Accepted by IROS 2017, Draft Version

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

Chiyu Dong[1], Yihuan Zhang[2] and John M. Dolan[3]

***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 autonomous driving cars: how to perform a safe and effective lane change behavior once received lane-change commands. Our method bridges the gaps between the higher level behavioral indications and the trajectory planners. There are two challenges in the task: 1) Estimating the surrounding vehicles' mutually effects from their trajectories which are high-dimensional or even continuous functions; 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 understandings and social behaviors among traffic participants. These scenarios not only require the autonomous driving car having robust perception and control, also demand the car sharing roads with human-driven cars. This suggests the car to be capable 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 human.

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

[2]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.

[3]John M. Dolan is with the The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA

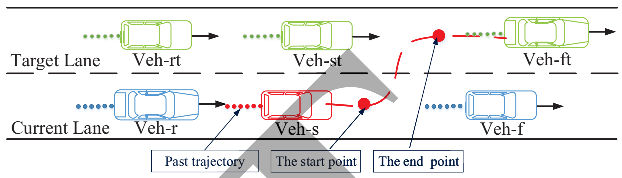

Fig. 1: A left lane change scenario. The red car (Veh-s) is the autonomous car (host car), which received a left-lane-change command; The blue cars are the leading car (Veh-f) and the following car (Veh-r) in the current lane, respectively; The vehicles in the target lane are also considered: the immediate left car (Veh-st), its leading car (Veh-ft) and its following car (Veh-rt). The method will generate a pair of start/end points to the left-lane-change behavior. The red dashed line just indicates the lane-change behavior, not necessary to be considered as a planned path.

In this paper, we proposed a method to address cooperative lane change, one scenario is 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 lane-change reaction by giving a proper start point and an end point. According to the autonomous driving planning architecture [2], the proposed predictive lane-change behavioral 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 following level planner to generate feasible trajectories. Given the start and end pose, there are numbers of methods to provide trajectories, [3], [4]. The proposed method has the following three highlights:

- Generalize surrounding cars' effects to 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.
- Formulate the lane change behavioral generator as a function in Reproducing Kernel Hilbert Space, and evaluate 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, prior arts to perform the interactive behaviors, especially for lane changes will be discussed; Section III-A overviews the 5-surrounding-car lane change scenario and functions of the behavioral generator; Section III-B briefly introduces Reproducing Kernel Hilbert Space (RKHS) and formulation of the lane-change behavioral generator in RKHS; Section III-C and III-D introduce the regression method and the kernel w.r.t. the RKHS; Section IV shows experimental results.

II. Related Work

As the autonomous driving techniques keep advancing, numerous cooperative planning algorithms have been proposed. There are three major categories of methods to address the social cooperation problem among cars and tackled the lane changing problem:

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

A. Rule-based methods

The rule-based methods are the most straightforward approaches. They are applied on testing vehicles since the 2007 DARPA Urban Challenge. Baker and Dolan [5] 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 and 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, lacking prior knowledge of surrounding vehicles' behaviors makes it hard to estimate or predict their movements and corresponding behaviors. Naranjo et al. [6] perform the lane-change decision making by using fuzzy logic. The method is also straightforward and simple to implement. However, it did not consider prior knowledge and prediction either.

B. Optimization-based methods

Nilson et al. [7] formulated the cooperative planning as an optimization problem under Model Predictive Control (MPC) framework. Weighted effect from acceleration and brake are optimized subject to the trajectory's shape and feasibility. The author provided a straight forward way to transform the problem into a well-defined optimization problem that can be solved by applying a specific solver. However, the manual tune 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

The probabilistic methods make the largest percentage of the solutions to lane changing or cooperative driving. Montemerlo et al.[8] integrated the 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 optimize a variant bellman equation, implicitly follows the MDP framework and Value Iterations. Each action is assigned cost for penalty. The lane changing behavior is a penalty term in the cumulative cost function which is optimized by the DP. Unfortunately, the algorithm did not consider other traffic participants. Yao et al. [9] 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 large number of samples, searching for the k-nearest-neighbors can be struggling. Galceran et al. [10] and Cunningham et al. [11] 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 and etc. , which can be used as upper level module upon our methods.

Ulbrich et al. [12] proposed a online POMDP for lane changing which benefits from real-time belief space search [13] and similar with Wei et al [14]. However, to achieve the real-time performance and straightly ultilize the POMDP framework, they discretized state and action spaces. To avoid discrete states, Bai et al, [15] proposed a continuous state POMDP using belief tree and the model was applied in navigating intersections. But its actions are discrete and represented by generalized policy graph (GPG). To address the challenge of continuous action space, Seiler et al [16] proposed an online and approximate solver for continuous action POMDP, but only tested in toy problems.

The POMDP solutions above still need manually designed probabilistic transition models and rewards functions. Sadigh et al. and Hadfield et al. [17], [18] establish those transition models by (inverse) reinforcement learning, but those solutions are limited to the specific scenario, such as metric/geometry of roads or intersections, numbers of 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 includes the leading car and the following car on the current lane, and the immediate neighboring car next to the autonomous car on 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 departure 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.

Reproducing Kernel Hilbert Space (RKHS) representation for planning is introduced by Marinho et al. [19]. In this work, trajectory is explicitly described by the Gaussian radial basis functions. Using 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 deploy similar methods over dynamic environment 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 prior dataset to estimate continuous value (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 $K : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$, 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 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. This is the so called the "kernel trick" in Machine Learning areas [20]. 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 vehicle's past trajectories $\mathbf{X} \overset{\text{def}}{=} \{x_i\}_1^N,\ x_i \in \mathbb{R}^{\mathrm{T}}$. T is the length of the interested historical poses. The input contributes to all elements in the output vector, use $\mathbf{\Gamma} = \{X\}_i^N$ as the training set, 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 $\mathcal{B} \subseteq \mathbb{R}^{\mathrm{D}}$ 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 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., $\mathcal{X} \times \mathcal{X} \to \mathbb{R}^{\mathrm{D\times D}}$,

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

Parallel to the scaler-valued kernel, the matrix-valued kernel has the reproducing property which is also given by the Representer Theorem [21]–[23] :

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

$\cdot$ is the normal inner product in Euclidean Space, $\boldsymbol{\alpha}$ is a ND-dimension coefficient. By constraining the behavior function $\mathbf{F}$ into the RKHS, we assumed that the function $\mathbf{F}$ is continuous and can be represented by linear combination of a set of basis. The basis are unknown to us, and we do not interested on 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 yields 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 close-form solution [22], [23]:

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

Substituting the evaluation from Equation 4 to 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 smoothies 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 perform matrix multiply with the pre-calcuated $(K + \lambda I)^{-1}\mathbf{b}$.

D. Kernels

The essential part of the method is the kernel. As mentioned in the paragraph above, the input and data are matrices, i.e., in $K(X_1, X_2)$, X_1, X_2 are TN-dimension matrices, where T is the interested period of time, 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 [20], [24]:

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

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 by Hilbert-Schmit norm , (a.k.a, Frobenius norm) or the Spectral norm [25]:

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

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 of 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. Trajectory of every car in the group is recorded from 10 seconds before to 10 seconds after the host car (Veh-s) acrossing the lane-marking. Segments of trajectories from all participants before the host car starting 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^* will be calculated by using new input and segments in the training set, and 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. Training process takes only few seconds; the average update rate of evaluating a new input is 0.09s.

A. Data Description

The public dataset on individual vehicle trajectories we use in this paper is from NGSIM [26], a program funded by the U.S. Federal Highway Administration. This trajectory data is so far unique in the history of traffic research and provides a great and valuable basis for the validation and calibration of microscopic traffic models. The I80 and the US101 are two datasets we test our method in the datasets from the I80 and the US101 highways.

The I80 dataset consists 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 [26]. 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. [26]. In both of the I80 and the US101 datasets, vehicle trajectory data provides 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 θ_{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} which lateral offset is large than d_{ref};

  Find surrounding vehicles' states in periods T_{lc};

  for *all trajectories of subject vehicle in* T_{lc}, **do**

    Calculate heading orientation θ_{lc} and mark the points where θ_{lc} = θ_{ref};

    Choose the closest two points as primary start and end points P_s, P_e;

    Using 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.

    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. 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. As shown in Fig. 2, blue dots represent the lane change trajectory, start and end points are presented in green and red dots.

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

kernels	μ_{start}	μ_{end}	σ_{start}	σ_{end}
Laplacian RBF[19]	-54.90	-116.61	24.10	43.40
Guassian RBF[19]	-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 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 exmaples of right lane changes. The extracted lane-changing groups will be manually double-checked and published with the paper.

C. Results compared with the ground-truth

We extracted 543 lane changing scenarios from the US-101 and I-80 data. The data arrangement refers to Fig. 1, i.e, at most five surrounding cars and six sequences of trajectories are considered: five from surrounding cars and one from the host car itself.

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

Four kernels are tested: Laplacian RBFs, Gaussian RBFs which are suggested in [19] and inverse multiquadric kernels (IMK) that are constructed by Frobenius norm $||\cdot||_F$ and Spectral norm $||\cdot||_S$. Results are shown in Table II. μ_{start} is the difference of start points between estimations and the ground-truth, σ_{start} is the standard deviation . μ_{end} is the difference of end points between estimations and the ground-truth, σ_{end} is the standard deviation.

From Table II, it shows that the polynomial kernel has the works perform, literally helps nothing. However, as the analyzed in section III-D, Frobenius norm and Spectral norm works similarly and both of them significantly outperform other kernels. And in term of standard deviation, the performance of Spectral norm is slightly better than Frobenius norm's. In the second row, the widely used Gaussian RBF kernel, which is also used in [19], performs worse than the inverse multiquadric kernels with Frobenius norm and Spectral norm.

Fig. 3 show the start/end points predictions of six scenarios from the testing group. Highlighted segments are used for prediction, which are the only input of the proposed method. The segments consist of all traffic participants' trajectories in a 3-seconds time window. (Take 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 that 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 car to estimate the lane change start point and the end point. The behavioral 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 Frobenius norm or Spectral norm. In the training process, a linear operator is obtained, which is consisted by collection of training data and its kernel values. After knowing this operator, given a new input, the behavioral generator function can be evaluated by multiplying between the linear operator and 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, which outperform other kernels.

In the future, there are two main challenges: 1) instead of estimating start/end points, it is more useful to directly generate a trajectory, as been represented in [19]. 2) developing more powerful kernels to capture differences between trajectories and interactions among traffic participants.

ACKNOWLEDGMENT

Appreciate Dr. Tianyu Gu and Sasanka Nagavalli for fruitful 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] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.

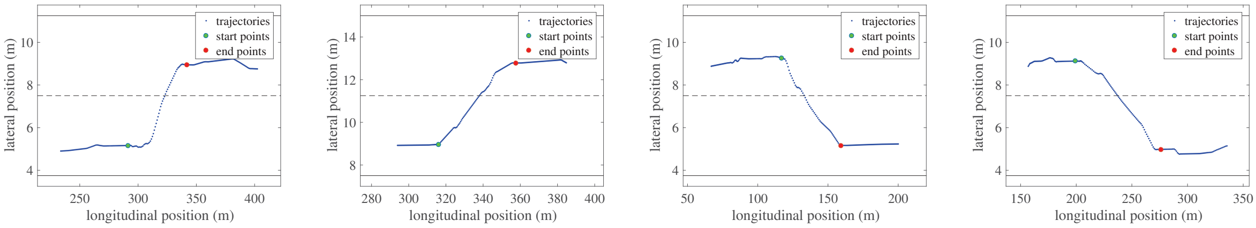

Fig. 2: Examples of lane change trajectories (blue dots) which are extracted from the dataset. The start and end points are labelled green and red, respectively. The car moves from left to right. The first two figures are examples for left-lane-change scenarios, and the last two are for right-lane-change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

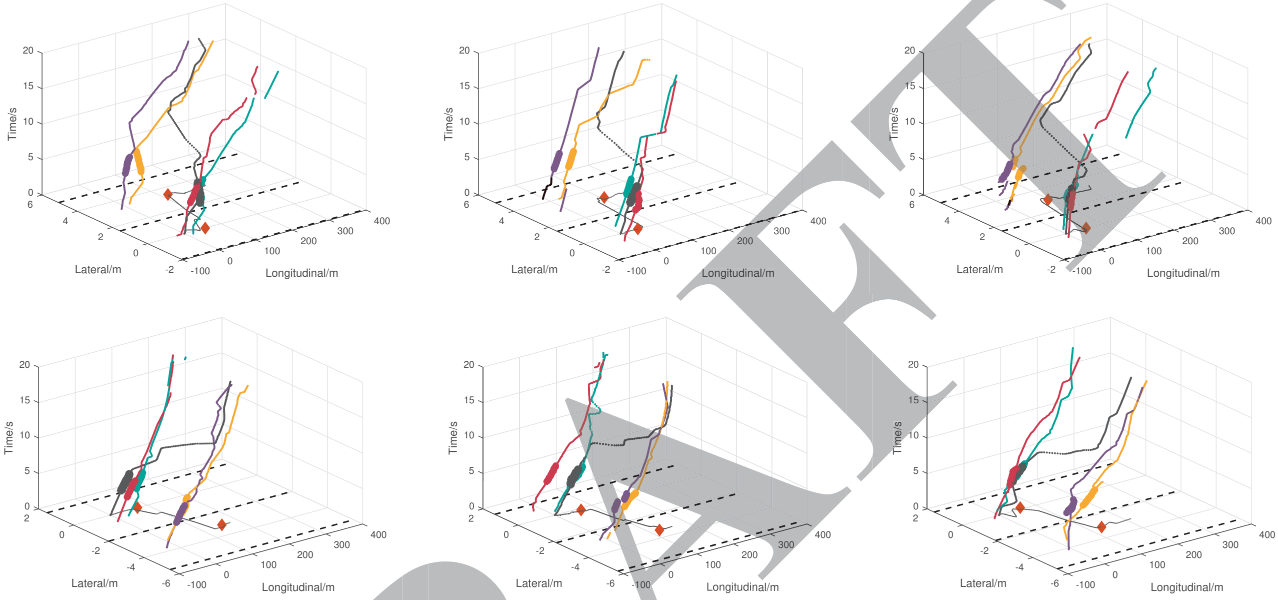

Fig. 3: Examples and results of the estimation. Vertical axis is time (s). Black curves are host vehicles' trajectories; colored curves are surrounding cars'; the black curves on the $Time = 0$ plane are the projected paths of the host vehicles; Dashed straight lines are lane dividers; Red diamonds are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

[11] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "Mpdm: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1670–1677.

[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] 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.

[14] 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.

[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] 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.

[18] 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.

[19] 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.

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

[21] G. S. Kimeldorf and G. Wahba, "A correspondence between bayesian estimation on stochastic processes and smoothing by splines," *The Annals of Mathematical Statistics*, vol. 41, no. 2, pp. 495–502, 1970.

[22] 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.

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

[24] 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.

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

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