Lane-Change Social Behavior Generator for Autonomous Driving Car by Non-parametric Regression in Reproducing Kernel Hilbert Space

Chiyu Dong*, Yihuan Zhang† and John M. Dolan‡

***Abstract*— Nowadays, self-driving cars are being applied to more complex urban scenarios including intersections, merging ramps or lane changes. It is, therefore, important for self-driving cars to behave socially with human-driven cars. In this paper, we focus on generating the lane change behavior for self-driving cars: perform a safe and effective lane change behavior once a lane-change command is received. Our method bridges the gap between higher-level behavior commands and the trajectory planner. There are two challenges in the task: 1) Analyzing the surrounding vehicles' mutual effects from their trajectories. 2) Estimating the proper lane change start point and end point according to the analysis of surrounding vehicles. We propose a learning-based approach to understand surrounding traffic and make decisions for a safe lane change. Our contributions and advantages of the approach are:**

1 **Considers the behavior generator as a continuous function in Reproducing Kernel Hilbert Space (RKHS) which contains a family of behavior generators;**
2 **Constructs the behavior generator function in RKHS by non-parametric regressions on training data;**
3 **Takes past trajectories of all related surrounding cars as input to capture mutual interactions and output continuous values to represent behaviors.**

Experimental results show that the proposed approach is able to generate feasible and human-like lane-change behavior (represented by start and end points) in multi-car environments. The experiments also verified that our suggested kernel outperforms the ones which were used in a previous method.

I. INTRODUCTION

As the autonomous driving industry grows faster, more self-driving cars or cars equipped with ADAS start running in public roads. Google starts testing in Mountain View urban areas earlier; Uber has been testing self-driving cars in Pittsburgh urban neighborhoods. GM, Audi, Tesla and Mercedes have already released ADAS features for their commercial vehicles. Those cars can perform level 3 autonomy according to NHTSA's "Levels of automation" [1]. However, the techniques are not mature enough to manage complex scenarios, such as intersections, ramp-merging and lane changes, which involve negotiations, intention understanding and social behaviors among traffic participants. These scenarios not only require the autonomous driving car to have robust perception and control, they also require that the car shares the road with human-driven cars. This suggests that the car needs to be able to behave socially with others. There are two aspects of social behavior: 1) Correctly understand human drivers' intentions or human driving styles. 2) React properly, similarly to humans.

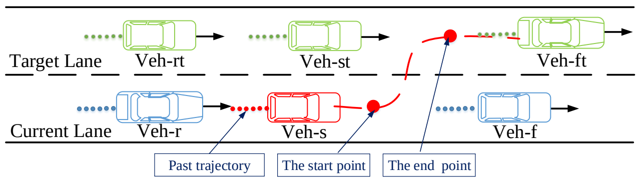

Fig. 1

In this paper, we propose a method to address cooperative lane change, with a representative scenario shown in Fig. 1. In the proposed method, we integrated these two aspects into a regression: understanding intentions by a collection of related traffic participants' past trajectories; and generating a proper start point and an end point for lane changing. In our autonomous driving planning architecture [2], the proposed predictive lane-change behavior generator works as a module in the Behavioral Planner. The module bridges the gap between higher-level commands (i.e., left-lane-change/right-lane-change) from the mission planner to the trajectory planner. The outputs provide advisory information, i.e., start and end points of the lane-change behavior, for the lower-level planner to generate feasible trajectories. Given the start and end pose, there are various of ways to determine trajectories [3]–[5]. The proposed method has the following three features:

- Uses the dataset to generalize surrounding cars' effects on the autonomous car's lane change behaviors from dataset.
- Surrounding cars' and the autonomous car's past trajectories are applied as input to the method, thus historical information is also used.
- Formulates the lane change behavior generator as a function in Reproducing Kernel Hilbert Space, and evaluates the start and end points from an input by a non-parametric regression (RKHS estimator). Therefore, no fitting model is assumed.

In the following section, previous work on performing the interactive behaviors, especially for lane changes, is discussed; Section III-A overviews the 5-surrounding-car lane change scenario and functions of the behavior generator; Section III-B briefly introduces Reproducing Kernel Hilbert Space (RKHS) and formulation of the lane-change behavior generator in RKHS; Sections III-C and III-D introduce the

*Chiyu Dong is with Department of Electrical and Computer Engineering Carnegie Mellon University, Pittsburgh, PA 15213 USA

†Yihuan Zhang is currently a visiting scholar in the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA. He is a Ph.D candidate in TongJi University, Shanghai, 200092, China.

‡John M. Dolan is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA

regression method and the kernel w.r.t. the RKHS; Section IV gives experimental results.

II. Related Work

Numerous cooperative planning algorithms for autonomous driving have been proposed. There are three major categories of methods to address the social cooperation problem among cars and tackle the lane changing problem:

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

A. Rule-based methods

The rule-based methods are the most straightforward approaches. They have been applied on test vehicles since the 2007 DARPA Urban Challenge. Baker and Dolan [6] developed CMU Boss's merge planner using a slot-based approach. Kinematic information is used to check merge-in feasibility of each slot, such as the distance to the Goal, remaining distance in the current lane, etc. 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 be implemented and proven robust to 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. [7] perform lane-change decision making by using fuzzy logic. The method is also straightforward and simple to implement. However, it also does not consider prior knowledge and prediction either.

B. Optimization-based methods

Nilson et al. [8] formulated cooperative planning as an optimization problem under a Model Predictive Control (MPC) framework. The weighted effects of acceleration and braking are optimized subject to the trajectory's shape and feasibility. The author provided a straightforward way to transform the problem into a well-defined optimization problem that can be solved by applying a specific solver. However, the manual tuning of weights is difficult. Also, the equation to be optimized and objective functions are also designed by hand, without the use of data.

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 for penalty. 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. Yao et al. [10] search for k-nearest-neighbors in a lane-change scenario database to generate a trajectory. Measuring differences between trajectories and scenarios remains a problem. And if the dataset contains a large number of samples, searching for the k-nearest-neighbors is time-consuming. Galceran and Cunningham et al. [11], [12] make the decision 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. [13], [14] detect whether the other car will merge in by using PGM. However, this method only provides binary output of either Yield or Not Yield.

Ulbrich et al. [15] and Wei et al. [16] proposed an online POMDP for lane-change using real-time belief space search [17] . 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. [18] 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). Seiler et al. [19] proposed an online and approximate solver for a continuous action POMDP, but only tested in toy problems. The POMDP solutions above still need manually designed probabilistic transition models and reward functions. Sadigh et al. and Hadfield et al. [20], [21] establish those transition models by (inverse) reinforcement learning, but their solutions are limited to the specific scenario, such as the numbers of traffic participants.

III. Method

A. The lane change scenario.

In our proposed method, the behavior generator is formulated as a function of the related surrounding cars. As shown in Fig. 1, the trajectories of all related surrounding cars and the autonomous car are taken as input. The related surrounding cars of the autonomous car include the leading car and the following car in the current lane, and the immediate neighboring car next to the autonomous car in the target lane and its leading and following cars. The output of the method is the suggested lane-change behavior, which is represented by the lane-change start point and the end point. Ideally, the start point of the lane-change behavior is defined as the position where the autonomous car's heading depart from the orientation of the current lane; the end point is the position where the autonomous car's heading converges to the orientation of the target lane.

B. Formulate the behavior generator in RKHS.

The reproducing Kernel Hilbert Space (RKHS) representation for planning is introduced by Marinho et al. [22]. In this work, a trajectory is explicitly described by Gaussian radial basis functions, using a functional gradient to optimize a cost functional to avoid static obstacles or navigate a high-dimensional arm. However, they did not explore the opportunity to apply similar methods to dynamic environments and cooperative scenarios. We follow their formulation, but instead of using functional gradient to find the optimal solution, our proposed method relies on RKHS non-parametric regression and a prior dataset to estimate continuous values i.e., the start/end points.

Reproducing Kernel Hilbert Space $\mathbf{H}$ contains families of smooth functions which are defined by a Mercer Kernel. The Mercer Kernel is a continuous mapping $K : \mathcal{X} \times \mathcal{X} \to \mathrm{I\!R}$, i.e., $K(x, y) =< f, g >_H$, where $f := K_x, g := K_y,\ f, g \in\ \mathbf{H}$. A function f in $\mathbf{H}$ 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. With the help of the kernel, it is possible to evaluate the function without explicit definition of the function (or basis functions) in the high-dimensional functional space [23]. A behavior generator is a function $\mathbf{F} : \mathcal{X} \to \mathcal{B}$, which maps a vector of trajectories ($\mathbf{X} \in \mathcal{X}$) to a behavior ($b \in \mathcal{B}$). $\mathcal{X}$ is a coordinate space which contains vectors of N surrounding vehicles' past trajectories $\mathbf{X} \overset{\text{def}}{=} \{x_i\}_1^N,\ x_i \in \mathrm{I\!R}^{\mathrm{T}}$. 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 $\mathbf{F}(\mathbf{\Gamma}) = \{[f_1(X_1), ..., f_D(X_1)], ..., [f_1(X_N), ..., f_D(X_N)]\}$. The output range $\mathcal{B} \subseteq \mathrm{I\!R}^{\mathrm{D}}$ represents the behavior. In the lane-changing problem, we are interested in two points: the start and the end points of the lane-changing behavior. Thus $D = 2$ in the current setup. Since the dimension of the range (the output domain) is $D > 1$, 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., $\mathcal{X} \times \mathcal{X} \to \mathrm{I\!R}^{\mathrm{D \times D}}$,

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

Parallel to the scalar-valued kernel, the matrix-valued kernel has the reproducing property which is also given by the Representer Theorem [24], [25] :

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

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

C. Non-parametric regression for the end points in RKHS

As the kernel representation of the behavior generator function was defined in the paragraph above, the function should be estimated from data and properly evaluated at given input. Note that we do not explicit define the form of the function; instead, we use linear combination of kernels, as mentioned in Equation 2. Once the kernel is decided (often given by users or separately learned from data), the only parameter left to be optimized is the coefficient $\boldsymbol{\alpha}$. Thus the approximation results in 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 behavior output, $J(f)$ is the penalty term. Here $||f||_H$ is used as the penalty term (or the regulation term). And the coefficient has a closed-form solution [24], [25]:

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

Substituting the evaluation from Equation 4 into Equation 2 yields the estimated behavior generator function $\hat{f}$. Given a new input X', the estimated behavior $\hat{b}$ 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 behaviors. K^* is the new kernel result given incoming input, a $D \times ND$ matrix. The regularization factor λ leverages the smoothness and accuracy of the regression function. Note that the $(K + \lambda I)^{-1} \mathbf{b}$ part can be pre-calculated offline given the training samples. Once an input comes in, only the K^* will be re-evaluated, and matrix multiplication is preformed with the pre-calculated $(K + \lambda I)^{-1} \mathbf{b}$.

D. Kernels

The essential part of the method is the kernel. As mentioned above, the input and data are matrices, i.e., in $K(X_1, X_2)$, X_1, X_2 are TN-dimensional matrices, where T is the period of time, and N is the number of surrounding vehicles (including the autonomous car itself). In this problem setup, there are six vehicles which need to be taken into consideration: five surrounding cars and the autonomous driving car. Then the problem is to use a proper kernel to calculate the inner product in RKHS. We use the inverse multiquadric kernel [23], [26]:

$$K(X_1, X_2) = \frac{1}{\sqrt{||X_1 - X_2||^2 + c}} \quad \text{where } c > 0 \tag{6}$$

Since the input $X_1, X_2 \in \mathcal{X}$ are matrices, the norm should measure the distance between two matrices. The kernel can be constructed using the Hilbert-Schmit norm, (a.k.a. the Frobenius norm) or the Spectral norm [27]:

$$||A||_F = \sqrt{tr(A^T \cdot A)} \tag{7}$$

where $tr(\cdot)$ is the trace function, $A = X_1 - X_2$.

$$||A||_S = ||A||_2 = \sqrt{\lambda_{max}(A^T \cdot A)} \tag{8}$$

Both the Frobenius norm and the Spectral norm consider the singular values of two matrices' difference. Since in Equation 7,

$$\sqrt{tr(A^T \cdot A)} = \sqrt{\sum \sigma_i^2} \tag{9}$$

σ_i is the i-th singular value of the matrix A, and in Equation 8,

$$\sqrt{\lambda_{max}(A^T \cdot A)} = \max_i \sigma_i \tag{10}$$

The singular values measure the major differences between two matrices which contain two trajectories.

IV. Experimental Results

In the experiments, real data are used in training and testing. Lane-change scenarios with all participants, as in Fig. 1, are grouped and extracted from the dataset. Each group contains one host car (Veh-s) and surrounding cars, i.e., Veh-f, Veh-r, Veh-rt, Veh-ft, Veh-st in Fig. 1. The trajectory of every car in the group is recorded from 10 seconds before to 10 seconds after the host car (Veh-s) crosses the lane-marking. Segments of trajectories from all participants before the host car starts turning towards the target lane are taken as input X. For training, the real start and end points' positions are considered known values $\mathbf{b}$ to obtain the parameter α in Equation 4. Knowing the coefficient parameter α, in testing, a new kernel response K^* is calculated by using new input and segments in the training set, to finally obtain the estimation $\hat{b}$. Results of the testing are the start and end points of the lane-change behavior. These points are compared with ground-truth, which is extracted from the same dataset. The program runs in real-time with an Intel Core i7 level processor in single thread on a standard laptop. The training process takes only a few seconds; the average update time for evaluating a new input is 0.09s.

A. Data Description

The public dataset of individual vehicle trajectories we use in this paper is from NGSIM [28], a program funded by the U.S. Federal Highway Administration. These trajectory data are so far unique in the history of traffic research and provide a great and valuable basis for the validation and calibration of microscopic traffic models. We test our method on the datasets from the I80 and the US101 highways.

The I80 dataset consists of three 15-minute periods: 4:00 p.m. to 4:15 p.m., 5:00 p.m. to 5:15 p.m., and 5:15 p.m. to 5:30 p.m. These periods represent the buildup of congestion, or the transition between uncongested and congested conditions, and full congestion during the peak period [28]. A total of 45 minutes of data are available in the US101 dataset, which are segmented into three 15-minute periods: 7:50 a.m. to 8:05 a.m., 8:05 a.m. to 8:20 a.m., and 8:20 a.m. to 8:35 a.m. [28]. In both the I80 and the US101 datasets, vehicle trajectory data provide precise location of each vehicle within the study area every one-tenth of a second.

TABLE I: NGSIM data features.

Feature	Definition
Vehicle Speed (m/s)	Speed of vehicles in current lane and target lane
Longitudinal Position (m)	Longitudinal Position of vehicles in current lane and target lane
Lateral Position (m)	Lateral Position of vehicles in current lane and target lane
Vehicle Length (m)	Length of vehicle

```
Algorithm 1 Lane change extraction in NGSIM:
Input: Original dataset of NGSIM $S_{ori}$, lateral offset threshold $d_{ref}$, heading orientation threshold $\theta_{ref}$, match points threshold $D_{ref}$
Output: Lane change trajectories along with surrounding vehicles' trajectories, start and end points
Format the original dataset into id-indexed dataset $S_{id}$ and time-indexed dataset $S_t$;
for all id in $S_{id}$, do
    Find periods $T_{lc}$ in which lateral offset is larger than $d_{ref}$;
    Find surrounding vehicles' states in periods $T_{lc}$;
    for all trajectories of subject vehicle in $T_{lc}$, do
        Calculate heading orientation $\theta_{lc}$ and mark the points where $\theta_{lc}$ = $\theta_{ref}$;
        Choose the closest two points as primary start and end points $P_s, P_e$;
        Use curve fitting method to fit the lane change trajectory and find two intersection points $Q_s, Q_e$;
        if Distance($P_s$,$Q_s$) $\leq D_{ref}$ & Distance($P_e$,$Q_e$) $\leq D_{ref}$ then
            Mark $P_s, P_e$ and store trajectories according to time frames. m
        end
    end
end
```

B. Lane change extraction and start/end points determine for ground-truth.

Based on the trajectory data, the lane change trajectories along with surrounding vehicles' trajectories are extracted for the purpose of lane change social behavior studying[1]. Table I shows a summary of data features used in our paper. Note that speed, longitudinal position, lateral position, length and width are attributes for subject and surrounding vehicles. Extraction details are presented in Algorithm 1. Suggested value for d_{ref} is half of the lane width ($1.875m$), D_{ref} is equal to the lane width ($3.75m$). As shown in Fig. 2, blue dots represent the lane change trajectory, start and end points are presented in green and red dots. The car traverses from the left-hand-side to the right-hand-side of the frame. The first two figures show example of left lane changes; the last two figures are examples of right lane changes.

[1]https://github.com/donnydcy/LC_NGSIM

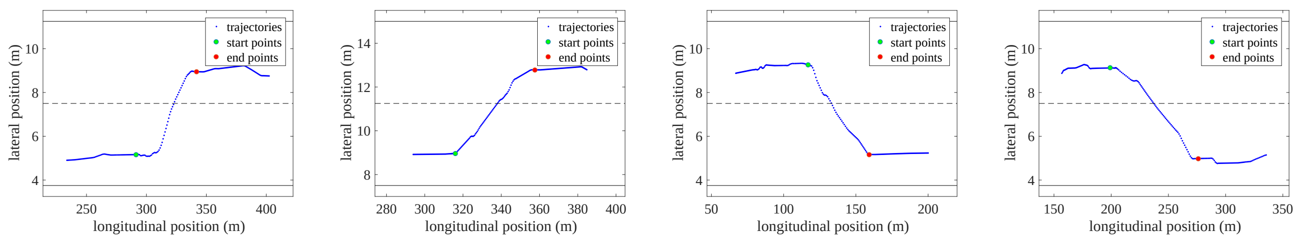

Fig. 2: Examples of lane change trajectories (blue dots) which are extracted from the dataset. The start and end points are labelled green and red, respectively. The car moves from left to right in each plot. The first two figures are examples for left-lane-change scenarios, and the last two are for right-lane-change scenarios.

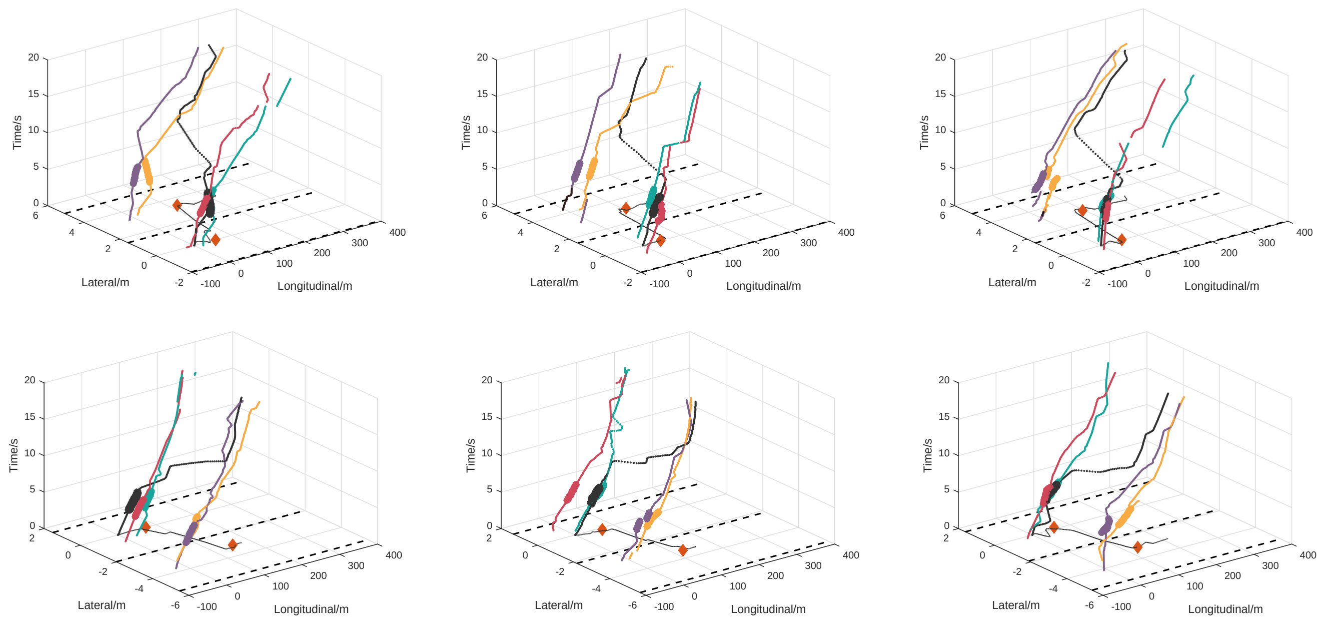

Fig. 3: Examples and results of the estimation. Vertical axis is time (s). Black curves are host vehicles' trajectories; colored curves are surrounding cars'; the black curves on the $Time = 0$ plane are the projected paths of the host vehicles; Dashed straight lines are lane dividers; Red diamonds are predicted start/end points. Highlighted segments on the curves are used to predict the start/end points.

C. Results compared with the ground-truth

We extracted 543 lane changing scenarios from the US-101 and I-80 data. As Fig. 1 shows, at most five surrounding cars and six trajectories are considered: five from surrounding cars and one from the host car itself.

450 groups of trajectories are randomly selected as training sets, and the remaining 93 groups are used for testing. To concentrate on the recent past, training trajectories are pruned and only retain the last 30 steps (3 seconds) before the host vehicle starts the lane-change (when the heading departs from the orientation of the current lane.).

Four kernels are tested: Laplacian RBFs, Gaussian RBFs which are suggested in [22] or inverse multiquadric kernels (IMK) constructed by with the Frobenius norm $||\cdot||_F$ and the Spectral norm $||\cdot||_S$. Results are shown in Table II. μ_{start} is the difference in the start point between the estimate and the ground-truth, and σ_{start} is the standard deviation. μ_{end} is the difference in end point between the estimate and the ground-truth, and σ_{end} is the standard deviation.

Table II shows that the RBFs have large errors. However, as the analyzed in section III-D, the Frobenius norm and Spectral norm, work similarly, both significantly outperform the other kernels. In terms of standard deviation, the performance of the Spectral norm is slightly better than that of the Frobenius norm. In the second row, the widely used Gaussian RBF kernel, which is also used in [22], performs worse than the inverse multiquadric kernels with Frobenius norm and Spectral norm.

Fig. 3 shows the start/end point predictions of six scenarios from the testing group. Highlighted segments are used for prediction, and are the only input of the proposed method.

TABLE II: Statistical results for different kernels compared with ground-truth, All units are meters (m).

kernels	μ_{start}	μ_{end}	σ_{start}	σ_{end}
Laplacian RBF[22]	-54.90	-116.61	24.10	43.40
Gaussian RBF[22]	-13.55	-31.15	16.20	25.42
IMK with $||\cdot||_F$	-0.95	-18.50	6.38	13.77
IMK with $||\cdot||_S$	1.78	-17.90	5.87	13.10

The segments consist of all traffic participants' trajectories in a 3-second time window. (Takes left-lane-change as an example, all traffic participants in a lane changing scenario are defined in Fig. 1.) The red diamonds are the output of the method, which indicate the start points and the end points of the lane-change behavior. The real lane-change paths generated by human drivers are shown as the black curves on the $Time = 0$ plane. The red diamonds (the outputs) are close to the turn points of the black curves, which indicates that the predicted start/end points correspond to feasible lane-change behaviors.

V. CONCLUSIONS

In this paper, we proposed a novel social behavioral method for autonomous driving vehicle to estimate the lane change start point and the end point. The behavior generator is formulated as a function in Reproducing Kernel Hilbert Space, which is obtained by a non-parametric regression with kernels. We also suggest using the inverse multiquadric kernels that are constructed by the Frobenius norm or Spectral norm. In the training process, a linear operator is obtained, which is formed by collection of training data and their kernel values. After finding this operator, given a new input, the behavior generator function can be evaluated by multiplying the linear operator by the kernel response of the input. Experimental results show that the proposed method with the suggested kernels can estimate the start/end points of lane changing accurately, with limited mean error and standard deviation that outperform other kernels.

In the future, instead of estimating start/end points, the method will be further developed to generate a trajectory, as is done in [22];

ACKNOWLEDGMENT

Thanks to Dr. Tianyu Gu, Dr. Yuqing Tang and Sasanka Nagavalli for discussions on path planning and POMDP.

REFERENCES

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

[2] J. Wei, J. M. Dolan, and B. Litkouhi, "Autonomous vehicle social behavior for highway entrance ramp management," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 201–207.

[3] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2061–2067.

[4] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 250–256.

[5] W. Luo, N. Chakraborty, and K. Sycara, "Distributed dynamic priority assignment and motion planning for multiple mobile robots with kinodynamic constraints," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 148–154.

[6] 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.

[7] 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.

[8] J. Nilsson and J. Sjöberg, "Strategic decision making for automated driving on two-lane, one way roads using model predictive control," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1253–1258.

[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] W. Yao, H. Zhao, P. Bonnifait, and H. Zha, "Lane change trajectory prediction by using recorded human driving data," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 430–436.

[11] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction." in *Robotics: Science and Systems*, 2015.

[12] ——, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, 2017, in Press.

[13] 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.

[14] ——, "Interactive ramp merging planning in autonomous driving: Multi-Merging leading PGM (MML-PGM)," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (ITSC2017)*, Oct. 2017 ACCEPTED.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 3909–3917.

[22] 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.

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

[24] 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.

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

[26] C. A. Micchelli, "Interpolation of scattered data: distance matrices and conditionally positive definite functions," in *Approximation theory and spline functions*. Springer, 1984, pp. 143–145.

[27] R. Horn and C. Johnson, *Norms for vectors and matrices*. Cambridge, England: Cambridge University Press, 1990.

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