Using Graph-Theoretic Machine Learning to Predict Human Driver Behavior

Rohan Chandra, Aniket Bera, and Dinesh Manocha

Abstract—Studies have shown that autonomous vehicles (AVs) behave conservatively in a traffic environment composed of human drivers and do not adapt to local conditions and socio-cultural norms. It is known that socially aware AVs can be designed if there exists a mechanism to understand the behaviors of human drivers. We present an approach that leverages machine learning to predict, the behaviors of human drivers. This is similar to how humans implicitly interpret the behaviors of drivers on the road, by only observing the trajectories of their vehicles. We use graph-theoretic tools to extract driver behavior features from the trajectories and machine learning to obtain a computational mapping between the extracted trajectory of a vehicle in traffic and the driver behaviors. Compared to prior approaches in this domain, we prove that our method is robust, general, and extendable to broad-ranging applications such as autonomous navigation. We evaluate our approach on real-world traffic datasets captured in the U.S., India, China, and Singapore, as well as in simulation.

I. Introduction

Autonomous Vehicles (AVs) are an active area of research, successfully employing tools from machine learning [27], perception [30], planning and driver behavior modeling [52]. Recently, there have been multiple breakthroughs in perception-based tasks in autonomous driving in areas that include object detection [28], tracking [6], [10], trajectory prediction [11], [7], [12], and planning [14], [40]. While these advances have been widely successful, current AVs still lack the ability to interact with multiple human drivers in dense traffic scenarios [13] such as intersections and merging on highways.

As a precautionary measure, AVs are designed to behave conservatively in order to maximize safety [53]. A recent study [55] conducted real-world AV experiments and collected factors that may associate with how people's opinions change before and after experiencing a ride in an AV. However, conservative AV behavior is not always desirable or necessary, particularly given potential consequences like low efficiency and shortsighted behavior, which frequently frustrate other human drivers [58]. For example, in [58], a Tesla driver is observed to be executing a lane-change maneuver. The Tesla AutoPilot slows down to wait for an excessively large gap in the target lane thereby blocking the traffic behind it in the current lane. This causes frustration and inefficiency among the blocked drivers. Furthermore, some studies [53] have pointed out that aggressive driving for AVs is even desirable in certain situations like reconnaissance, material transport, emergency

handling, or efficiency-sensitive application. Sometimes it is even desirable simply based on human preference.

There is prior work on predicting human driver behavior from trajectory data using machine learnings [47], [52], [9], [8], [5]. The two main approaches for this task include inverse reinforcement learning (IRL) and machine learning (regression and clustering techniques). Due to the data-driven nature of these methods, they incur two predominant limitations, which are inherent to most learning-based techniques in artificial intelligence research. First, these data-driven methods are constrained to a narrow range of traffic environments and fail to generalize to different environments. Furthermore, it has been shown both empirically and theoretically [4] that data-driven methods are not robust to fluctuations or noise in the sensor measurements (GPS, lidars, depth cameras etc.). These limitations prevent the current driver behavior prediction systems from being used in other AD tasks such as navigation.

Furthermore, in autonomous driving, it is important to handle the unpredictability and aggressive nature of human drivers during navigation. While there is considerable research on designing navigation algorithms [51], much of it assumes little to no interaction with human drivers. However, in real-life circumstances, drivers may act irrationally by moving in front of other vehicles, suddenly changing lanes, or aggressively overtaking. One such instance occurred in 2016 when an AV by Google collided with an oncoming bus during a lane change maneuver [19]. The AV assumed that the bus driver was going to yield; instead, the bus driver accelerated. Therefore, we need navigation methods that can account for different driver behaviors.

Main Contributions: In light of the limitations presented by data-driven methodologies, we present a fundamentally different approach to driver behavior prediction that can *generalize* to widely varying traffic scenarios while also being *robust* to realistic fluctuations in sensor noise. Our model not only alleviates the problems of prior approaches in order to predict human driver behavior, but also extends existing navigation research to behaviorally-guided navigation. Our main contributions include:

1) A new approach to predict driver behavior from raw vehicle trajectories using graph-theoretic machine learning. In this approach, called StylePredict, we use the concept of vertex centrality functions [48] and spectral analysis to measure the likelihood and intensity of driving styles such as overspeeding, overtaking, sudden lane-changes, etc. This process generates driver behavior features that can then be used for training machine learning algorithms.

2) Extending current navigation research [51] to *behaviorally-guided* local navigation. This novel

approach to navigation computes a local trajectory for the AV, taking into account the aggressiveness or conservativeness of human drivers. For example, the AV learns to slow down around aggressive human drivers while confidently overtaking conservative drivers.

StylePredict can be deployed in real-world traffic. We test extensively on real-world traffic datasets (Section V-B, Table III) collected in India, Singapore, U.S.A, and China. These datasets contain sensor noise (latency, precision, presence of outliers, etc.) identical to that expected in the real world. In Section IV-E and Table II, we demonstrate robustness of our method to these sensor issues.

II. RELATED WORK

A. Graph-based Machine Learning

Graph-based machine learning is a sub-field in machine learning where the input data is organized as graphs. While the core learning algorithms themselves, including neural networks, LSTMs [29], and convolutional neural networks, remain the same, they are now referred to as graph neural networks (GNNs) [62], Graph-LSTMs [12], and graph convolutional networks (GCNs) [63], respectively. Graph-based machine learning algorithms have been widely used in trajectory prediction, computer vision and natural language processing [64]. GNNs, Graph-LSTMs, and GCNs, however, are "deep" networks and require a huge amount of training data in order to produce meaningful results.

In this work, we instead use "shallow" graph-based machine learning, which includes learning algorithms based on logistic regression [44], multi-layer perceptrons, and support vector machines [18], which require fewer computational resources than deep learning-based methods.

B. Data-Driven Methods for Driver Behavior Prediction

Data-driven methods broadly follow two approaches. In the first approach, various machine learning algorithms including clustering, regression, and classification are used to predict or classify the driver behavior as either aggressive or conservative. These methods have mostly been studied in traffic psychology and the social sciences [46], [54], [61]. So far, there has been relatively little work to improve the robustness and ability to generalize to different traffic scenarios, which require ideas from computer vision and robotics. In this work, we bridge the gap between robotics, computer vision, and the social sciences and develop an improved graph-theoretic machine learning model for human driver behavior prediction that alleviates the limitations of prior approaches.

The second approach uses trajectory data to learn reward functions for human behavior using inverse reinforcement learning (IRL) [47], [52], [49]. IRL-based methods, however, have certain limitations. IRL requires large amounts of training data and the learned reward functions are unrealistically tailored towards scenarios only observed in the training data [47], [49]. For instance, the approach proposed in [47] requires 32 million data samples for optimum performance. Additionally, IRL-based methods are sensitive to noise in the trajectory data [52], [49]. Consequently, current IRL-based methods are restricted to simple and sparse traffic conditions.

C. Navigation Research in Autonomous Driving

Navigation in robotics is a well studied area of research. At a broad level, navigation methods can be categorized into approaches for vehicle control, motion planning, and end-to-end learning-based methods. Techniques for vehicular control methods assume apriori an accurate motion model of the vehicle. Such methods can be used for controlling vehicles at high speeds or during complex maneuvers. Motion planning methods can be further sub-divided into lattice-based [25], probabilistic search-based [32], or use non-linear control optimization [37] approaches.

In addition to vehicular control and motion planning methods, many learning-based techniques are also used [23], [24], [22]. These methods are based on reinforcement learning where one finds an optimal policy that directly maps the sensor measurements to control commands such as velocity or acceleration and steering angle. Li et al. [35] formulate the navigation problem as one of action prediction using the proximity relationship between agents along with their visual features.

However, the above methods do not consider the interaction among human drivers. Typically, in order to model dynamic obstacles, prior methods have either assumed a linear constant velocity model [22]. Our behavior-based formulation can be integrated with these methods. We refer the reader to [51] for a detailed review on recent planning and navigation methods.

D. Interpretation of Driver Behavior in Social Science

Many studies have attempted to define driver behavior for traffic-agents. Sagberg et al. [50] extract and summarize the common elements from these definitions and propose a unified definition for driver behavior. We incorporate this definition in our driver behavior model.

Definition II.1. *(Sagberg et al. [50] Driver behavior refers to the high-level "global behavior", such as aggressive or conservative driving. Each global behavior can be expressed as a combination of one or more underlying "specific styles". For example, an aggressive driver (global behavior) may frequently overspeed or overtake (specific styles).*

The main benefit of Sagberg's definition is that it allows for a formal taxonomy for driver behavior classification. Specific indicators can be classified as either *longitudinal* styles (along the axis of the road) or *lateral* (perpendicular to the axis of the road). We can formally characterize driver behavior by mathematically modeling the underlying specific indicators.

Problem II.1. *In a traffic video with N vehicles during any time-period Δt, given the trajectories of all vehicles, our objective is to mathematically model the specific styles for all drivers during Δt.*

In Section IV-B, we elucidate on "mathematically modeling" a specific style. In Section III, we construct the "traffic-graph" data structure used by our approach. We introduce the ideas of vertex centrality in Section IV-A followed by a presentation of our main approach in Section IV-B. We describe the experiments and results in Section V. We conclude the paper in Section VII.

III. Representing Traffic Data Using Graphs

The behavior of drivers depend on their interactions with nearby drivers. StylePredict models the relative interactions between drivers by representing traffic through weighted undirected graphs called "traffic-graphs". In this section, we describe the construction of these graph representations. If we assume that the trajectories of all the vehicles in the video are extracted using state-of-the-art localization methods [2] and are provided to our algorithm as an input, then the traffic-graph, $\mathcal{G}_t$, at each time-step t can be defined as follows,

Definition III.1. *A "traffic-graph", $\mathcal{G}_t$, is a dynamic, undirected, and weighted graph with a set of vertices $\mathcal{V}(t)$ and a set of edges $\mathcal{E}(t) \subseteq \mathcal{V}(t) \times \mathcal{V}(t)$ as functions of time defined in the 2-D Euclidean metric space with metric function $f(x,y) = \|x - y\|^2$. Two vertices $v_i, v_j \in \mathcal{V}$ are connected if and only if $f(v_i, v_j) < \mu$, where μ is a distance threshold parameter.*

We use N to represent the maximum number of vehicles tolerated by our system. N is typically fixed as some large integer (e.g., 1000) for each vehicle. Most real world commercial and academic systems use large high-performing computers to run computations involving large matrices [17]. Therefore, large values of N do not impose a computational burden on our approach. Road-agents at each time instance t in a traffic scenario can be represented using a traffic-graph $\mathcal{G}_t$. Each vertex in the graph $\mathcal{G}_t$ is represented by the vehicle position in the global coordinate frame, *i.e.* $v_i \leftarrow [x_i, y_i]^\top \in \mathbb{R}^2$. The spatial distance between two vehicles is assigned as the cost of the edge connecting the two vehicles.

In computational graph theory, every graph $\mathcal{G}$ can be equivalently represented by an adjacency matrix, A. For a particular traffic-graph $\mathcal{G}$, the adjacency matrix A is given by $A(i,j) = (v_i, v_j)$ if $f(v_i, v_j) < \mu, i \neq j$ (otherwise 0). Adjacency matrices allow linear vector operations to be performed on graph structures, which are useful for analyzing individual vertices. For example, each non-zero entry in the jth column corresponding to the ith row of the adjacency matrix stores the relative distance between the ith and j^{th} vehicles. A is initialized as an $N \times N$ identity matrix.

However, considering the traffic-graph and its corresponding adjacency matrix only at a current time-step t is not useful in describing the behavior of a driver. The behavior of a driver also depends on their actions from previous time-steps. To accommodate this notion, at each time-step t, we populate A with principle sub-matrices A_t of size $t \times t$,

$$
\underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} \mathbf{A_1} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}_{\mathbf{A_2}} & \begin{matrix} \mathbf{a_{13}} \\ \mathbf{a_{23}} \end{matrix} \\ \begin{matrix} \mathbf{a_{31}} & \mathbf{a_{32}} \end{matrix} & \mathbf{1} \end{bmatrix}}_{\mathbf{A_3}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}}_{\mathbf{A_{N \times N}}} .
$$

The sub-matrix for the next time-step, A_{t+1}, is obtained by the following update,

$$
A_{t+1} = \overbrace{\left[\begin{array}{c|c} [A_t]_{\mathbf{t \times t}} & 0 \\ \hline 0 & 1 \end{array} \right]}^{\mathbf{(t+1) \times (t+1)}} + \delta\sigma(\delta)^\top, \tag{1}
$$

where $\delta\sigma(\delta)^\top \in \mathbb{R}^{(t+1) \times (t+1)}$ is a sparse update matrix and $\delta, \sigma(\delta)$ are update vectors defined as follows,

$$
\delta = \begin{bmatrix} \overbrace{\delta_{11}}^{\delta_{11} \neq \mathbf{0}} & 0 \\ \delta_{21} & 0 \\ \vdots & \vdots \\ \delta_{t1} & 0 \\ 0 & 1 \end{bmatrix}_{\mathbf{(t+1) \times 2}} \qquad \sigma(\delta) = \begin{bmatrix} 0 & \overbrace{\delta_{11}}^{\delta_{11} \neq \mathbf{0}} \\ 0 & \delta_{21} \\ \vdots & \vdots \\ 0 & \delta_{t1} \\ 1 & 0 \end{bmatrix}_{\mathbf{(t+1) \times 2}} .
$$

Here, σ is a permutation that swaps the two columns in δ. If the jth row of δ is non-zero, then that implies that the jth road-agent has formed a new edge with a new vehicle that came into its proximity; this new vehicle will be added to the current traffic-graph. This new vehicle is identified by a unique ID number provided by the localization sensor (GPS or lidar). For example, if a vehicle with ID $1(j = 1)$ has formed a new edge connection with another vehicle. This corresponds to $\delta_{11} \neq 0$. The update rule in Equation 1 ensures that a vehicle adds edge connections to new vehicles while retaining edge connections with previously seen vehicles.

A candidate vehicle is categorized as "new" with respect to a vehicle if there does not exist any prior edge connection between the vehicles *and* the speed of the old vehicle is greater than the candidate vehicle. If an edge connection already exists between the vehicle, then the candidate vehicle is said to have been "observed" or "seen". The dimension of A is constant $(N \times N)$. Once the upper limit N has been achieved (N different vehicles have been observed), then A is re-initialized as an $N \times N$ identity matrix. As the behaviors of each vehicle are determined in an online manner, "erasing" the old vehicles from the matrix to make way for new vehicles does not affect their behavior computation; their behaviors have already been computed and stored. If the number of vehicles is less than N, then the "unused" entries in A are simply left as 0. Finally, vehicles appearing and disappearing from the field of view of the ego-vehicle does not impact the size of A. If a vehicle does not remain in the field of view of the ego-vehicle for a sufficient amount of time, then our algorithm does not consider that vehicle in the adjacency matrix A.

IV. StylePredict: Mapping Trajectories to Behavior

A. Centrality Measures

In graph theory and network analysis, centrality measures [48] are real-valued functions $\zeta : \mathcal{V} \longrightarrow \mathbb{R}$, where $\mathcal{V}$ denotes the set of vertices and $\mathbb{R}$ denotes a scalar real number that identifies key vertices within a graph network. So far, centrality functions have been restricted to identifying influential personalities in online social media networks [42]
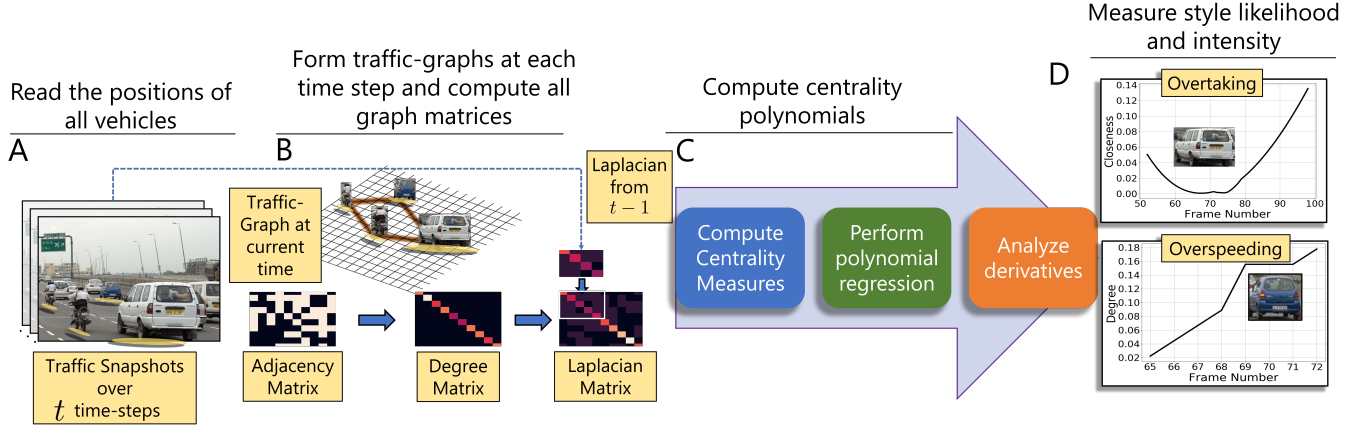
Figure 1: **Overview:** The autonomous vehicle reads the positions of all vehicles in realtime. The positions and corresponding spatial distances between vehicles are represented through a traffic-graph $\mathcal{G}_t$ (Section III). We use the centrality functions defined in Section IV-A to model the specific driving style corresponding to the global behaviors as outlined in Table I.

Table I: Definition and categorization of driving behaviors [50]. We measure the likelihood and intensity of specific styles by analyzing the first-and second-order derivatives of the centrality polynomials.

Global	Specific	Centrality	SLE	SIE
Aggressive	Overspeeding	Degree (ζ_d)	|1st Derivative|	|2nd Derivative|
	Overtaking / SLC	Closeness (ζ_c)	|1st Derivative|	|2nd Derivative|
	Weaving	Closeness (ζ_c)	Extreme Points	ε-sharpness
Conservative	Driving Slowly	Degree (ζ_d)	|1st Derivative|	|2nd Derivative|
	No Lane-change	Closeness (ζ_c)	|1st Derivative|	|2nd Derivative|

and key infrastructure nodes in the Internet [33], to rank web-pages in search engines [43], and to discover the origin of epidemics [57]. There are several types of centrality functions. The ones that are of particular importance to us are the degree centrality and the closeness centrality denoted as $\zeta_d(t)$ and $\zeta_c(t)$, respectively. These centrality measures are defined in [8] (See section III-C). Each function measures a different property of a vertex. Typically, the choice of selecting a centrality function depends on the current application at hand. In this work, the closeness centrality and the degree centrality functions measure the likelihood and intensity of specific driving styles such as overspeeding, overtaking, sudden lane-changes, and weaving [8].

B. Algorithm

Here, we present the main algorithm, called *StylePredict*, for solving Problem II.1. StylePredict maps vehicle trajectories to specific styles by computing the likelihood and intensity of the latter using the definitions of the centrality functions. The specific styles are then used to assign global behaviors [50] according to Table I. We summarize the StylePredict algorithm as follows:

1) Obtain the positions of all vehicles using localization sensors deployed on the autonomous vehicle and form traffic-graphs at each time-step (Section III).
2) Compute the closeness and degree centrality function values for each vehicle at every time-step.
3) Perform polynomial regression to generate uni-variate polynomials of the centralities as a function of time.

4) Measure likelihood and intensity of a specific style for each vehicle by analyzing the first- and second-order derivatives of their centrality polynomials.
5) Classify the centrality polynomials, obtained from step 3, as either aggressive or conservative using machine learning algorithms such as Multi-Layer Perceptrons (MLPs).

We depict the overall approach in Figure 1. We begin by using the construction described in Section III to form the traffic-graphs for each frame and use the definitions in [8] to compute the discrete-valued centrality measures. Since centrality measures are discrete functions, we perform polynomial regression using regularized Ordinary Least Squares (OLS) solvers to transform the two centrality functions into continuous polynomials, $\zeta_c(t)$ and $\zeta_d(t)$, as a function of time. We describe polynomial regression in detail in the following subsections. We compute the likelihood and intensity of specific styles by analyzing the first- and second-order derivatives of $\zeta_c(t)$ and $\zeta_d(t)$ (this step is discussed in further detail in Section IV-D).
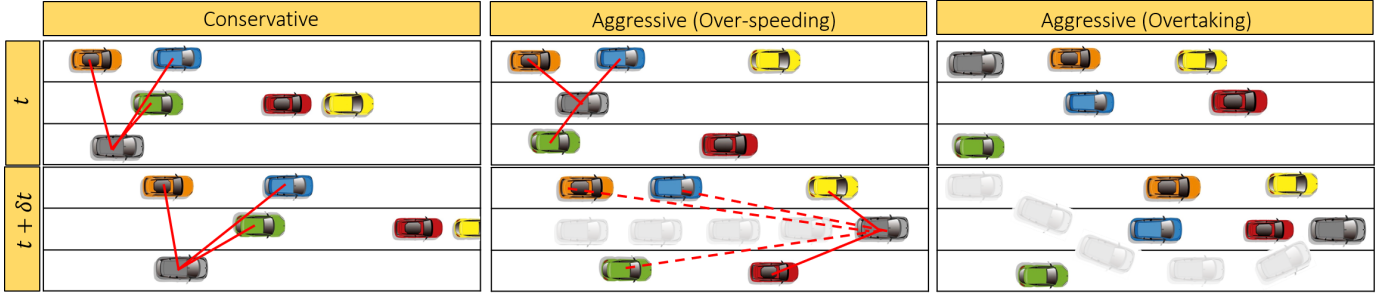
C. Polynomial Regression

In order to study the behavior of the centrality functions with respect to how they change with time, we convert the discrete-valued $\zeta[t]$ into continuous-valued polynomials $\zeta(t)$, using which we calculate the first- and second-order derivatives of the centrality functions as explained in Section IV-D.

In this work, we choose a quadratic[1] centrality polynomial can be expressed as $\zeta(t) = \beta_0 + \beta_1 t + \beta_2 t^2$, as a function of time. Here, $\beta = [\beta_0 \ \beta_1 \ \beta_2]^\top$ are the polynomial coefficients. These coefficients can be computed using ordinary least squares (OLS) equation as follows,
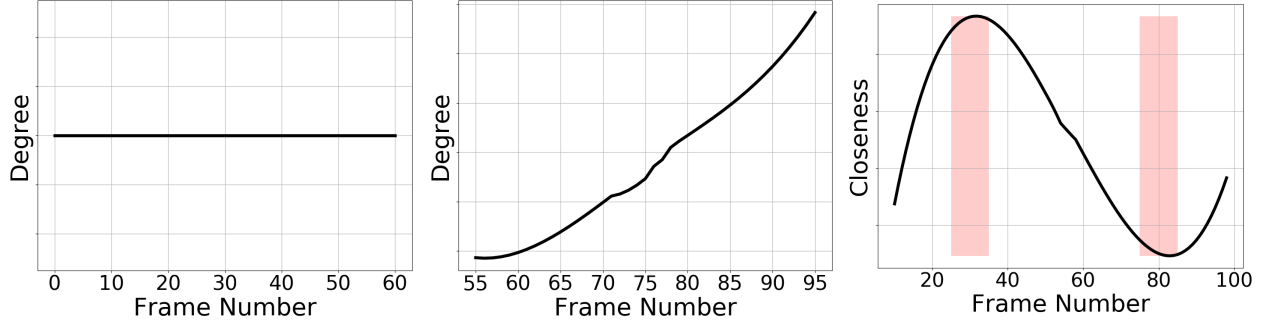
$$\beta = (M^\top M)^{-1} M^\top \zeta^i \qquad (2)$$

Here, $M \in \mathbb{R}^{T \times (d+1)}$ is the Vandermonde matrix [39]. and is given by,

[1]A polynomial with degree 2.

(a) In all three scenarios, the ego-vehicle is a gray vehicle marked with a blue outline. *(left)* A conservative vehicle, *(middle)* an overspeeding vehicle in the same lane, and *(right)* a weaving and overtaking vehicle.

(b) Constant degree centrality function for conservative vehicle.

(c) Monotonically increasing centrality function for overspeeding vehicle.

(d) Extreme points for closeness centrality function for weaving vehicle.

Figure 2: **Measuring the Likelihood of Specific Styles:** We measure (degree and closeness centrality) the likelihood that an ego-vehicle (grey with a blue outline) has a specific driving style by computing the magnitude of the derivative of the centrality functions as well as the functions' extreme points. In Figure 2b, the derivative of the degree centrality function is 0 because the ego-vehicle does not observe any additional new neighbors (See Section IV-D), so the degree centrality is a constant function; therefore, the vehicle is conservative. In Figure 2c, the vehicle overspeeds and, consequently, the rate of observing new neighbors is high, which is reflected in the magnitude of the derivative of the degree centrality being positive. Finally, in Figure 2d, the ego-vehicle demonstrates overtaking/sudden lane-changes and weaves through traffic. These behaviors are reflected in the magnitude of the slope and the location of extreme points, respectively, of the closeness centrality function.

$$M = \begin{bmatrix} 1 & t_1 & t_1^2 & \dots & t_1^d \\ 1 & t_2 & t_2^2 & \dots & t_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_T & t_T^2 & \dots & t_T^d \end{bmatrix}$$

D. Style Likelihood and Intensity Estimates

In the previous sections, we used polynomial regression on the centrality functions to compute centrality polynomials. In this section, we analyze and discuss the first and second derivatives of the degree centrality, $\zeta_d(t)$, and closeness centrality, $\zeta_c(t)$, polynomials. Based on this analysis, which may vary for each specific style, we compute the Style Likelihood Estimate (SLE) and Style Intensity Estimate (SIE) [8], which are used to measure the probability and the intensity of a specific style.

a) Overtaking/Sudden Lane-Changes: Overtaking is when one vehicle drives past another vehicle in the same or an adjacent lane, but in the same direction. The closeness centrality increases as the vehicle navigates towards the center and vice-versa. The SLE of overtaking can be computed by measuring the first derivative of the closeness centrality polynomial using $\mathrm{SLE}(t) = \left| \frac{\partial \zeta_c(t)}{\partial t} \right|$. The maximum likelihood $\mathrm{SLE}_{\max}$ can be computed as $\mathrm{SLE}_{\max} = \max_{t \in \Delta t} \mathrm{SLE}(t)$. The SIE of overtaking is computed by simply measuring the second

derivative of the closeness centrality using $\mathrm{SIE}(t) = \left| \frac{\partial^2 \zeta_c(t)}{\partial t^2} \right|$. Sudden lane-changes follow a similar maneuver to overtaking and therefore can be modeled using the same equations used to model overtaking.

b) Overspeeding: The degree centrality can be used to model overspeeding. As A_t is formed by adding rows and columns to A_{t-1} (See Equation 1), the degree of the ith vehicle (denoted as θ_i) is calculated by simply counting the number of non-zero entries in the ith row of A_t. Intuitively, a drivers that are overspeeding will observe new neighbors along the way (increasing degree) at a higher rate than conservative, or even neutral, drivers. Let the rate of increase of θ_i be denoted as θ_i'. By definition of the degree centrality and construction of A_t, the degree centrality for an aggressively overspeeding vehicle will monotonically increase. Conversely, the degree centrality for a conservative vehicle driving at a uniform speed or braking often at unconventional spots such as green light intersections will be relatively flat. Therefore, the likelihood of overspeeding can be measured by computing,

$$\mathrm{SLE}(t) = \left| \frac{\partial \zeta_d(t)}{\partial t} \right|$$

Similar to overtaking, the maximum likelihood estimate is given by $\mathrm{SLE}_{\max} = \max_{t \in \Delta t} \mathrm{SLE}(t)$. Figures 2b and 2c visualize how the degree centrality can distinguish between an

overspeeding vehicle and a vehicle driving at a uniform speed.

c) Weaving: A vehicle is said to be weaving when it "zig-zags" through traffic. Weaving is characterized by oscillation in the closeness centrality values between low values towards the sides of the road and high values in the center. Mathematically, weaving is more likely to occur near the critical points (points at which the function has a local minimum or maximum) of the closeness centrality polynomial. The critical points t_c belong to the set $\mathcal{T} = \left\{ t_c \big| \frac{\partial \zeta_c(t_c)}{\partial t} = 0 \right\}$. Note that $\mathcal{T}$ also includes time-instances corresponding to the domain of constant functions that characterize conservative behavior. We disregard these points by restricting the set membership of $\mathcal{T}$ to only include those points t_c whose $\varepsilon-$sharpness [20] of the closeness centrality is non-zero. The set $\mathcal{T}$ is reformulated as follows,

$$\mathcal{T} = \left\{ t_c \bigg| \frac{\partial \zeta_c(t_c)}{\partial t} = 0 \right\}$$
$$\text{s.t.} \max_{t \in \mathcal{B}_\varepsilon(t_c)} \frac{\partial \zeta_c(t)}{\partial t} \neq \frac{\partial \zeta_c(t_c)}{\partial t} \quad (3)$$

where $\mathcal{B}_\varepsilon(y) \in \mathbb{R}^d$ is the unit ball centered around a point y with radius ε. The SLE of a weaving vehicle is represented by $|\mathcal{T}|$, which represents the number of elements in $\mathcal{T}$. The SIE(t) is computed by measuring the $\varepsilon-$sharpness value of each $t_c \in \mathcal{T}$. Figure 2d visualizes how the degree centrality can distinguish between an overspeeding vehicle and a vehicle driving at a uniform speed.

d) Conservative Vehicles: Conservative vehicles, on the other hand, are not inclined towards aggressive maneuvers such as sudden lane-changes, overspeeding, or weaving. Rather, they tend to stick to a single lane [1] as much as possible, and drive at a uniform speed [50] below the speed limit. Correspondingly, the values of the closeness and degree centrality functions in the case of conservative vehicles remain constant. Mathematically, the first derivative of constant polynomials is 0. The SLE of conservative behavior is therefore observed to be approximately equal to 0. Additionally, the likelihood that a vehicle drives uniformly in a single lane during time-period Δt is higher when,

$$\left| \frac{\partial \zeta_c(t)}{\partial t} \right| \approx 0 \text{ and } \max_{t \in \mathcal{B}_\varepsilon(t^*)} \text{SLE}(t) \approx \text{SLE}(t_c).$$

The intensity of such maneuvers will be low and is reflected in the lower values for the SIE.

E. Robustness to Noise

In the formulation above, our algorithm assumes perfect sensor measurements of the global coordinates of all vehicles. However, in real-world systems, even state-of-the-art methods for vehicle localization incur some measurement errors. We consider the case in which the raw sensor data is corrupted by some noise ϵ. Without loss of generality, we prove robustness to noise for the degree centrality. Further, the analysis can be extended to other centrality functions. The discrete-valued centrality vector for the ith agent is given by $\zeta^i \in \mathbb{R}^{T \times 1}$. Therefore, $\zeta^1[2]$ corresponds to the degree centrality value of the 1st agent at $t = 2$.

In the previous section, we showed that a noiseless estimator may be obtained by solving an ordinary least squares (OLS) system given by Equation 2. However, in the presence of noise ϵ, the OLS system described in Equation 2 is modified as,

$$\tilde{\beta} = (M^\top M)^{-1} M^\top \tilde{\zeta}^i \quad (4)$$

where $\tilde{\zeta}^i = \zeta^i + \epsilon$. Then we can prove that $\|\tilde{\beta} - \beta\| = \mathcal{O}(\epsilon)$. We defer the proof to the supplementary material.

F. Behavior Classification Using Machine Learning

We treat the centrality polynomials computed in Section IV-C as features in a supervised learning paradigm. While our formulation is such that any classification algorithm can be used, we select Multi-Layer Perceptron (MLP) as the classification model due to its superior performance. We defer a comparison between different ML algorithms to Section V.

Formally, let Φ denote the MLP model that takes in a centrality feature vector, $\zeta(t)$, as input and produces a 1-hot vector encoding, $\hat{y} = \Phi(\zeta(t))$, of the behavior prediction as output. Let y denote the corresponding ground-truth label for that agent. Then, for N agents, a loss function can be framed as follows,

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \|y_i - \Phi(\zeta^i(t))\|^2 \quad (5)$$

where θ denote the MLP model parameters. The goal of the classification problem is to find the optimum values of θ, say θ^*, that minimizes Equation 5. More simply,

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$$

V. EXPERIMENTS AND RESULTS

We begin with a discussion of the evaluation metrics, the Time Deviation Error (TDE) and the weighted classification accuracy, for validating and measuring the accuracy of behavior prediction methods in Section V-A. Then, we describe the real-world traffic datasets and simulation environment used for testing our approach and outline the annotation algorithm used to generate ground-truth labels for aggressive and conservative vehicles in Section V-B. We use the TDE to validate our approach and analyze the results in real-world traffic datasets in Section V-C. Finally, we analyze the weighted accuracy of StylePredict and compare with state-of-the-art graph classification and behavior prediction methods in Section V-D.

A. Evaluation Metrics

1) Time Deviation Error (TDE) [8]: We use the TDE to validate our approach to modeling driver behavior using StylePredict. The TDE measures the temporal difference between the moments when a human identifies a behavior and when that same behavior is modeled using StylePredict. The TDE is given by the following equation,

$$\text{TDE}_{\text{style}} = \left| \frac{t_{\text{SLE}} - \mathbb{E}[T]}{f} \right| \quad (6)$$

where $\mathbb{E}$ denotes the expected time-stamp of an exhibited behavior in the ground-truth annotated by a human and f is the frame rate of the video. t_{SLE} is obtained using $\arg\max_{t\in\Delta t} \text{SLE}(t)$ as explained in Section IV-B, $\mathbb{E}[T]$ is computed using Algorithm 1, described in the following section.

2) Weighted Classification Accuracy: To measure the accuracy of StylePredict in predicting future behaviors, we report a weighted classification accuracy, which is defined as the fraction of correctly predicted behaviors, weighted by class frequencies .

B. Datasets and Simulation Environment

Our testing environments consist of both simulation and real-world trajectory data. The simulation includes a top-view of the traffic while the trajectory data has been captured from front-view vehicle-based sensors. Both settings are the same in the sense that they provide the same type of information – the coordinates of each vehicle with respect to a fixed frame of reference (camera center in the case of top-view or the ego-vehicle in the case of front view).

a) Simulation Environment: We use the Highway-Env simulator [34] developed using PyGame. The simulator consists of a 2D environment where vehicles are made to drive along a multi-lane highway using the Bicycle Kinematic Model [45] as the underlying motion model where the linear acceleration model is based on the Intelligent Driver Model (IDM) [59] and the lane changing behavior is based on the MOBIL [31] model. We note here that more sophisticated car models such as the Ackermann steering model may be used. While many popular vehicle simulators [21], [38] do provide the option of Ackermann modeling, these simulators do not provide the behavior-rich environment needed for testing our algorithm. Therefore, we restrict ourselves to [34] that can generate aggressive and conservative driver behaviors. Furthermore, most simulators that use Ackermann steering do so for modeling safety by preventing slipping of the tires during tight turns such as U-turns or intersection turns. Since our environment consists of a straight road with no turns, the Ackermann model holds little advantage over the Bicycle kinematic model in our case.

b) Real-World Datasets: We have evaluated StylePredict on traffic data collected from geographically diverse regions of the world. In particular, we use data collected in Pittsburgh (U.S.A) [15], New Delhi (India) [7], Beijing (China) [60], and Singapore (private dataset). The format of the data includes the timestamp, road-agent I.D., road-agent type, and spatial coordinates obtained via GPS or lidars. We understand that the characteristics of drivers in a particular city may not mirror those in other cities of the same country. Therefore, all results presented in this work correspond to the traffic in the specific city where the dataset is recorded.

One of the main issues with these datasets is that they do not contain labels for aggressive and conservative driving behaviors. Therefore, we obtain ground-truth driver behavior annotations using Algorithm 1. We directly use the raw trajectory data from these datasets without any pre-processing or filtering step.

Algorithm 1: Computing $\mathbb{E}[T]$ for each video in a dataset.

Input : M participants, set of starting frames
$S = \{s_1, s_2, \ldots, s_M\}$, set of ending frames
$E = \{e_1, e_2, \ldots, e_M\}$
Output: $\mathbb{E}[T]$ for a video
1 $s^* = \min S$
2 $e^* = \max E$
3 Initialize a counter $c_t = 0$ for each frame $t \in [s^*, e^*]$
4 **for** $t \in [s^*, e^*]$ **do**
5   **if** $t \in [s_m, e_m]$ **then**
6          $c_t \leftarrow c_t + 1$
7   **end**
8      $\mathcal{P}(T = t) = c_t$
9 **end**
10 $\mathbb{E}[T] = \sum_t t c_t, \; t = s^*, s^* + 1, \ldots, e^*$

Table II: **Analysing Simulation Results using TDE:** We analyze StylePredict by varying the traffic density, number of lanes, and the noise parameter ϵ (Equation 4). We observe that TDE increases as these parameters increase in value. We discuss these results in detail in Section V-C.

Density	TDE	Robustness	TDE	# Lanes	TDE
$N = 5$	0.08s	$\epsilon = 10^{-4}$	0.001s	$L = 2$	0.10s
$N = 13$	0.15s	$\epsilon = 10^{-3}$	0.001s	$L = 4$	0.27s
$N = 20$	0.56s	$\epsilon = 10^{-2}$	0.013s	$L = 6$	0.53s
$N = 25$	0.79s	$\epsilon = 10^{-1}$	0.050s	$L = 8$	0.98s

For each video, the final ground-truth annotation (or label) is the expected value of the frame at which the ego-vehicle is most likely to be executing an aggressive driving style. This is denoted as $\mathbb{E}[T]$. The goal for any driver behavior prediction model should be to predict the aggressive style at a time stamp as close to $\mathbb{E}[T]$ as possible. The implied difference between the two time stamps is measured by the TDE metric.

The TDE metric is computed by Equation 6. Here, $t_{\text{SLE}} = \arg\max_{t\in\Delta t} \text{SLE}(t)$, as explained in Section IV-B. We use Algorithm 1 for computing $\mathbb{E}[T]$. We recruited $M = 35$ participants with driving experience in at least two countries out of USA, Singapore, China, and India. This ensured that participants are "expert" annotators we are able to obtain gold-standard labels. For each video, every participant was asked to mark the starting and end frames for the time-period during which a vehicle is observed executing an aggressive maneuver. Participants were asked to watch out for typical traits such as overspeeding, overtaking, sudden lane-changes, weaving, driving slowly in single lanes etc. Once the start and end frames are recorded, we proceed by using Algorithm 1 as explained in Section V-B(b). Participants were allowed to scrub back and forth during a video and replay any moment any number of times. Furthermore, participants were allowed to zoom into a video to inspect styles more closely. We ignore repetitions and did not observe any control errors.

For each video, we end up with $S = \{s_1, s_2, \ldots, s_M\}$ and $E = \{e_1, e_2, \ldots, e_M\}$ start and end frames, respectively. We extract the overall start and end frame by finding the minimum

Table III: We report the Time Deviation Error (TDE) (in seconds (s)) for the following driving styles: Overspeeding (OS), Overtaking (OT), Sudden Lane-Changes (SLC), and Weaving (W) along with their % appearance in various real-world datasets. On average, we find that it is easiest to predict weaving and sudden lane-changes in India. This observation agrees with our cultural analysis in Section V-C

Dataset	Styles							
	OS		OT		SLC		W	
	TDE	%	TDE	%	TDE	%	TDE	%
U.S. [15]	0.25s	83	0.67s	2	0.23s	14	0.26s	1
Singapore	0.54s	27	0.88s	27	1.21s	27	1.28s	18
China [60]	0.74s	24	0.44s	32	0.39s	36	0.23s	8
India [7]	0.81s	16	0.38s	40	0.19s	28	0.06s	16

Table IV: We compare the weighted classification accuracy of StylePredict versus supervised learning-based SOTA methods on the Argoverse dataset [15]. Additionally, we compare the accuracy of different supervised learning machine learning and deep learning algorithms.

Dataset	Method	Weighted Accuracy
Argoverse	DANE [36]	65.50%
	Cheung et al. [16]	62.50%
	StylePredict *w. LR*	69.90%
	StylePredict *w. RNN*	70.80%
	StylePredict *w. SVM*	75.00%
	StylePredict *w. MLP*	**89.90%**

and maximum value in S and E, respectively (lines $1-2$). We denote these values as s^* and e^*. Next, we initialize a distinct counter, c_t, for each frame $t \in [s^*, e^*]$ (line 3). We increment a counter c_t by 1 if $t \in [s_m, e_m]$ (lines $4-7$). The value of the counter c_t is assigned to $\mathcal{P}(T)$ (line 8). The $\mathbb{E}[T]$ of $\mathcal{P}(T)$ can then be computed using the standard definition of expectation of a discrete probability mass function (line 10). Algorithm 1 is applied separately for each video in each dataset.

C. Validating StylePredict Using TDE

In Table II, we report the average TDE in seconds (s) in simulation environments. We used Algorithm 1 to compute the TDE in various simulation settings. First, we varied the traffic density by increasing the number of vehicles from 5 to 25. As the number of vehicles grows, the TDE increases, which is to be expected since it is harder for a human participant to spot different styles in denser traffic resulting in a detection delay and therefore higher TDE. Next, we analyzed the robustness property by varying the noise parameter ϵ (Equation 4). We opted for $\epsilon = \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ as this range reflects the most common values of error that may occur in nature. TDE for values lower than 10^{-4} all converged to 0. We naturally observe that the TDE increases with more noise. Finally, we varied the number of lanes from $2-8$ (1 lane is invalid as lateral styles cannot be observed in a single lane) and observe that the TDE increases with the number of lanes. This is because, with more space available, it is unclear to human participants whether a particular maneuver is aggressive or neutral. On the other hand, overtaking and lane-changing in a $2-$lane highway is very evident and easy to spot, resulting in a lower TDE.

In Table III, we report the average TDE in seconds (s) in different geographical regions and cultures for the following driving styles: Overspeeding (OS), Overtaking (OT), Sudden Lane-Changes (SLC), and Weaving (W). The traffic conditions differ significantly due to the varying cultural norms in different countries such as Singapore, the United States (U.S.), China, and India. For instance, traffic is more regulated in the U.S. than in Asian countries such as India or China, where vehicles do not conform to standard rules such as lane-driving. Such differences contribute to different driving behaviors. Our quantitative results in Table III and qualitative results in Figure 3 show that our driver behavior modeling algorithm is not affected by cultural norms. Across all cultures, the average TDE is less

than 1 second for every specific style. Aggressive vehicles are still associated with high centrality values, while conservative vehicles remain associated with low centrality values.

In Figure 3, we show traffic recorded in Singapore *(top row)*, the U.S. *(second row)*, China *(third row)*, and India *(bottom row)*. In each scenario, the first three columns depict the trajectory of a vehicle executing a specific style between some time intervals. The last column shows the corresponding centrality plot. The shaded colored regions overlaid on the graphs in Figures 3d and 3p are color heat maps that correspond to $\mathcal{P}(T)$ (line 8, Algorithm 1). The orange dashed line indicates the mean time frame, $\mathbb{E}[T]$, and the blue dashed line indicates t_{SLE}. The main result can be observed by noting the negligible distance between the two dashed lines, *i.e.* the TDE.

In the first row (corresponding to traffic in Singapore), for instance, our approach accurately predicts a maximum likelihood of a sudden lane-change by the white sedan at around the 75th frame (blue dashed line, Figure 3d), with an average TDE of 0.88 seconds. Similarly, in the second row (corresponding to traffic in the U.S.), we precisely predict the maximum likelihood of the vehicle overspeeding by the vehicle denoted by the red dot at around the 30th frame with a TDE of 0.25 seconds. Note that in both cases the TDE (the distance between the blue and orange dashed vertical lines) is < 1 second.

D. Analyzing Behavior Prediction Using Weighted Accuracy

We compare our approach with Dynamic Attributed Network Embedding (DANE) [36] and Cheung et al. [16]. Both baselines predict human behavior but differ in their techniques. DANE also uses a graph-based approach (although not based on centrality) where the main step consists of computing the spectrum of the Laplacian matrix. Cheung et al., on the other hand, use linear lasso regression on trajectory features, extracted from raw traffic videos. We present a comparison using the weighted accuracy with DANE and Cheung et al. in Table IV where we show an improvement of up to 25%.

StylePredict uses a multi-layer perceptron (MLP) [26] for the classification task. However, other classifiers in the machine learning literature such as logistic regression (LR), support vector machines (SVM)), and deep neural networks (RNNs) may be used. In Table IV, we compare the results of replacing the MLP with different classifiers and benchmark the different output accuracies against the MLP.
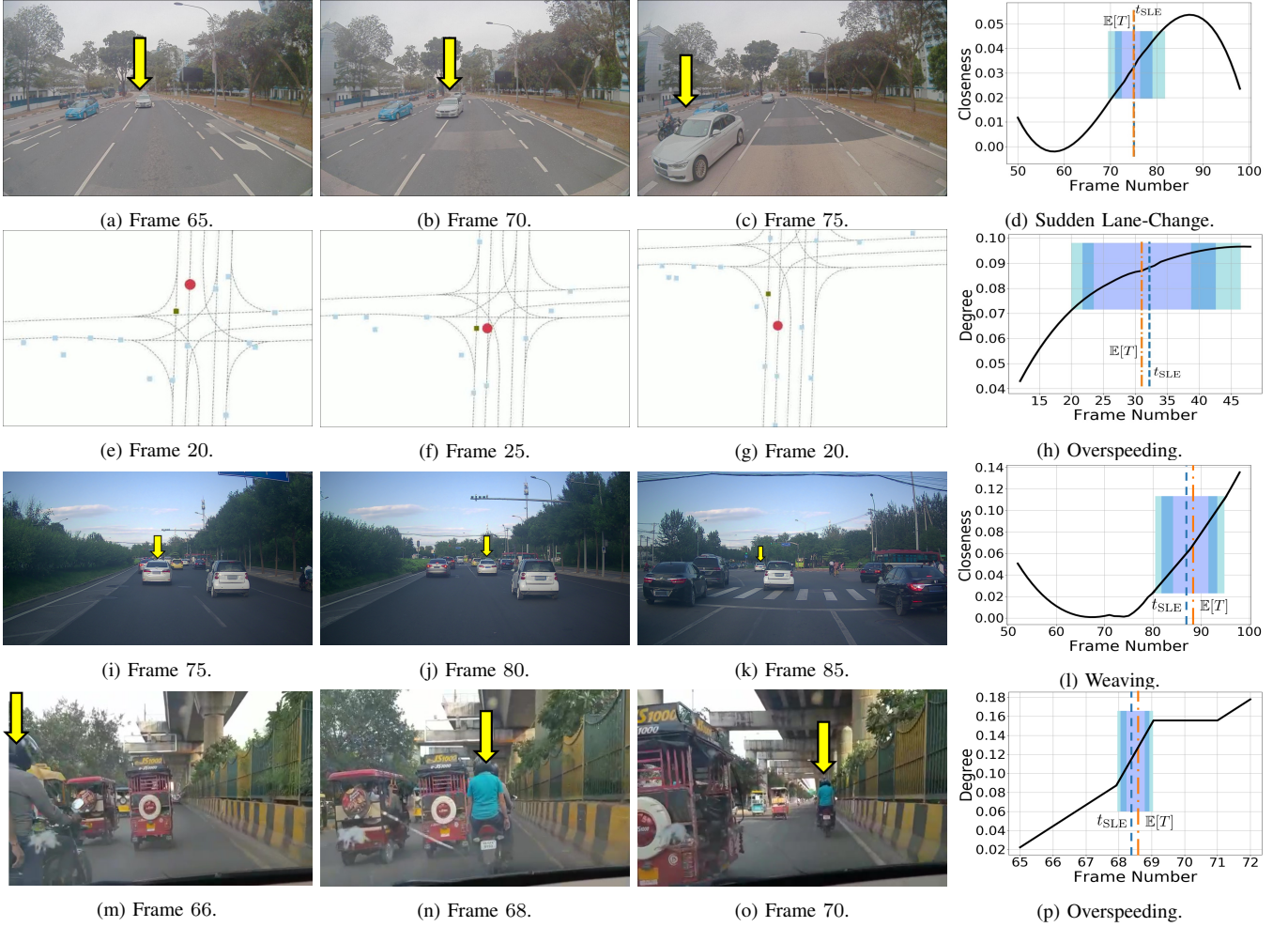
Figure 3: **Driver Behavior Modeling in Singapore** *(top row)*, **U.S.** *(second row)*, **China** *(third row)*, **and India** *(bottom row)*: In each row, the first three figures demonstrate the trajectory of a vehicle executing an aggressive driving style (sudden lane change, overspeeding, weaving, and overspeeding, respectively), while the fourth figure shows the corresponding closeness or degree centrality plot. The shaded colored regions overlaid on the graphs in the first two rows are color heat maps that correspond to $\mathcal{P}(T)$ (line 8, Algorithm 1).

VI. BEHAVIORALLY-GUIDED NAVIGATION

In autonomous navigation, it is important to handle the unpredictable and/or aggressive nature of human drivers (See Section I). In this section, we show that, unlike existing navigation methods [51], StylePredict can be used to train a behaviorally-guided navigation policy that takes into account the conservative or aggressive nature of human drivers. We begin by generating behaviorally-guided trajectories for vehicles in existing traffic simulators [34] using StylePredict for training a reinforcement learning (RL) navigation policy in Section VI-A. This is followed by a discussion of the RL model used to obtain the behaviorally-guided navigation policy in Section VI-B. Finally, in Section VI-C, we show that such a behaviorally-guided navigation policy allows an AV to perform more efficient lane changes and adapt its speed according to the behavior of the vehicles around it.

A. Augmenting Traffic Simulators with Driver Behavior

Existing traffic simulators [34], [38] assume a fixed motion model that does not take into account the nature of other drivers. Such simulators inevitably produce navigation policies that are not behaviorally-guided. Therefore, we use the StylePredict algorithm to augment such traffic simulators with driver behavior information such that they generate behaviorally-guided trajectories. The behavior of a vehicle in the Highway-Env simulator [34] is controlled using a set of parameters described in the supplementary material. To simulate aggressive and conservative behaviors, we find the appropriate range for these parameters by iteratively performing the following steps:

1) We initialize the simulator parameters as random values.
2) We use the parameters to simulate the trajectories for each vehicle. By varying the parameters, the vehicles perform longitudinal and lateral maneuvers with certain likelihoods and intensities.
3) We use StylePredict to measure these likelihoods (SLE) and intensities (SIE) following the framework introduced in Section IV. These measures indicate the aggressiveness or conservativeness of a driver. For instance, a high likelihood and intensity of lane changing and overspeeding would indicate an aggressive driver. Based on this feedback, we update the parameters for the next episode.
4) We repeat steps 2 and 3 until the likelihoods and inten-

(a) AV approaches aggressive vehicle.

(b) AV slows down, but does not overtake .

(c) AV resumes acceleration.

(d) AV approaches conservative vehicle.

(e) AV confidently switches to adjacent lane.

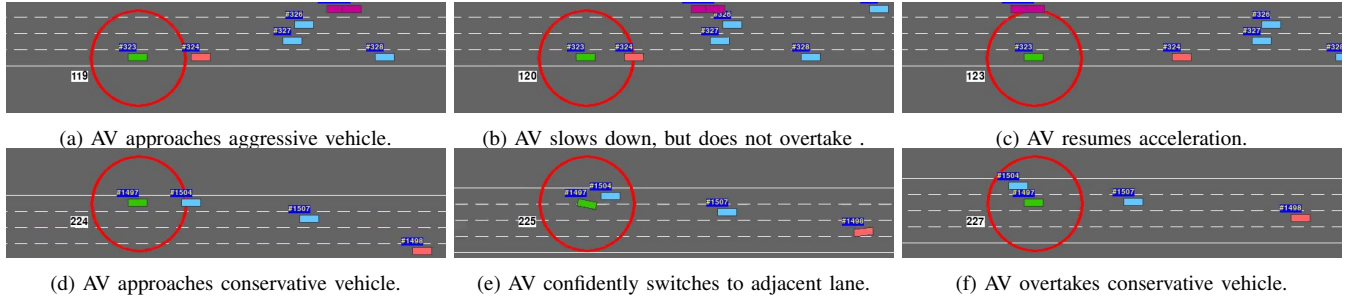(f) AV overtakes conservative vehicle.

Figure 4: **Behavior-Guided Navigation:** While interacting with aggressive (*top*) and conservative (*bottom*) vehicles. We indicate the AV, aggressive, and conservative vehicles in green, red, and blue, respectively. (*Top*) The AV senses that the red vehicle is aggressive and therefore decides to slow down instead of overtaking. (*Bottom*) The AV notices the conservative vehicle in front and decides to confidently overtake it.

sities of maneuvers match those of a desired behavior (aggressive or conservative).

The final set of parameter values corresponding to aggressive and conservative behaviors is used to instantiate aggressive and conservative vehicles at runtime, shown as red and blue vehicles in Figure 3. Our enhanced traffic simulator results in trajectories with varying levels of aggressiveness, in terms of maneuvers like overspeeding, overtaking, and so on.

B. Training the Behaviorally-Guided Navigation Policy

We use RL to learn a behaviorally-guided navigation policy offline using the enhanced simulator with behaviorally-augmented trajectories, and deploy it at runtime. We formulate the RL model as a Markov Decision Process (MDP) and use deep Q-learning [41] to train the navigation policy. We refer the reader to [40] (Sections IV and V) for further details on the RL model.

We frame the navigation problem as a Markov Decision Process (MDP) represented by $\mathcal{M} := \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}\}$. The sets of possible states and actions are denoted by $\mathcal{S}$ and $\mathcal{A}$, respectively. $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ captures the state transition dynamics, γ is the discounting factor, and $\mathcal{R}$ is the set of rewards defined for all of the states in the environment. The state of the world S at any time step is equal to a matrix $F \times V$, which includes the state s of every vehicle in the environment. V is the number of vehicles considered, and F is the number of features used to represent the state of a vehicle. The environment consists of a highway road with four single-direction lanes, along with conservative and aggressive traffic agents that are generated using StylePredict. The ego-vehicle can take five different actions: $\mathcal{A} =$ {"accelerate", "decelerate", "right lane-change", "left lane-change", "idle"}. The motivation behind the reward function $\mathcal{R}$ is based on our objective for training an agent that can safely and efficiently navigate in dense traffic while respecting other road agents in its neighborhood. The state transition matrix $\mathcal{T}$ boils down to a state transition probability $P(s'|s)$, which is defined as the probability of beginning from a current state s and arriving at a new state s'. This probability is calculated by the kinematics of the simulator, which depends on the underlying motion models, and thus it is equal to 1, establishing a deterministic setting.

Table V: We measure the average speed (Avg. Spd.) and the number of lane changes (#LC) for the ego-vehicle when it is (left) navigating in the default simulator (without any varying behaviors), (middle) interacting with conservative agents only, and (right) interacting with a combination of conservative and aggressive traffic-agents.

Model	Default		Conservative		Aggressive	
($n = 20$)	Avg. Spd. (m/s)	#LC	Avg. Spd. (m/s)	#LC	Avg. Spd. (m/s)	#LC
MLP	22.65	4.1	19.7	2.9	28.8	2.6
GCN	22.26	0.82	18.9	2.33	29	1.4
Model	Default		Conservative		Aggressive	
($n = 10$)	Avg. Spd. (m/s)	#LC	Avg. Spd. (m/s)	#LC	Avg. Spd. (m/s)	#LC
MLP	23.75	6.25	21.4	3.5	29.16	2.06
GCN	23.6	0.35	20.6	1.6	28.9	1.3

C. Benefits of Behaviorally-Guided Navigation

Experiment Setup: We evaluate both dense ($N = 20$) and sparse ($N = 10$) highway traffic scenarios, where N represents the number of vehicles. We perform experiments with two different RL policies using a graph convolutional network (GCN) [63] and a multi-layer perceptron (MLP) [26]. MLP and GCN are two different neural network architectures used to train a navigation policy using deep reinforcement learning. The choice of the underlying architecture determines, in part, the type of navigation policy trained and may result in different characteristics. The MLP and the GCN differ in several ways, including input data format, applications, and internal operations. We employ Q-Learning [41] to learn an autonomous navigation policy. Specifically, we use the MLP and GCN to receive observations of the state space as input and implicitly model the behavioral interactions between aggressive and conservative agents and the ego-vehicle. Ultimately, the MLP and GCN learn a function that receives a feature matrix that describes the current state of the traffic as input and provides us with the optimal Q values of the state space. Finally, the ego-vehicle can use the learned model during evaluation time in order to navigate its way around the traffic by choosing the best action that corresponds to the maximum Q value for its state at every time step.

We apply two metrics to evaluate over 100 episodes and average them at test time:

- Average Speed of the AV, which captures the distance per second covered in a varying time interval.
- Number of lane changes performed by the AV on average during the given duration. In general, fewer lane changes imply that the ego vehicle can cover the same distance

with fewer maneuvers.

Finally, we vary the behavior of traffic agents between conservative and aggressive behaviors and evaluate the performance of our action prediction and navigation algorithms. In the conservative environment (Table V, "Conservative" column), the environment is populated by conservative agents only, while in the aggressive environment ("Aggressive" column), a significant number of the agents are aggressive, but there are also some conservative agents.

Analysis and Insights: We observe two advantages of behaviorally-guided navigation, which we discuss below:

1) Intelligent Lane-Changing: We observe that behavior-guided navigation allows AVs to switch lanes more intelligently, than those navigating without any knowledge of driver behavior. More specifically, in conservative traffic the AV learns to confidently overtake slow-moving traffic whereas in aggressive traffic the AV executes fewer lane changes around aggressive agents in its vicinity.

To support this claim, we visualize this result in Figure 4. In Figures 4a, 4b, and 4c, we show that the AV quickly approaches the aggressive vehicle from the rear. However, rather than overtaking it (the AV has sufficient space), the AV chooses to decelerate and wait to resume acceleration when it is safe to do so. In contrast, in Figures 4d, 4e, and 4f, we show that the AV (green) confidently overtakes the conservative vehicle (blue).

However, when navigating without driver behavior modeling, the AV does not follow a consistent lane-changing pattern. For instance, we see in Table V under the "Default, #LC" column, that the AV performs an excessive number of lane changes with one policy (MLP), but mostly follows the traffic with very few lane changes with the other policy (GCN). Dependence of lane changes on the type of navigation policies is undesirable as the choice of policy often depends on several factors such as computational resources, type of application, different performance measuring metrics, and so on. These factors may vary for different situations and may potentially result in unsafe lane-changing in dense traffic. With the default simulator, the # of lane changes performed by a vehicle depends *exclusively* on the type of navigation policy (Table V - 4.1 for MLP and 0.82 for GCN in the dense traffic setting).

2) Adapting Speed to Different Behaviors: Behavior-guided navigation allows AVs to adapt their speeds to the nature of traffic around them. To be specific, AVs slow down more often in preparation to overtake slow moving conservative agents whereas they are able to maintain high speeds in fast moving aggressive traffic. We provide evidence for this result in Table V under the Avg. Spd. columns, where we show that the average speed of the ego-vehicle is lower in conservative environments than in aggressive environments. This variation of speed is desirable since it is reflects that ego-vehicle slows down to overtake conservative vehicles and is able to maintain a high speed in fast-moving aggressive traffic. Such adaptability results in better fuel efficiency, increased safety, and reduced likelihood of frustrations among drivers. However in default traffic, the ego-vehicle is unable to adapt its speed (Table V).

VII. CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

We have presented a new approach for driver behavior modeling that uses the idea of vertex centrality from computational graph theory to explicitly model the behavior of human drivers in realtime traffic using only the trajectories of the vehicles in the global coordinate frame. Our approach is robust, general, and can be integrated with existing navigation methods to perform behaviorally-guided navigation.

There are several interesting directions of future work. Our work is currently limited to straight roads. It would be useful to apply our approach to additional scenarios, including roundabouts, intersections, and merging. Another aspect of future work includes extending our approach for decision-making. While our current approach stops at modeling the driver behavior, a natural extension of our work includes combining our algorithm with motion and decision planning techniques for end-to-end self-driving.

APPENDIX A
PROVING $\|\tilde{\beta} - \beta\| = \mathcal{O}(\epsilon)$.

Proof. M is $T \times (d+1)$ Vandermonde matrix where $d \ll T$ is the degree of the resulting centrality polynomial. Vandermonde matrices are known to be ill-conditioned with high condition number $\kappa = \frac{\sigma_{\max}}{\sigma_{\min}}$ that increases exponentially with time T. From the noisy system given by Equation 4, we have,

$$\begin{aligned}
\tilde{\beta} &= (M^\top M)^{-1} M^\top \tilde{\zeta}^i \\
\tilde{\beta} &= (M^\top M)^{-1} M^\top (\zeta^i + \epsilon) \\
\tilde{\beta} &= \beta + (M^\top M)^{-1} M^\top \epsilon
\end{aligned} \tag{7}$$

From Equation 7, $\|\tilde{\beta} - \beta\| = \|(M^\top M)^{-1} M \epsilon\|$ which can be shown to be approximately in the order $\mathcal{O}(\kappa \epsilon)$. Therefore, the error between the true solution β and the estimated solution in the presence of noise $\tilde{\beta}$, depends on the condition number κ of the matrix M. A higher value of κ implies that the trailing singular values of $M^\top M$, denoted by $\Sigma = \{\sigma_1^2, \sigma_2^2, \ldots, \sigma_d^2\}$ have very small magnitudes. When inverting the matrix $M^\top M$, as in Equation 7, the singular values are inverted (by taking the reciprocal) and are represented by $\Sigma^{-1} = \{(\sigma_1^2)^{-1}, (\sigma_2^2)^{-1}, \ldots, (\sigma_d^2)^{-1}\}$. After the inversion, the trailing singular values now have large magnitudes, $\|(\sigma_i^2)^{-1}\| \gg 1$. When multiplied by ϵ, these large inverted singular values amplify the error, resulting in a large value of $\|\tilde{\beta} - \beta\|$. In Figure 5, it can be seen by the red curve that under general conditions, κ increases exponentially for even small matrices (we fix $d = 2$ and increase T from 0 to 20).

However, there are many techniques that bound the condition number of a matrix by regularizing its singular values. In our application, we use the well-known Tikhonov regularization [3]. Under this regularization, after the inversion operation is applied on $M^\top M$, the magnitude of the resulting inverted singular values are constrained by adding a parameter α. The modified inverted singular values can be expressed as $\frac{\sigma_i}{\sigma_i^2 + \alpha^2}$. The addition of the α^2 in the denominator keeps the overall magnitude of the inverted singular value from "blowing up". In our approach, as M is fixed for every T, we need only
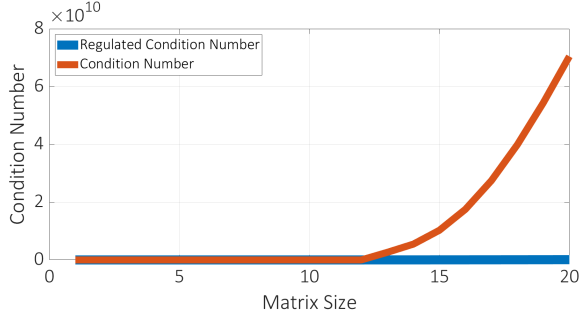
Figure 5: **Robustness to Noise:** We show that by regularizing the noisy OLS system given by Equation 4, we can reduce the original condition number (red curve) while at the same time upper bounding the reduced condition number (blue curve) by $\delta \longrightarrow 1$. The reduced condition number helps stabilize the noisy estimator $\tilde{\beta}$.

search for the optimal α once for every T. Using the Tikhonov regularization, we upper bound the condition number κ by $\delta \to 1$ (Figure 5). Therefore $\|\tilde{\beta} - \beta\| = \mathcal{O}(\kappa\epsilon) = \mathcal{O}(\epsilon)$ as $\kappa \to 1$.

$\square$

APPENDIX B
SIMULATION PARAMETERS

We use the Highway-Env simulator [34] developed using PyGame [56]. The simulator consists of a 2D environment where vehicles are made to drive along a multi-lane highway using the Bicycle Kinematic Model [45] as the underlying motion model. The linear acceleration model is based on the Intelligent Driver Model (IDM) [59], while the lane changing behavior is based on the MOBIL [31] model.

The linear acceleration model is based on the Intelligent Driver Model (IDM) [59] and is computed via the following kinematic equation,

$$\dot{v}_\alpha = a\left[1 - \left(\frac{v_\alpha}{v_0^\alpha}\right)^4 - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right] \qquad (8)$$

Here, the linear acceleration, $\dot{v}_\alpha$, is a function of the velocity v_α, the net distance gap s_α and the velocity difference Δv_α between the ego-vehicle and the vehicle in front. Equation 8 is a combination of the acceleration on a free road $\dot{v}_{free} = a[1 - (v/v_0)^4]$ (*i.e.* no obstacles) and the braking deceleration, $-a(s^*(v_\alpha, \Delta v_\alpha)/s_\alpha)^2$ (*i.e.* when the ego-vehicle comes in close proximity to the vehicle in front). The deceleration term depends on the ratio of the desired minimum gap ($s^*(v_\alpha, \Delta v_\alpha)$) and the actual gap ($s_\alpha$), where $s^*(v_\alpha, \Delta v_\alpha) = s_0 + vT + \frac{v\Delta v}{2\sqrt{ab}}$. s_0 is the minimum distance in congested traffic, vT is the distance while following the leading vehicle at a constant safety time gap T, and a, b correspond to the comfortable maximum acceleration and comfortable maximum deceleration, respectively.

The lane changing behavior is based on the MOBIL [31] model. According to this model, there are two key parameters when considering a lane-change:

1) *Safety Criterion*: This condition checks if, after a lane-change to a target lane, the ego-vehicle has enough room to accelerate. Formally, we check if the deceleration

Table VI: We show the simulation parameters that define the conservative and aggressive vehicle classes.

Model	Parameter	Conservative	Aggressive
IDM	Time gap (T)	1.5s	1.2s
	Min distance (s_0)	5.0 m	2.5 m
	Max comfort acc. (a)	3.0 m/s^2	6.0 m/s^2
	Max comfort dec. (b)	6.0 m/s^2	9.0 m/s^2
MOBIL	Politeness (p)	0.5	0
	Min acc gain (Δa_{th})	0.2 m/s^2	0 m/s^2
	Safe acc limit (b_{safe})	3.0 m/s^2	9.0 m/s^2

of the successor a_{target} in the target lane exceeds a pre-defined safe limit b_{safe}:

$$a_{\text{target}} \geq -b_{safe}$$

2) *Incentive Criterion*: This criterion determines the total advantage to the ego-vehicle after the lane-change, measured in terms of total acceleration gain or loss. It is computed with the formula,

$$\tilde{a}_{\text{ego}} - a_{\text{ego}} + p(\tilde{a}_n - a_n + \tilde{a}_o - a_o) > \Delta a_{th}$$

where $\tilde{a}_{\text{ego}} - a_{\text{ego}}$ represents the acceleration gain that the ego-vehicle would receive after to the lane change. The second term denotes the total acceleration gain/loss of the immediate neighbours (the new follower in the target, a_n, and the original follower in the current lane, a_o) weighted with the politeness factor, p. By adjusting p the intent of the drivers can be changed from purely egoistic ($p = 0$) to more altruistic ($p = 1$). We refer the reader to [31] for further details.

The lane change is executed if both the safety criterion is satisfied, *and* the total acceleration gain is more than the defined minimum acceleration gain, Δa_{th}.

Additionally, the desired velocity v_0 is set to 25 meters per second and 40 meters per second for the conservative and aggressive vehicle classes, respectively. Finally, the desired velocities for the conservative vehicles were uniformly distributed with a variation of $\pm 10\%$ to increase the heterogeneity in the simulation environment.

REFERENCES

[1] K. I. Ahmed, "Modeling drivers' acceleration and lane changing behavior," Ph.D. dissertation, MIT, 1999.
[2] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.
[3] D. Calvetti and L. Reichel, "Tikhonov regularization of large linear problems," *BIT Numerical Mathematics*, vol. 43, no. 2, pp. 263–283, 2003.
[4] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[5] R. Chandra, A. Bera, and D. Manocha, "Stylepredict: Machine theory of mind for human driver behavior from trajectories," *arXiv preprint arXiv:2011.04816*, 2020.

[6] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, "Densepeds: Pedestrian tracking in dense crowds using front-rvo and sparse features," *arXiv preprint arXiv:1906.10313*, 2019.

[7] ——, "Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8483–8492.

[8] R. Chandra, U. Bhattacharya, T. Mittal, A. Bera, and D. Manocha, "Cmetric: A driving behavior measure using centrality functions," *arXiv preprint arXiv:2003.04424*, 2020.

[9] R. Chandra, U. Bhattacharya, T. Mittal, X. Li, A. Bera, and D. Manocha, "Graphrqi: Classifying driver behaviors using graph spectrums," *arXiv preprint arXiv:1910.00049*, 2019.

[10] R. Chandra, U. Bhattacharya, T. Randhavane, A. Bera, and D. Manocha, "Roadtrack: Realtime tracking of road agents in dense and heterogeneous environments," *arXiv*, pp. arXiv–1906, 2019.

[11] R. Chandra, U. Bhattacharya, C. Roncal, A. Bera, and D. Manocha, "Robusttp: End-to-end trajectory prediction for heterogeneous road-agents in dense traffic with noisy sensor inputs," in *ACM Computer Science in Cars Symposium*, 2019, pp. 1–9.

[12] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha, "Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms," *IEEE Robotics and Automation Letters*, 2020.

[13] R. Chandra, M. Mahajan, R. Kala, R. Palugulla, C. Naidu, A. Jain, and D. Manocha, "Meteor: A massive dense & heterogeneous behavior dataset for autonomous driving," *arXiv preprint arXiv:2109.07648*, 2021.

[14] R. Chandra and D. Manocha, "Gameplan: Game-theoretic multi-agent planning with human drivers at intersections, roundabouts, and merging," *arXiv preprint arXiv:2109.01896*, 2021.

[15] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] E. Cheung, A. Bera, E. Kubin, K. Gray, and D. Manocha, "Identifying driver behaviors using trajectory features for vehicle navigation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3445–3452.

[17] N. Corp., "Tesla unveils top av training supercomputer powered by nvidia a100 gpus," https://blogs.nvidia.com/blog/2021/06/22/tesla-av-training-supercomputer-nvidiaa100-gpus/, 2021.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[19] A. Davies, "Google's self-driving car caused its first crash," 2016.

[20] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, "Sharp minima can generalize for deep nets," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1019–1028.

[21] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[22] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[23] T. Fan, X. Cheng, J. Pan, D. Monacha, and R. Yang, "Crowdmove: Autonomous mapless navigation in crowded scenarios," *arXiv preprint arXiv:1807.07870*, 2018.

[24] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," *arXiv preprint arXiv:1808.03841*, 2018.

[25] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments: Part ii," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1070–1076.

[26] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[27] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[28] T. Guan, J. Wang, S. Lan, R. Chandra, Z. Wu, L. Davis, and D. Manocha, "M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers," *arXiv preprint arXiv:2104.11896*, 2021.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[31] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[32] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[33] G. Lawyer, "Understanding the influence of all nodes in a network," *Scientific reports*, vol. 5, no. 1, pp. 1–9, 2015.

[34] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," *arXiv preprint arXiv:1911.12250*, 2019.

[35] C. Li, Y. Meng, S. H. Chan, and Y.-T. Chen, "Learning 3d-aware egocentric spatial-temporal interaction via graph convolutional networks," *arXiv preprint arXiv:1909.09272*, 2019.

[36] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 387–396.

[37] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[38] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[39] D. Manocha, *Algebraic and numeric techniques in modeling and robotics*. University of California at Berkeley, 1992.

[40] A. Mavrogiannis, R. Chandra, and D. Manocha, "B-gap: Behavior-guided action prediction for autonomous navigation," *arXiv preprint arXiv:2011.03748*, 2020.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[42] S. A. Morelli, D. C. Ong, R. Makati, M. O. Jackson, and J. Zaki, "Empathy and well-being correlate with centrality in different social networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 37, pp. 9843–9847, 2017.

[43] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[44] M. Park, K. Jang, J. Lee, and H. Yeo, "Logistic regression model for discretionary lane changing under congested traffic," *Transportmetrica A: transport science*, vol. 11, no. 4, pp. 333–344, 2015.

[45] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 812–818.

[46] G. Qi, Y. Du, J. Wu, and M. Xu, "Leveraging longitudinal driving behaviour data with data mining techniques for driving style analysis," *IET intelligent transport systems*, vol. 9, no. 8, pp. 792–801, 2015.

[47] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. Eslami, and M. Botvinick, "Machine theory of mind," *arXiv preprint arXiv:1802.07740*, 2018.

[48] F. A. Rodrigues, "Network centrality: An introduction," *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*, p. 177, 2019.

[49] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and Systems*, 2016.

[50] F. Sagberg, Selpi, G. F. Bianchi Piccinini, and J. Engström, "A review of research on driving styles and road safety," *Human factors*, vol. 57, no. 7, pp. 1248–1275, 2015.

[51] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[52] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24 972–24 978, 2019.

[53] D. Seth and M. L. Cummings, "Traffic efficiency and safety impacts of autonomous vehicle aggressiveness," *simulation*, vol. 19, p. 20, 2019.

[54] B. Shi, L. Xu, J. Hu, Y. Tang, H. Jiang, W. Meng, and H. Liu, "Evaluating driving styles by normalizing driving behavior based on personalized

driver modeling," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, pp. 1502–1508, 2015.

[55] X. Shi, Z. Wang, X. Li, and M. Pei, "The effect of ride experience on changing opinions toward autonomous vehicle safety," *Communications in Transportation Research*, vol. 1, p. 100003, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772424721000032

[56] P. Shinners, "Pygame," http://pygame.org/, 2011.

[57] M. Šikić, A. Lančić, N. Antulov-Fantulin, and H. Štefančić, "Epidemic centrality—is there an underestimated epidemic impact of network peripheral nodes?" *The European Physical Journal B*, vol. 86, no. 10, p. 440, 2013.

[58] D. Tesla, "25 miles of full self driving | tesla challenge 2 | autopilot |," https://www.youtube.com/watch?v=Rm8aPR0aMDE, 2019.

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

[60] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[61] W. Wang, J. Xi, A. Chong, and L. Li, "Driving style classification using a semisupervised support vector machine," *IEEE Transactions on Human-Machine Systems*, vol. 47, pp. 650–660, 2017.

[62] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[63] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, p. 11, 2019.

[64] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

Dinesh Manocha is the Paul Chrisman-Iribe Chair in Computer Science & Electrical and Computer Engineering and Distinguished University Professor at University of Maryland College Park. His research interests include virtual environments, physically-based modeling, and robotics.He has published more than 600 papers & supervised 40 PhD dissertations. He is an inventor of 10 patents, which are licensed to industry. He is a Fellow of AAAI, AAAS, ACM, and IEEE, member of ACM SIGGRAPH Academy, and Bézier Award recipient from Solid Modeling Association. He received the Distinguished Alumni Award from IIT Delhi the Distinguished Career in Computer Science Award from Washington Academy of Sciences. He was a co-founder of Impulsonic, a developer of physics-based audio simulation technologies, which was acquired by Valve Inc in November 2016.

Rohan Chandra is a PhD student in computer science at University of Maryland, College Park, USA (UMD). He received a Bachelors degree in ECE from Delhi Technological University, New Delhi, India, and a Masters degree in computer science from UMD. His research interests include trajectory prediction, behavior modeling, and navigation in autonomous driving.

Aniket Bera is an Assistant Research Professor in the Department of Computer Science with affiliated appointments with the Brain and Behavior Institute and Maryland Robotics Center. Prior to this, he was a Research Assistant Professor at the University of North Carolina at Chapel Hill, where he also received his Ph.D. in 2017. His core research interests are in Human Modeling and Simulation, Affective Computing, and Virtual Reality. He is a faculty with the GAMMA group and has advised and co-advised over 20 MS and Ph.D. students. He has authored over 50 papers and his work has won multiple awards at top VR conferences. His research involves novel combinations of methods and collaborations in affective computing, computer graphics, computational psychology, machine learning, and social robotics to develop real-time computational models to learn and simulate human-like agents with expressive behaviors. Dr. Bera has previously worked in many research labs, including Disney Research and Intel Labs, and his work has been featured in many media outlets including Forbes, WIRED, FastCompany, etc. A more detailed description can be found at http://cs.umd.edu/ ab.