Interaction-aware Kalman Neural Networks for Trajectory Prediction

Ce Ju[1,†], Zheng Wang[2,†], Cheng Long[2], Xiaoyu Zhang[3] and Dong Eui Chang[4]

***Abstract*— Forecasting the motion of surrounding obstacles (vehicles, bicycles, pedestrians and etc.) benefits the on-road motion planning for intelligent and autonomous vehicles. Complex scenes always yield great challenges in modeling the patterns of surrounding traffic. For example, one main challenge comes from the intractable interaction effects in a complex traffic system. In this paper, we propose a multi-layer architecture *Interaction-aware Kalman Neural Networks* (IaKNN) which involves an interaction layer for resolving high-dimensional traffic environmental observations as interaction-aware accelerations, a motion layer for transform the accelerations to interaction-aware trajectories, and a filter layer for estimating future trajectories with a Kalman filter network. Attributed to the multiple traffic data source, our end-to-end trainable approach technically fuses dynamic and interaction-aware trajectories boosting the prediction performance. Experiments on the NGSIM dataset demonstrate that IaKNN outperforms the state-of-the-art methods in terms of effectiveness for traffic trajectory prediction.**

I. Introduction

An autonomous driving system can be broadly categorized into three hierarchical subsystems, namely *perception/localization*, *planning* and *control* [1], [2]. The perception subsystem refers to the ability to acquire information from the environment via multiple vehicle sensors like GPS, LiDAR, RADAR, and Camera. It categorizes sensor data by their semantic meaning; The localization subsystem determines the global and local position of ego-vehicle with respect to High Definition Map and the vehicle's coordinate system respectively; The planning subsystem typically includes mission planning, behavioral planning, and motion planning, generating an efficient and safe trajectory (specific positions and associated target velocities) for the autonomous vehicle to follow from a start waypoint to a goal waypoint [3]; The control subsystem refers to the ability to execute the planned actions in trajectories from planning by sending accelerations, brake, and steering messages to the actuator on intelligent vehicles.

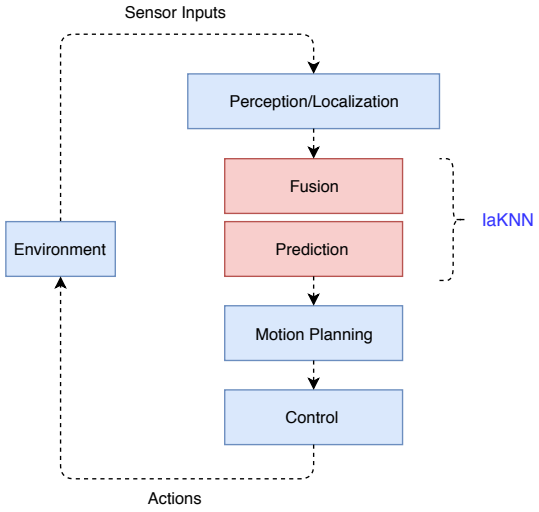

Fig. 1: Illustration of autonomous driving system overview

Each subsystem in an autonomous driving system has technical difficulties from both hardware and software. For example, one hardcore for the planning is of generating ego-vehicle trajectories based on low-resolution perception information captured and estimated by the low-cost sensing subsystem. Moreover, forecasting the motion of surrounding static and dynamic obstacles is also a big challenge to most of the on-road autonomous driving systems. Attributed the tremendous progress in recent years, the planning with static obstacles has been adequately explored [4]–[7]. However, the immature technologies in sensing, computing and artificial intelligence still make planning with dynamic obstacles impossible. A practical and efficient solution to this challenge, which is widely adopted in the autonomous industry, is technically to make the on-road dynamic obstacles become almost static, for example, Baidu's open-source software platform for autonomous vehicles development (Apollo) [8].

To handle the above challenges, an independent subsystem *prediction* is laid after *perception* and before *motion planning* to weaken the risk of the on-road planing with dynamic obstacles by predicting the future motion of obstacles, referring to Figure 1. Specifically, an effective prediction subsystem needs to handle the on-road challenges including noisy sensing information and complex traffic scenes. Existing on-road prediction subsystem is categorized into three gradually intelligent models, namely the physics-based motion model, the maneuver-based motion model, and the interaction-aware motion model [9]–[11]. The physics-based motion model is the one based on the kinematics; The maneuver-based motion model is the one designed for a particular maneuver

† Equal Contribution
[1]WeBank Co., Ltd., AI Department, ceju@webank.com
[2]Nanyang Technological University, Computer Science and Engineering Department, wang_zheng@ntu.edu.sg, c.long@ntu.edu.sg
[3]University of Michigan, Ann Arbor, Electrical and Computer Engineering Department, zhxiaoyu@umich.edu
[4]Korea Advanced Institute of Science and Technology, Electrical Engineering Department, dechang@kaist.ac.kr
*This research was in part supported by WeBank Co., Ltd., the Institute for Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2019-0-01396, Development of framework for analyzing, detecting, mitigating of bias in AI model and training data), and by the ICT R&D program of MSIP/IITP [2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion].

in which the future trajectory of a vehicle is predicted by searching the trajectories which have been clustered as a priori. The interaction-aware motion model is the one which captures the interactive effects among vehicles by predicting the trajectories of multiple vehicles collectively. Especially, many recent interaction-aware motion models adopt deep learning approach [12]–[18].

In this paper, we propose a specific model for the prediction subsystem called *Interaction-aware Kalman Neural Networks* (IaKNN). IaKNN is a multi-layer architecture consisting of three layers, namely an interaction layer, a motion layer, and a filter layer. The interaction layer is a deep neural network with multiple convolution layers laying before the LSTM encoder-decoder architecture. Fed with the past trajectories of vehicles that are close to one another, this layer extracts the *accelerations* that capture not only those raw acceleration readings but also the interactive effects among vehicles in the form of social force, a latent variable (which is a measure of internal motivation of an individual in a social activity in sociology and has been used for studying the motion trajectories of pedestrians [19]). The extracted accelerations are called *interaction-aware accelerations*. The motion layer is similar to the existing physics-based motion model which transforms accelerations into trajectories by using kinematics models. Here, instead of feeding the motion layer with the accelerations read from sensors directly, we feed with those interaction-aware accelerations that are outputted by the interaction layer and call the transformed trajectories *interaction-aware trajectories*. The filter layer consists of mainly a Kalman filter for optimally estimating the future trajectories based on the interaction-aware trajectories outputted by the motion layer. The novelty in this layer is that we incorporate two LSTM neural networks [20] for learning the time-varying process and measurement noises that would be used in the update step of the Kalman filter, and this is the first of its kind for trajectory prediction. In summary, our IaKNN model enjoys the merits of both the physics-based model (the motion layer) and the interaction-based model (the interaction layer) and employs neural-network-based probabilistic filtering for accurate estimation (the filter layer). In experiments, we evaluate IaKNN on the *Next Generation Simulation* (NGSIM) dataset [21] and the empirical results demonstrate the effectiveness of IaKNN.

In summary, the major contributions of this paper are listed as follows: our approach, to the best of our knowledge, is the first neural network-based filtering algorithm for on-road trajectory prediction, which is end-to-end trainable to learn the time-varying process and measurement noises with LSTM neural networks in a Kalman filter. For the normal framework of an on-road autonomous driving system, the value of our methodology is the integration of the techniques in sensor fusion and data-driven approach motion prediction, referring to Figure 1, more practical to the problem in a dynamic system. We perform extensive experiments on the NGSIM dataset, which shows that IaKNN consistently outperforms the state-of-the-art methods in terms of effectiveness.

II. Related Work

1) State Estimation: State estimation is a mature subfield in robotics with the aim to estimate the state of a robot from various noisy measurements. One comprehensive survey of classic approaches of state estimation refers to [22]. The limitation in traditional state estimation models is lack of *prior* knowledge purified from database to be the preset parameter adapting time varying influences. Nowadays, neural network approaches have been explored largely for state estimation in autonomous industry. Coskun et al. [23] train the triple-LSTM neural networks architecture to learn the kinematic motion model, process noise, and measurement noise for estimating human pose in a camera image. Haarnoja et al. [24] adopt the discriminative approach in state estimation where neural networks is used to learn features from highly complex observations, and then filtered the features to underlying states.

2) Data-driven Approach Trajectory Prediction: Trajectory prediction, which is a traditional topic in the field of intelligent vehicle society, has been largely studied, referring to the survey [11], [25]. Among those methods for this topic, the data-driven ones are promising. For example, Ma et al. [17] propose an LSTM-based two-layers model TrafficPredict for heterogeneous traffic-agents in an urban environment.

To increase robustness and accuracy in multi-agent tracking problems, the data-driven fashion can model more complex "interactions" between agents than the hand-crafted functions. For example, Alahi et al. [12] propose a deep learning model to predict the motion dynamics of pedestrians in a crowded scene in which they build a fully connected layer called social pooling to learn the social tensor based on pedestrians. Gupta et al. [13] propose a GAN-based encoder-decoder framework for trajectory prediction with a pooling mechanism to aggregate information across people. In the field of intelligent vehicles, Deo and Trivedi [14] extract a social tensor with a convolutional social pooling layer and then feed the social tensor to a maneuver-based motion model for trajectory prediction. Kuefler et al. [15] and Bhattacharyya et al. [16] use imitation learning approach to learn human drivers' behaviors for trajectory prediction. The learned policies are able to generate the future driving trajectories that match those of human drivers better and can also interact with neighboring vehicles in a more stable manner over long horizons. Zhao et al. [26] design an encoder-decoder architecture called *multi-agent tensor fusion network* to extract multi-agent interactions with the spatial structure of agents and the scene context, and predict recurrently to agents' future trajectories.

Our IaKNN model differs from these models in two aspects. First, IaKNN captures the interactive effects in a form of accelerations which could then be feed to kinematics models and thus it enjoys the merits of both the classic Physics models and the data-driven process (of capturing the interactive effects). Second, IaKNN employs the Kalman filter for optimizing the state estimation, where LSTM neural

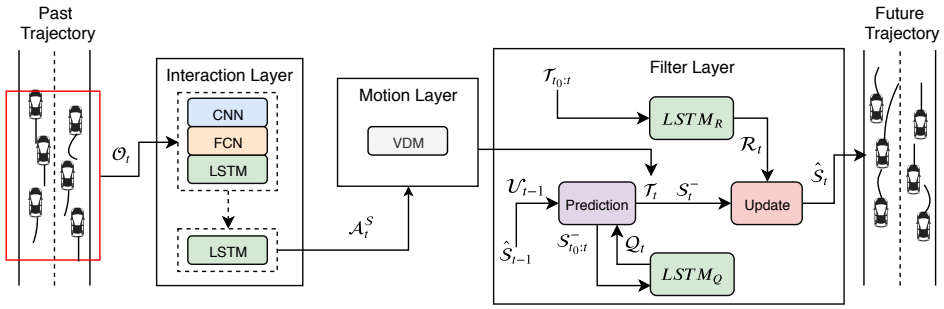

Fig. 2: Illustration of the **IaKNN** Model: In the diagram, at timestamp t, the environmental observation $\mathcal{O}_t$ flows into the interaction layer which generates the *interaction-aware acceleration* $\mathcal{A}_t^S$. Then, we calculate the *interaction-aware trajectory* of vehicles $\mathcal{T}_t$ w.r.t Vehicle Dynamic Model (VDM) in motion layer. In the end, time-varying multi-agent Kalman neural networks run over the predicted time horizon L to fuse dynamic trajectory $\mathcal{S}_t$ and *interaction-aware trajectory* $\mathcal{T}_t$. Particularly, the time-varying process and measurement noises in the filter layer are set by zero-mean Gaussian noises with covariance formulated in a gated-structure neural network.

networks are used for learning the time-varying process and measurement noises that are used in the Kalman model, and this is the first of its kind for trajectory prediction.

III. Traffic Datasets

To the best of our knowledge, there are four publicly available traffic datasets, namely Cityscapes [27], KITTI [28], ApolloScape [29], and NGSIM [21]. Cityscapes, KITTI and ApolloScape are collected from the first person perspective which have been widely adopted for single-agent systems in the field of intelligent vehicle society. NGSIM, is collected on the southbound US101 road and the eastbound I-80 road with a software application called NG-VIDEO which transcribes vehicles' trajectories from an overhead video. In this work, we only use NGSIM since among the 4 datasets, NGSIM is the only one that is suitable for a study in the multi-agent system which we target in this paper. Additionally, as reported in some existing studies [30], [31], noises vary significantly in NGSIM, and this is one of the motivations that we proposed to learn the time-varying covariances in the Kalman filter.

IV. Kalman Filter

In this part, we provide some background of the Kalman filter (KF) which is used as a building block in our model in this paper. KF is an optimal state estimator in the mean square error (MSE) sense with a linear (dynamic) model and Gaussian noise assumptions. Suppose the state, control, and observation of the linear model are s_t, u_t and z_t, respectively. The model could be expressed with a process equation and a measurement equation as follows.

$$\begin{aligned} s_t &= \mathcal{F} \cdot s_{t-1} + \mathcal{B} \cdot u_{t-1} + \omega, \\ z_t &= \mathcal{H} \cdot s_t + \eta, \end{aligned}$$

where $\mathcal{F}$ is a dynamic matrix, $\mathcal{B}$ is a control matrix, $\mathcal{H}$ is an observation matrix, which are all known. Moreover, $\omega \sim \mathcal{N}(0, \mathcal{Q})$ is the process noise and $\eta \sim \mathcal{N}(0, \mathcal{R})$ is the measurement noise based on the noise covariance matrices $\mathcal{Q}$ and $\mathcal{R}$, respectively.

The process of KF is as follows. It iterates between a prediction phase and an update phase for each of the observations. In the prediction phase, the current state s_t^- and the error covariance matrix $\mathcal{P}_t^-$ are estimated as follows.

$$\begin{aligned} s_t^- &= \mathcal{F} \cdot \hat{s}_{t-1} + \mathcal{B} \cdot u_{t-1}, \\ \mathcal{P}_t^- &= \mathcal{F} \cdot \hat{\mathcal{P}}_{t-1} \cdot \mathcal{F}^T + \mathcal{Q}. \end{aligned}$$

In the update phase, once the current observation z_t is received, the Kalman gain $\mathcal{K}_t$, the prior estimation $\hat{s}_t$ and the error covariance matrix $\hat{\mathcal{P}}_t$ are calculated as follows.

$$\begin{aligned} \mathcal{K}_t &= \mathcal{P}_t^- \cdot \mathcal{H}^T \cdot (\mathcal{H} \cdot \mathcal{P}_t^- \cdot \mathcal{H}^T + \mathcal{R})^{-1}, \\ \hat{s}_t &= s_t^- + \mathcal{K}_t \cdot (z_t - \mathcal{H} \cdot s_t^-), \\ \hat{\mathcal{P}}_t &= (I - \mathcal{K}_t \cdot \mathcal{H}) \cdot \mathcal{P}_t^-. \end{aligned}$$

For a comprehensive review of KF, the readers could refer to standard references [32].

KF is effective and commonly used as a basic data processing skill in autonomous industry where sophisticated KFs, for instance, Extended KF (EKF) and Particle Filtering (PF) are also adopted in the literature [33], [34]. In our work, we take a novel approach only adopting KF as the filtering part in neural-network based filtering algorithm. It is worth mention that, the filter we used is in some sense an advanced version of KF where the noise covariances are being learned online but not pre-set.

V. Problem Statement

We assume there are N vehicles in the multi-agent system (traffic scene). For each vehicle at timestamp t, we collect its position p_t, velocity v_t, acceleration a_t, vehicle width w_t, vehicle length l_t, and relative distances $\{d_t^j\}_{j=1}^{N-1}$ with other agents. We call the observations of all vehicles *environmental observation at timestamp* t denoted as o_t. Given the past h-length environmental observations $\mathcal{O}_t := \{o_{t-h+1}, o_{t-h+2}, \cdots, o_t\}$, we aim to predict the future L-length trajectories of each vehicle.

Note that the number of vehicles N is set to be 6 in our experiments because our setting is the two-lane dynamics where the ego-vehicle has one in front, one at the rear, and three vehicles in the neighboring lane. N is an independent variable in our methodology, and thus it is straightforward to increase N to consider more vehicles for trajectory prediction tasks.

TABLE I: Notations and meanings (at timestamp t).

Notation	Meaning
$\mathcal{O}_t$	Traffic Environment Observation
$\mathcal{A}_t^{\mathcal{S}}$	Interaction-aware Acceleration
$\mathcal{T}_t$	Interaction-aware Trajectory
$\mathcal{F}$	State Transition Matrix
$\mathcal{B}$	Control Matrix
$\mathcal{S}_t$	Dynamic Trajectory
$\mathcal{S}_t^-$	*Priori* Estimation of Dynamic Trajectory
$\hat{\mathcal{S}}_t$	*Posteriori* Estimation of Dynamic Trajectory
$\mathcal{U}_t$	Dynamic Acceleration
$\mathcal{Q}_t$	Process Noise Covariance
$\mathcal{R}_t$	Measurement Noise Covariance
G_t	Ground Truth of Future Trajectory
N	Number of Vehicles
L/L'	Observation/Prediction Time Horizon

VI. Methodology

In this section, we present our architecture *interaction-aware Kalman neural networks* (IaKNN). Figure 2 gives an overview of the architecture, where the notations are explained as follows. $\mathcal{A}^{\mathcal{S}}$ is the portfolio of interaction-aware accelerations outputted by the interaction layer. $\mathcal{T}$ is the portfolio of interaction-aware trajectories computed by the motion layer, and $\mathcal{S}$ and $\mathcal{V}$ are the state and the control of the Kalman filter in the filter layer, respectively, where $\mathcal{R}$ and $\mathcal{Q}$ are the noise covariance matrices, both learned by LSTM neural networks. Besides, in this paper, t_0, t, L' and L represent the starting time, current time, observation time horizon and prediction time horizon, respectively, where $t_0 \leq t \leq t_0 + L'$.

In the following, we present three layers of IaKNN, namely the **interaction layer**, the **motion layer** and the **filter layer**. The notations that are frequently used throughout the paper are given in Table I.

A. Interaction Layer

In the interaction layer, we aim to extract the *interaction-aware accelerations* $\mathcal{A}^{\mathcal{S}}$ from the past traffic environment observations $\mathcal{O}_t$.

1) Interaction-aware Acceleration: Normally, the motion of a vehicle would be determined by its own vehicle dynamics. Nevertheless, in a multi-agent system which we target in this paper, the situation is much more complex since drivers of vehicles would be affected by those of other vehicles that are nearby (or they would interact with one another). For example, a vehicle would be forced to slow down if another vehicle nearby tries to cut the lane in the front. In fact, the motion of vehicles is determined by not only their physical accelerations but also the interactive effects among vehicles. Inspired by the classical social force model [35], which models the intention of a driver to avoid colliding with dynamic or static obstacles, we propose to extract those accelerations such they capture both the raw accelerations recorded and the interactions among vehicles nearby. We call them the *interaction-aware accelerations* and denote them by $\mathcal{A}^{\mathcal{S}}$.

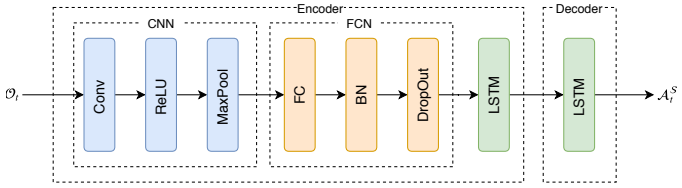

Fig. 3: Illustration of **Interaction Layer**

Specifically, at timestamp t, traffic environment observations $\mathcal{O}_t$ includes a sequence of recorded accelerations $a_{t_0:t}$, vehicle widths $w_{t_0:t}$, vehicle lengths $l_{t_0:t}$, and relative distances $d_{t_0:t}$ of agents in the system. By following [19], we compute the so-called *repulsive interaction forces* $e_{t_0:t} := \exp\big((v^i + v^j) \cdot \Delta t - d^{ij}\big)_{t_0:t}$, where superscripts i and j represent two vehicles that are close to each other and include them in $\mathcal{O}_t$. Thus, the interaction operator formula at timestamp t is written in details as,

$$\mathcal{A}_t^{\mathcal{S}} = \textbf{Interaction}_{\{\mathcal{W},b\}}\big(a_{t_0:t}, w_{t_0:t}, l_{t_0:t}, d_{t_0:t}, e_{t_0:t}\big).$$

The interaction layer is implemented as a neural network as presented in Figure 3. The architecture of interaction layer is an LSTM encoder-decoder. In the encoder, we build convolutional layers (CNN) regarded as a social tensor extractor, fully-connected layers (FCN) regarded as a mixer of the social features, and merge the deep features into the encoder LSTM. In the decoder, the decoder LSTM outputs the predicted accelerations. Note that we applied the batch normalization to LSTMs with the vertical connections, which are transported from one layer to another, and it is technologically similar to some existing studies [36], [37]. In addition, we introduce the DropOut (with the fraction of 0.5) in case of overfitting.

2) Operator Representation: At timestamp t, the interaction layer in an operator formula is written as,

$$\textbf{Interaction}_{\{\mathcal{W},b\}} : \mathcal{O}_t \longmapsto \mathcal{A}_t^{\mathcal{S}},$$

where $\mathcal{O}_t$ is a portfolio of past environmental observations from t_0 to t and $\mathcal{A}_t^{\mathcal{S}}$ is the portfolio of the interation-aware accelerations from $t+1$ to $t+L$.

B. Motion Layer

In the motion layer, we aim to calculate the *interaction-aware trajectories* $\mathcal{T}$ based on the *interaction-aware accelerations* $\mathcal{A}^{\mathcal{S}}$ from the interaction layer.

The main intuition of the motion layer comes from the primary kinematic equation which establishes a relationship among *position, time and velocity*. Our strategy is to use higher-order derivatives of a position for better forecasting. Specifically, let p_t be the position of a dynamic obstacle at timestamp t. We write p_t with the Taylor expansion as follows.

$$p_t = p_{t-1} + v_{t-1} \cdot \Delta t + \frac{1}{2} a_{t-1} \cdot \Delta t^2 + O\Big(\Delta t^3\Big) \qquad (1)$$

where v_{t-1} represents the velocity at timestamp $t-1$, a_{t-1} represents the acceleration at timestamp $t-1$, and the Big-O term captures all remaining terms which would be

ignored. Moreover, we replace the acceleration term a_t with an *interaction-aware acceleration* $\mathcal{A}^{\mathcal{S}}$ which is derived from the environment observations.

We specify the velocity term v in Equation 1 as follows. Suppose the current timestamp is t. For v_t, we take the velocity readings which are currently available and transform them to v_t by using a dynamic model - depending on the agent type, we adopt different dynamic models for this task, which shall be introduced shortly. For $v_{t+1}, v_{t+2}, ...$, we estimate their values by applying an integral function based on the interaction-aware accelerations as follows.

$$v_{t+i} := \int_{t}^{t+i} \mathcal{A}^{\mathcal{S}} dt,$$

where $i = 1, 2, ..., L$.

Next, we introduce the dynamic models, vehicle dynamic model (VDM), which map motion along the axes of the global reference frame to motion along the axe fo the robot's local reference frame. By following [38], we implement the vehicle dynamic model as a classical bicycle model [39]. Specifically, suppose $s := (x, y, \theta, v_x, v_y, r)$ is the current reading involving velocities, where x and y are the coordinates, θ is the orientation, v_x and v_y are velocities, and r is the yaw rate. The bicycle model is written as follows.

$$\begin{aligned}
\dot{x} &= v_x \cdot \cos\theta - v_y \cdot \sin\theta, \\
\dot{y} &= v_x \cdot \sin\theta + v_y \cdot \cos\theta, \\
\dot{\theta} &= r.
\end{aligned}$$

$\dot{x}$ and $\dot{y}$ are the transformed velocities along the x and y directions, respectively. For more details, the readers are referred to [38]. Note that the nonlinear VDM model is not being estimated in the Kalman filter, and it is used only to transform the velocity readings which are based on a vehicle-centric coordinating system to those based on a global coordinating system. That is, VDM model is simply a plug-in transformation function, and Kalman filter used in this paper is based on a linear system (on positions and velocities).

To simplify this layer, we regard all agents as mass points and their motion behaviors are described in the basic kinematic motion equations $\dot{x} = v_x$ and $\dot{y} = v_y$.

1) Operator Representation: At timestamp t, the motion layer in an operator formula is written as,

$$\textbf{Motion} : \mathcal{A}_t^{\mathcal{S}} \longmapsto \mathcal{T}_t,$$

where $\mathcal{T}_t$ is an interaction-aware trajectory from $t+1$ to $t+L$.

C. Filter Layer

In the filter layer, we establish a model based on the Kalman filter to estimate the dynamic trajectories $\mathcal{S}_t$ based on the *interaction-aware trajectories* $\mathcal{T}_t$ used as observations.

1) Filter Model: To fit the Kalman filter as described in Section IV, we let the dynamic trajectories $\mathcal{S}_t$ be the states, the *interaction-aware trajectories* $\mathcal{T}_t$ be the observations and the dynamic accelerations $\mathcal{U}_t$ be the controls in a linear model. As a result, the equation of the linear dynamic model could be written as follows.

$$\mathcal{S}_t = \mathcal{F} \cdot \mathcal{S}_{t-1} + \mathcal{B} \cdot \mathcal{U}_{t-1} + \omega_t, \tag{2}$$

$$\mathcal{T}_t = \mathcal{S}_t + \eta_t \tag{3}$$

where $\mathcal{F}$ is the state transition matrix, $\mathcal{B}$ is the control matrix, $\omega_t \sim \mathcal{N}\big(0, \mathcal{Q}_t\big)$ are the time-varying process noises and $\eta_t \sim \mathcal{N}\big(0, \mathcal{R}_t\big)$ are the measurement noises. The time-varying covariances $\mathcal{Q}_t$ and $\mathcal{R}_t$ will be learned by time-varying noise models which consist of LSTM neural networks and will be introduced later. Note that here we assume the observation matrix $\mathcal{H}$ is an identity matrix for simplicity.

2) Specifications of the Layer: We assume N agents (dynamic obstacles) in the multi-agent system. At timestamp t, the state $\mathcal{S}_t$ and the observation $\mathcal{T}_t$ of our Equation 2 and 3 could be written as follows.

$$\mathcal{S}_t := \begin{bmatrix} \mathcal{S}_t^1 \\ \vdots \\ \mathcal{S}_t^N \end{bmatrix}_{(2\cdot N\cdot L)\times 1} \quad \text{and} \quad \mathcal{T}_t := \begin{bmatrix} \mathcal{T}_t^1 \\ \vdots \\ \mathcal{T}_t^N \end{bmatrix}_{(2\cdot N\cdot L)\times 1},$$

where the state $\mathcal{S}_t^i$ includes positions p_k^i from GPS and velocities v_k^i from the wheel odometer and the observation $\mathcal{T}_t^i$ includes the predicted positions $\bar{p}_k^i$ and the predicted velocities $\bar{v}_k^i$, where $t+1 \le k \le t+L$. Specifically, we have the following.

$$\mathcal{S}_t^i := \begin{bmatrix} p_{t+1}^i \\ v_{t+1}^i \\ \vdots \\ p_{t+L}^i \\ v_{t+L}^i \end{bmatrix}_{(2\cdot L)\times 1} \quad \text{and} \quad \mathcal{T}_t^i := \begin{bmatrix} \bar{p}_{t+1}^i \\ \bar{v}_{t+1}^i \\ \vdots \\ \bar{p}_{t+L}^i \\ \bar{v}_{t+L}^i \end{bmatrix}_{(2\cdot L)\times 1},$$

where the subscript L is the predicted time horizon.

Next, we define the state transition matrix $\mathcal{F}$ and the control matrix $\mathcal{B}$ in our model. Firstly, we define two matrix blocks M_1 and M_2 as follows.

$$M_1 := \begin{bmatrix} 1 & \Delta t & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \Delta t \\ & & & & 1 \end{bmatrix}_{(2\cdot L)\times(2\cdot L)}$$

and

$$M_2 := \begin{bmatrix} \frac{1}{2}\Delta t^2 & & & \\ \Delta t & & & \\ & \ddots & & \\ & & \frac{1}{2}\Delta t^2 & \\ & & \Delta t & \end{bmatrix}_{(2\cdot L)\times L}$$

where Δt is the time difference between two adjacent traffic environment observations. Then, $\mathcal{F}$ and $\mathcal{B}$ are block diagonal matrices that are defined as follows.

$$\mathcal{F} := diag\big(\underbrace{M_1, \ldots, M_1}_{N}\big), \text{ and } \mathcal{B} := diag\big(\underbrace{M_2, \ldots, M_2}_{N}\big).$$

3) Prediction and Updated Steps: The prediction step of the Kalman filter is defined as,

$$\mathcal{S}_t^- = \mathcal{F} \cdot \hat{\mathcal{S}}_{t-1} + \mathcal{B} \cdot \mathcal{U}_{t-1},$$
$$\mathcal{P}_t^- = \mathcal{F} \cdot \hat{\mathcal{P}}_{t-1} \cdot \mathcal{F}^T + \mathcal{Q}_t,$$

and the update step is as,

$$\mathcal{K}_t = \mathcal{P}_t^- \cdot (\mathcal{P}_t^- + \mathcal{R}_t)^{-1},$$
$$\hat{\mathcal{S}}_t = \mathcal{S}_t^- + \mathcal{K}_t \cdot (\mathcal{T}_t - \mathcal{S}_t^-),$$
$$\hat{\mathcal{P}}_t = (\mathcal{I} - \mathcal{K}_t) \cdot \mathcal{P}_t^-,$$

where $\mathcal{Q}_t$ and $\mathcal{R}_t$ are the outputs of the time-varying noise models that we introduce next.

4) Time-varying Noise Model: Since our desired filter model is time-varying, we assume both the process noises and the measurement noises to follow a zero-mean Gaussian noise model with its covariances formulated as $\mathcal{Q}_t := LSTM_Q(\mathcal{S}_{t_0:t}^-)$ and $\mathcal{R}_t := LSTM_R(\mathcal{T}_{t_0:t})$, respectively. Here, we adopted LSTM because the noises are generated sequentially with trajectories and LSTM is known for its capability of capturing sequential dynamics [40]. The concrete $\mathcal{Q}_t$ and $\mathcal{R}_t$ are tractable via backpropagation under the end-to-end trainable architecture. Besides, $\mathcal{Q}_t$ and $\mathcal{R}_t$ are not PSD in general, but in practice, it is possible to implement a LQ decomposition to guarantee the PSD property of $\mathcal{Q}_t$ and $\mathcal{R}_t$ as some existing studies do [24].

5) Operator Representation: At timestamp t, the filter layer in an operator formula is written as,

$$\textbf{Filter}_{\{\mathcal{W},b\}} : \{\hat{\mathcal{S}}_{t-1}, \mathcal{T}_t, \mathcal{U}_{t-1}\} \longmapsto \hat{\mathcal{S}}_t,$$

where $\hat{\mathcal{S}}_t$ is the *Posteriori* estimation of dynamic trajectory from $t+1$ to $t+L$.

D. Loss Function

The loss function of IaKNN (**Interaction**$_{\{\mathcal{W},b\}}$, **Motion**, and **Filter**$_{\{\mathcal{W},b\}}$) is defined as the sum of displacement error of *Posteriori* estimation $\hat{\mathcal{S}}$ of dynamic trajectories and ground truth G over all time steps and agents, as follows.

$$\mathcal{L}_{\{\mathcal{W},b\}} := \frac{1}{(L'+1)\cdot N} \cdot \sum_{i=1}^{N} \sum_{t=t_0}^{t_0+L'} ||\hat{\mathcal{S}}_t^i - G_t^i||^2,$$

where G_t^i means the ground truth of the future trajectory of i-th agent at the start timestamp t. Noth that L' is the observation time horizon as defined in the above.

VII. Experiments

A. Dataset

We evaluate our approach IaKNN on two public datasets, namely US Highway 101 (US-101) and Interstate 80 (I-80) of the NGSIM program. Each dataset contains (x, y)-coordinates of vehicle trajectories in a real highway traffic with 10Hz sampling frequency over a 45-min time span. The 45-min dataset consists of three 15-min segments of mild, moderate and congested traffic conditions. We follow the experimental settings that were proposed by existing studies [14], [41] and combine US-101 and I-80 into one dataset. As a result, the dataset involves 100,000 frames of raw data.

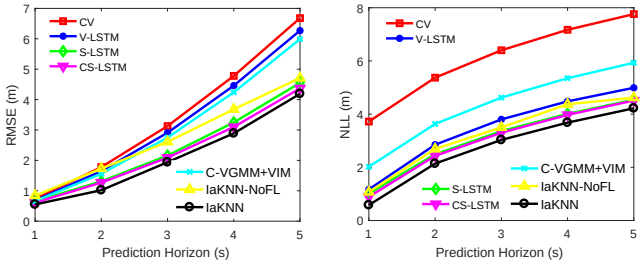

Fig. 4: Illustration of RMSE and NLL of model *CV, V-LSTM, S-LSTM, C-VGMM+VIM, CS-LSTM, IaKNN-NoFL, and IaKNN*. For both evaluation metrics, we plot its average for the prediction time horizon in 5s.

We construct the multi-agent training traffic scene in the following construction procedure. Firstly, we align the raw data by its timestamps. Secondly, we form a multi-agent traffic scene by picking one host vehicle and including five closest vehicles on its traffic lane or two adjacent traffic lanes. Finally, we set the window size for extraction as 7 seconds to generate the training scenes. We will explain the setting of window size in the next section.

B. Baselines

The following baseline models will be compared with our model IaKNN.

- *Constant Velocity* (*CV*): Model of the primary kinematic equation with constant velocity.
- *Vanilla-LSTM* (*V-LSTM*): Model of Seq2Seq. It is from a sequence of past trajectories to a sequence of future trajectories [42].
- *Social LSTM* (*S-LSTM*): Model of LSTM-based neural network with a *social pooling* for pedestrian trajectory prediction. As demonstrated in [12], the model performs consistently better than traditional models such as the linear model, the collision avoidance model and the social force model. Therefore, we do not compare these traditional methods in our experiments.
- *C-VGMM+VIM*: One variational Gaussian mixture models with Markov random fields, taken into account the interaction effects between agents [43].
- *Convolutional Social Pooling-LSTM* (*CS-LSTM*): Maneuver based motion model which will generate a multi-modal predictive distribution [14].
- *IaKNN-NoFL*: The proposed method IaKNN without the filter layer.

Note that the model *GAIL-GRU* is not taken into account in the comparison, because it has access to the future ground-truth trajectories of neighboring vehicles to predict the ego-vehicle's trajectory, while the others have not [15].

C. Evaluation Metrics

Two metrics, namely the *root-mean-square error* (RMSE) and *negative log-likelihood* (NLL), are used to measure the effectiveness of IaKNN. In particular, the first 2-seconds trajectories and the rest 5-seconds trajectories are used as

past trajectories and the ground truth in a 7-seconds [1] multi-agent training traffic scene, respectively.

- RMSE: the root mean squared sum accumulated by the displacement errors over the predicted positions and real positions during the prediction time horizon.
- NLL: the sum of the negative log probabilities of the predicated positions against the ground-truth positions during the prediction time horizon (we consider a predicted position to be correct if its distance from the ground-truth one is bounded by a small threshold and wrong otherwise).

D. Implementation Details

The default length of the past trajectories is two seconds and the time horizon of the predicted trajectories is one to five seconds. The default number of hidden units in LSTMs in the interaction layer and filter layer is set to 32 and all LSTM weight matrices are initialized using a uniform distribution over $[-0.001, 0.001]$. The weight matrices for other layers are set with the Xavier initialization. The biases are initialized to zeros. Additionally, in the training process, we adopt the Adam stochastic gradient descent with hyper-parameters $\beta_1 = 0.9$, $\beta_2 = 0.99$ and set the initial learning rate to be 0.001. In order to avoid the gradient vanishing, a maximum gradient norm constraint is set to 5. For the parameters of baselines, we follow the original settings in the open source code. The experiments are conducted on a machine with Intel(R) Xeon(R) CPU E5-1620 and one NVIDIA GeForce GTX 1070 GPU.

E. Result Analysis

The performance results of baselines and our method on the traffic scene are shown in Figure 4. We compute the RMSE and NLL for all traffic scenes and plot the average for the prediction time horizon in 5s. The naive *CV* produces the largest prediction errors in all comparison methods. *V-LSTM, S-LSTM* and *CS-LSTM* perform similarly in terms of both metrics which is mainly because they are all LSTM-based neural networks. Additionally, *S-LSTM, CS-LSTM*, and *IaKNN-NoFL* perform better than *V-LSTM* and *C-VGMM+VIM*, especially in RMSE, and this is mainly because the formulation establishes on multi-agent system setting, which also takes into account the interactive effects among vehicles in modeling. IaKNN outperforms slightly all baselines in terms of both metrics. Specifically, we observe that it outperforms the best baseline *CS-LSTM* with about 10% improvement. This may be explained by the fact that the filter layer in our IaKNN model estimates the underlying trajectories from both interaction-aware trajectories $\mathcal{T}$ and dynamic trajectories $\mathcal{S}$ in a traffic scene and the interaction layer has done a good job in capturing the interactive effects among the surrounding vehicles. The combination of the deep neural network and probabilistic filter makes our model more applicable for the real-time trajectory prediction in the traffic scene.

[1]The window size is equal to the sum of the prediction horizon and the observation. We set the prediction horizon to be 5 seconds by following the work of CS-LSTM [14]. Regarding the observation horizon, existing algorithms used different settings as follows: Social LSTM (3.2 seconds), CS-LSTM (3 seconds), Trafficpredict (2 seconds). We set the observation horizon to be 2 seconds, resulting in a window size of 7 seconds.

F. Case Studies

We illustrate the results by showing the predicted 2-second trajectories of vehicles by IaKNN and the real ones in the two lane-change traffic scenarios in Figure 5. The results demonstrate the effectiveness of the prediction by IaKNN intuitively. We observe that the predicted trajectories (color blue) are close to the real ones (color green) for all vehicles on the lanes in the Figure 5.

To prevent the traffic accidents, vehicles keep a safe following distance from others. The predicted trajectories by IaKNN as shown Figure 5 confirm this statement, which is one of the clear superiorities that the multi-agent model outperforms the single-agent model in the traffic trajectory prediction scene. Specifically, we observe predicted trajectories of the front vehicles by IaKNN in Figure 5 have clear intentions to change their lanes, one way to increase the safety following distance, in the future seconds. Changing lane is one of the crucial issues that an autonomous vehicle need to take actions to manage the traffic risks before it happens. Hence, IaKNN is a more sensitive traffic prediction algorithm than the past algorithms in autonomous vehicle industry based on the comprehensive modeling in an interaction-aware multi-agent environment.

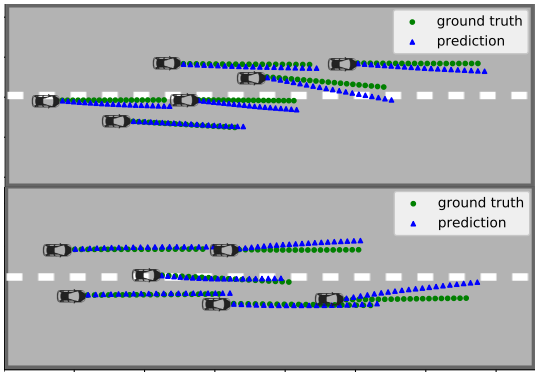

Fig. 5: Case studies of the prediction result by IaKNN: The predicted trajectories and the real ones are drawn in blue and green color, respectively. For each vehicle, we plot its future 2s trajectory.

VIII. Conclusion and Future Work

In this study, we propose an architecture, IaKNN, to predict the motion of surrounding vehicles in a dynamic environment, in which we make the first attempt to generate an intractable quantity from complex traffic scene yielding a new interaction-aware motion model. Extensive experiments show that IaKNN outperforms the baseline models in terms of effectiveness on the NGSIM dataset. Further work will be carried out to extend IaKNN to a probabilistic formulation and combine IaKNN with a maneuver-based model in which road topology and the traffic information are taken into account.

REFERENCES

[1] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[2] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[3] Junqing Wei, Jarrod M Snider, Tianyu Gu, John M Dolan, and Bakhtiar Litkouhi. A behavioral planning framework for autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 458–464. IEEE, 2014.

[4] Tianyu Gu, Jarrod Snider, John M Dolan, and Jin-woo Lee. Focused trajectory planning for autonomous on-road driving. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 547–552. IEEE, 2013.

[5] Matthew McNaughton. Parallel algorithms for real-time motion planning. 2011.

[6] Matthew McNaughton, Chris Urmson, John M Dolan, and Jin-Woo Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *2011 IEEE International Conference on Robotics and Automation*, pages 4889–4895. IEEE, 2011.

[7] Wenda Xu, Junqing Wei, John M Dolan, Huijing Zhao, and Hongbin Zha. A real-time motion planner with trajectory optimization for autonomous vehicles. In *2012 IEEE International Conference on Robotics and Automation*, pages 2061–2067. IEEE, 2012.

[8] Haoyang Fan, Fan Zhu, Changchun Liu, Liangliang Zhang, Li Zhuang, Dong Li, Weicheng Zhu, Jiangtao Hu, Hongye Li, and Qi Kong. Baidu apollo em motion planner. *arXiv preprint arXiv:1807.08048*, 2018.

[9] C Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1071–1081. ACM, 2005.

[10] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE transactions on robotics*, 21(6):1077–1091, 2005.

[11] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal*, 1(1):1, 2014.

[12] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[13] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number CONF, 2018.

[14] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. *arXiv preprint arXiv:1805.06771*, 2018.

[15] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 204–211. IEEE, 2017.

[16] Raunak P Bhattacharyya, Derek J Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J Kochenderfer. Multi-agent imitation learning for driving simulation. *arXiv preprint arXiv:1803.01044*, 2018.

[17] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. *arXiv preprint arXiv:1811.02146*, 2018.

[18] Ce Ju, Zheng Wang, and Xiaoyu Zhang. Socially aware kalman neural networks for trajectory prediction. *arXiv preprint arXiv:1809.05408*, 2018.

[19] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] James Colyar and John Halkias. Us highway 101 dataset. *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

[22] Timothy D Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[23] Huseyin Coskun, Felix Achilles, Robert S DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In *ICCV*, pages 5525–5533, 2017.

[24] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop kf: Learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems*, pages 4376–4384, 2016.

[25] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[26] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019.

[27] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[28] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[29] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 954–960, 2018.

[30] Christian Thiemann, Martin Treiber, and Arne Kesting. Estimating acceleration and lane-changing dynamics from next generation simulation trajectory data. *Transportation Research Record*, 2088(1):90–101, 2008.

[31] Mizanur Rahman, Mashrur Chowdhury, and Jerome McClendon. Real-time traffic data prediction with basic safety messages using kalman-filter based noise reduction model and long short-term memory neural network. *arXiv preprint arXiv:1811.03562*, 2018.

[32] Gary Bishop, Greg Welch, et al. An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, 8(27599-3175):59, 2001.

[33] John L Crassidis, F Landis Markley, and Yang Cheng. Survey of nonlinear attitude estimation methods. *Journal of guidance, control, and dynamics*, 30(1):12–28, 2007.

[34] Josep Aulinas, Yvan R Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. *CCIA*, 184(1):363–371, 2008.

[35] Dirk Helbing, Illés Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487, 2000.

[36] César Laurent, Gabriel Pereyra, Philémon Brakel, Ying Zhang, and Yoshua Bengio. Batch normalized recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2657–2661. IEEE, 2016.

[37] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[38] Romain Pepy, Alain Lambert, and Hugues Mounier. Path planning using a dynamic vehicle model. In *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, volume 1, pages 781–786. IEEE, 2006.

[39] Saied Taheri. An investigation and design of slip control braking systems integrated with four-wheel steering. 1992.

[40] Zheng Wang, Cheng Long, Gao Cong, and Ce Ju. Effective and efficient sports play retrieval with deep representation learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 499–509. ACM, 2019.

[41] Nachiket Deo and Mohan M Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1184. IEEE, 2018.

[42] SeongHyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. *arXiv preprint arXiv:1802.06338*, 2018.

[43] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. *IEEE Transactions on Intelligent Vehicles*, 3(2):129–140, 2018.