasa, and G. Gordon, "Functional Gradient Motion Planning in Reproducing Kernel Hilbert Spaces," pp. 1–17, Jan 2016.

[22] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[23] M. A. Alvarez, L. Rosasco, N. D. Lawrence *et al.*, "Kernels for vector-valued functions: A review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[24] C. A. Micchelli and M. Pontil, "On learning vector-valued functions," *Neural computation*, vol. 17, no. 1, pp. 177–204, 2005.

[25] L. Song, J. Huang, A. Smola, and K. Fukumizu, "Hilbert space embeddings of conditional distributions with applications to dynamical systems," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 961–968.

[26] NGSIM, "U.S. Department of Transportation, NGSIM - Next generation simulation," http://www.ngsim.fhwa.dot.gov, 2007.

[27] Y. Zhang, Q. Lin, J. Wang, and S. Verwer, "Car-following behavior model learning using timed automata," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2353 – 2358, 2017, 20th IFAC World Congress.