ABSTRACT

Title of Dissertation: TOWARDS AUTONOMOUS DRIVING IN DENSE, HETEROGENEOUS, AND UNSTRUCTURED TRAFFIC

Rohan Chandra
Doctor of Philosophy, 2022

Dissertation Directed by: Professor Dinesh Manocha
Department of Computer Science

This dissertation addressed many key problems in autonomous driving towards handling dense, heterogeneous, and unstructured traffic environments. Autonomous vehicles (AV) at present are restricted to operating on smooth and well-marked roads, in sparse traffic, and among well-behaved drivers. We developed new techniques to perceive, predict, and plan among human drivers in traffic that is significantly denser in terms of number of traffic-agents, more heterogeneous in terms of size and dynamic constraints of traffic agents, and where many drivers do not follow the traffic rules. In this thesis, we present work along three themes—perception, driver behavior modeling, and planning. Our novel contributions include:

1. Improved tracking and trajectory prediction algorithms for dense and heterogeneous traffic using a combination of computer vision and deep learning techniques.

2. A novel behavior modeling approach using graph theory for characterizing human drivers as aggressive or conservative from their trajectories.

3. Behavior-driven planning and navigation algorithms in mixed (human driver and AV) and unstructured traffic environments using game theory and risk-aware control.

Additionally, we have released a new traffic dataset, METEOR, which captures rare and interesting, multi-agent driving behaviors in India. These behaviors are grouped into traffic violations, atypical interactions, and diverse scenarios. We evaluate our perception work on tracking and trajectory prediction using standard autonomous driving datasets such as the Waymo Open Motion, Argoverse, NuScenes datasets, as well as public leaderboards where our tracking approach resulted in achieving rank 1 among over a 100 methods. We apply human driver behavior modeling in planning and navigation at unsignaled intersections and highways scenarios using state-of-the-art traffic simulators and show that our approach yields fewer collisions and deadlocks compared to methods based on deep reinforcement learning. We conclude the presentation with a discussion on future work.

TOWARDS AUTONOMOUS DRIVING IN DENSE, HETEROGENEOUS, AND UNSTRUCTURED TRAFFIC

by

Rohan Chandra

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Dr. Dinesh Manocha, Chair/Advisor
Dr. Derek Paley, Dean's Rep
Dr. Yiannis Aloimonos
Dr. Pratap Tokekar
Dr. Mac Schwager

To my family.

Acknowledgments

This dissertation was carried out through the collective efforts of many individuals. My advisor Dinesh Manocha deserves, of course, the first mention. His influence on my research extends beyond the boilerplate research supervision. Dinesh helped me understand the importance of not losing the forest for the trees in all aspects of scientific communication. I am grateful for the freedom he provided in order for me to pursue a wide range of exciting ideas to tackle important problems in autonomous driving.

Many thanks to my committee members—Dr. Yiannis Aloimonos, Dr. Derek Paley, Dr. Mac Schwager, and Dr. Pratap Tokekar—who graciously accommodated my back-and-forth emails trying to finalize a date for the defense. I thank Yiannis Aloimonos for many spirited conversations during my early years at UMD as well as for writing a letter of recommendation for me that helped me get into the PhD program at UMD. I also thank Pratap Tokekar for his advice and guidance in the postdoc search. Finally, I am grateful for the collaboration with Mac Schwager where I learned about risk sensitivity analysis and its interplay with driver behavior modeling and game theory. Many of these ideas form the basis of many future projects I have planned. The last section in Chapter 4 is the result of our joint work.

Next, I am grateful to all my co-authors and collaborators—Aniket Bera, Uttaran Bhattachrya, Tianrui Guan, Divya Kothandaraman, Angelos Mavrogiannis, and Trisha Mittal, . It would also be appropriate to mention here the people who indirectly provided the logistical and infrastructural

support to my research—our department coordinator Tom Hurst along with the entire staff of UMD CS and UMIACS. Finally, I am grateful to Tom Goldstein and Jordan Boyd-Graber for writing letters of recommendation for PhD programs and assisting me in the application process, and to Ashok Agrawala for his wisdom in all matters important.

Moving on, I am lucky to have a strong personal support system consisting of family and friends, who made research fun. I acknowledge all the members of "Manocha's Minions"—in particular, Senthil "Sentinel" Arul, Trisha Mittal, Utsav Patel, Adarsh Jagan, Pooja Guhan, Kasun Weerakoon, Uttaran Bhattacharya—and all other members of GAMMA group. I would also like to thank friends outside of GAMMA as well as those outside of UMD.

Finally, I am grateful for the continual support of my mom, dad, and brother, whose induction into the U.S. Marine Corps. became a constant source of inspiration for me.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Autonomous vehicles (AVs), intelligent transportation, electric vehicles, and advanced driving assistance systems (ADAS) are an active area of research in many scientific fields such as engineering, computer vision, artificial intelligence, control theory, and robotics. Researchers in these fields from both the academic and industry communities tackle problems related to sensing, perception, prediction, planning, and controls. Autonomous vehicles currently operate only on smooth, clearly marked roads, in sparse to moderate traffic, in clear weather, in the daytime, and, most importantly, among predictable, well-behaved drivers. We refer to such traffic as *structured* traffic.

Fully autonomous driving, however, require AVs to robustly, safely, and efficiently operate in *unstructured* environments. Examples of unstructured elements include objects previously unseen by the vehicle's perception system, idiosyncrasies—often unpredictable—of human drivers, hazardous weather and road conditions, and so on. Failure to deal with such scenarios, which occur more frequently than one might imagine, may result in accidents. Bridging the gap between structured and unstructured traffic environments is, therefore, one of the central open questions in autonomous driving research. There are three characteristics that elevate the difficulty of autonomous driving in such environments:

1. *High Density:* Figure 1.1 depicts a typical dense traffic scenario in developing nations.

Figure 1.1: Unstructured traffic conditions. Note the high density and high heterogeneity, lack of lane markings, and overall unstructured nature of traffic.

Since most computer vision AD algorithms rely on front-facing visual sensors, objects may be occluded from the AV's field of view raising safety concerns.

2. *High Heterogeneity:* Heterogeneity refers to the many unique types of road-agents in a traffic scenario. For example, in Figure 1.1, we can observe vans, cars, two-wheelers, and even unique agents like a cart loaded with over-sized poles.

3. *Non-conformity (unstructured):* Non-conformity among traffic-agents refers to the scenario where drivers do not obey traffic laws and social protocols including right-of-way rules, lane following, driving under the speed limit, etc.

Our thesis pushes the boundaries of autonomous driving towards dense, heterogeneous, and unstructured traffic environments, typical of developing nations such as India. We develop

new techniques to perceive, predict, and plan among human drivers in traffic that is 10 times denser, twice as heterogeneous, and where more drivers break traffic rules than follow them. In this dissertation, we present the challenges of perception, prediction, and planning in dense, heterogeneous, and unstructured traffic, and discuss ways to tackle them.

1.1 Main Contributions

The core contributions of this thesis are:

1. **Improved tracking in dense and unstructured traffic—**Using a combination of computer vision and deep learning techniques, we propose a new tracking algorithms that improve perception in dense and unstructured environments, outperforming over a 100 methods on standard benchmarks. To deal with the high density, we propose the following:

   (a) A better feature representation: Existing trackers that work for sparse traffic scenarios use rectangular bounding boxes to localize agents. Such boxes, however, overlap in dense traffic scenarios due to the close proximity between agents. Therefore, an alternate localization function, such as semantic segmentation, is required that targets pixels that belong only to the agent under consideration. We provide analytical guarantees for improved performance of using segmentation as the feature representation compared to using bounding boxes.

   (b) A better motion model: Velocity-based motion models are used to plan agent motion in dense settings. The reciprocal velocity obstacles (RVO), however, models agents as circular discs in a 2D plane from a top-down view. Most autonomous driving tracking

systems, on the other hand, assume a front-facing camera. In our work, we modify the RVO implementation for front-facing cameras by modeling agents as ellipses.

2. **New trajectory prediction algorithms for dense, heterogeneous, and unstructured traffic—** We propose a novel attention mechanism to take into account the different shapes, sizes, and mobility of the different types of agents. We also integrate our work on tracking with trajectory prediction to perform trajectory prediction in dense traffic. Finally to deal with unstructured traffic consisting of non-confirming agents, we combine behavior prediction with trajectory prediction to predict driver behavior as aggressive or conservative.

3. **A new algorithm for online driver behavior modeling—** Given the input trajectories of vehicles in a period of time, we characterize, in realtime, the driving style (on a continuous scale from aggressive to conservative) of the vehicle using graph theory. Our approach is generally applicable and we demonstrated our algorithm in India, China, Singapore, and USA. We further show that our algorithm is accurate up to human-level accuracy.

4. **Behavior compliant planning and navigation—**We propose a new framework for planning and navigation in human environments using mechanism design. This thesis presents the first application of mechanism design, a tool previously limited to economics and algorithm design, to robotics and optimal planning. The main benefit of this framework lies in its ability to perform optimal decision-making with humans-in-the-loop, *without requiring access to the humans' objective functions.* Our results from experiments at dynamic traffic intersections show that our framework improves throughput by at least 25%, time taken to reach the goal by 75%, and fuel consumption by 33% compared to auction-based approaches and signaled approaches using traffic-lights and stop signs.

5. **Software and datasets—**New datasets containing traffic from India consisting of rare and interesting, multiagent driving behaviors grouped into traffic violations, atypical interactions, and diverse scenarios. Open-source software to integrate the algorithms for perception, prediction, and planning into an end-to-end pipeline.

1.1.1 Applications

This thesis is expected to contribute to the following domains:

- **Multi-Agent Systems:** This is a broad area in the field of artificial intelligence. Our work on mechanism design and planning can extend to this area and solve issues related to coordination, negotiation, cooperation, competition, and teaming among intelligent agents.
- **Robotics:** Our work can extend naturally to ground robots operating in unstructured indoor and outdoor environments. Specific problems that this thesis could tackle include navigation in dense crowds and task-and-motion planning.
- **Artificial Intelligence and Reinforcement Learning:** Our thesis can address important problems in multi-agent reinforcement learning using the proposed concepts of mechanism design applied to planning and optimal control.

1.2 Overview of the thesis

Our thesis presents new algorithms for perception (tracking, trajectory prediction, and behavior prediction), online driver behavior modeling, and behaviorally compliant planning, in dense, heterogeneous, and unstructured traffic. We organize the material as follows—

- **Chapter 2—Perception in Dense and Heterogeneous Environments:** Our contributions included developing new techniques for visual perception in dense and heterogeneous traffic. AVs must perceive the dynamic environment around them and extract useful information. This includes detecting vehicles and static obstacles, tracking the movement of all road-agents, and predicting their future trajectories and behaviors. Starting with tracking, we describe two key challenges in dense, heterogeneous, and unstructured traffic environments. First, it is easy to miss occluded agents which results in a large number of false negatives and second, tracks are fragile in that they can be easily fragmented. We propose two algorithms that tackle these challenge. The first algorithm is a tracking-by-detection algorithm where we employ bounding box subtraction to remove background noise from an image patch corresponding to an agent. Our background subtraction approach ranked 1^{st} on the state-of-the-art benchmark leader-board at the time of submission in terms of fewest false negatives.. The second algorithm extends the previous algorithm by incorporating a novel motion model that simultaneously takes into account collision avoidance and inter-agent interactions to bolster current tracks, thereby addressing the second challenge.

  The next part in this chapter includes effective methods to predict the trajectories of human drivers. However, there are several issues associated with such environments that make prediction challenging. Highly dense traffic corresponds to more frequent inter-agent interactions which are hard to model due to the inherent uncertainty in human behavior. Moreover, prediction algorithms require huge datasets for training, the collection of which is a costly and time-consuming process. Finally, AVs must simultaneously perform low-level trajectory and high-level action prediction for real time navigation as opposed to current state-of-the-

art which handles trajectory and action prediction independent of each other. We developed three algorithms that address the limitations described above. In the first algorithm, the key aspect is to selectively focus attention on fewer agents. The algorithm consists of a novel attention mechanism that teaches the ego-vehicle to identify the agents that deserve more importance than others. For instance, a pedestrian in the way of the ego-vehicle requires more attention than, say, a parked car to the side. The second approach is an end-to-end approach that does not require manually labeled ground-truth trajectories to train the trajectory prediction network. The input to this algorithm consists only of raw traffic videos obtained from commodity sensors such as monocular RGB cameras. The algorithm uses a tracking algorithm to generate noisy trajectories from these videos. These trajectories replace the trajectory input used by the first algorithm. In the final algorithm, we perform simultaneous trajectory and action prediction. Architecturally, the network consists of a two-stream approach working in parallel. The first stream is essentially the first trajectory prediction algorithm while the second stream is used to perform action prediction.

- **Chapter** 3**—Online Driver Behavior Modeling:**Following our research on perception and prediction, we shifted our focus on planning and navigation in unstructured traffic scenarios. In such scenarios however, human driver behavior plays a key role in building better planning and navigation algorithms. For instance, developing nations do not have right-of-way rules at intersections. Human drivers often have to navigate such intersections by gauging the behavior of the other drivers and negotiate the intersection accordingly. We propose a new algorithm for modeling and characterizing human driver behavior as different levels of aggressiveness using the vehicle trajectories as input. Our contributions

in driver behavior modeling are the following. We propose a new metric, called **CMetric** [14], to quantify driver behavior from raw vehicle trajectories obtained from commodity sensors such as cameras and lidars. My approach uses the concept of vertex centrality functions from graph theory to measure the likelihood and intensity of driving styles such as overspeeding, overtaking, sudden lane-changes, etc. We test CMetric in both simulation and real-world traffic and evaluate its accuracy by comparing it to human evaluation. Specifically, we measure the time difference between the moments when a human identifies an aggressive behavior and when CMetric identifies the same behavior. Our experiments showed that CMetric can identify different behaviors with a time difference of less than 0.02 seconds. In a follow-on work, **GraphRQI** [15], we convert the CMetric approach into a machine learning algorithm by formulating the behavior prediction task as a multi-class classification problem.

- **Chapter 4—Behaviorally Compliant Planning in Human Environments:** We proposed three Planning-Algorithms for planning among AVs and human drivers at intersections with no traffic lights, signs, and most importantly, no right-of-way rules (such as the first-in first-out principle). The goal is to route drivers according to a turn-based ordering that minimizes time-to-goal in a way such that each driver is envy-free. We develop a new auction format that propounds a new allocation rule based on the behavior of each driver. We show that such an auction format is incentive-compatible, utility-maximizing, and computable in polynomial time.

- **Chapter 5—Software and Datasets:** We present a new traffic dataset, METEOR, which captures traffic patterns and multi-agent driving behaviors in unstructured scenarios. METEOR consists

of more than 1000 one-minute videos, over 2 million annotated frames with bounding boxes and GPS trajectories for 16 unique agent categories, and more than 13 million bounding boxes for traffic agents. METEOR is a dataset for rare and interesting, multi-agent driving behaviors that are grouped into traffic violations, atypical interactions, and diverse scenarios. Every video in METEOR is tagged using a diverse range of factors corresponding to weather, time of the day, road conditions, and traffic density. We use METEOR to benchmark perception methods for object detection and multi-agent behavior prediction. Our key finding is that state-of-the-art models for object detection and behavior prediction, which otherwise succeed on existing datasets such as Waymo, fail on the METEOR dataset. METEOR marks the first step towards the development of more sophisticated perception models for dense, heterogeneous, and unstructured scenarios.

- **Chapter 6—Limitations, Conclusion, and Future Work:**My long-term vision is to design robust, efficient, and safe multi-agent AI systems in which autonomous agents can **naturally act** in unstructured human environments. In striving towards this naturalness, autonomous agents must efficiently co-exist with humans in both cooperative and competitive scenarios. My research will draw upon various tools in AI including planning and optimal control, algorithmic game theory, and deep learning with applications in robotics and computer vision. A major challenge in planning and decision-making in human environments is to reason about the actions of the human agents. Several strategies have been used to varying degrees of success including worst-case analysis, probabilistic uncertainty, and dynamic game theory. In the first two cases, the human agent is framed as nature, whereas in the third case, the agent is assumed to optimize their particular objective function. In general

cases, this objective function is unknown. A large body of work has been done to learn the objective function via IRL. The challenges with IRL include necessity of large corpus of data and instability in training. I also plan to extend my dissertation research on safe, efficient, and robust autonomous driving in unstructured traffic along several themes.

Chapter 2: Perception for Dense and Heterogeneous Traffic

2.1 Overview

Perception is the task of collecting, processing, and analyzing visual, semantic, and other multi-modal information from the environment of an AV via various sensors like cameras, lidars, radars, IMUs, GPS, etc. Simply put, perception acts as the eyes of the AV. In this dissertation, we study the tracking and trajectory prediction of the ego-vehicle in dense and heterogeneous traffic environments. Tracking and trajectory prediction are integral components of autonomous driving software; without them, AVs cannot safely and efficiently plan in dynamic environments consisting of multiple vehicles. In light of their importance, research in tracking and trajectory prediction has exploded in recent times. But despite these advances, current solutions are still not robust enough in dense and heterogeneous traffic environments. There are three key challenges in such environments—

1. **High density traffic cause occlusions.** Especially when recorded from a front-facing camera, which causes the tracking algorithms to fail to sense critical objects.

2. **High heterogeneity:** Different road-agents, with varying shape and size, maneuver differently and result in different trajectories.

3. **Lack of robustness in unstructured environments:** The trajectories of agents in unstructured

traffic are inherently noisy. Trajectory prediction algorithms are inherently noisy to these trajectories.

We conclude this chapter with our work on joint trajectory and behavior prediction.

2.2 Related Work

2.2.1 Object Detection

Early methods for object detection include HOG [16] and SIFT [17], which manually extract features from images. Inspired by AlexNet, [18] proposed R-CNN and its variants [19, 20] for optimizing the object detection problem by incorporating a selective search.

More recently, the prevalence of CNNs has led to the development of Mask R-CNN [6], which extends Faster R-CNN to include pixel-level segmentation. The use of Mask R-CNN in pedestrian tracking has been limited, although it has been used for other pedestrian-related problems such as pose estimation [21].

2.2.2 Pedestrian and Vehicle Tracking

There is extensive work on pedestrian tracking [22, 23]. Bruce et al. [24] and Gong et al. [25] predict pedestrians' motions by estimating their destinations. Liao et al. [26] compute a Voronoi graph from the environment and predict the pedestrian motion along the edges. Mehran et al. [27] apply the social force model to detect anomalous pedestrian behaviors from videos. Pellegrini et al. [28] use an energy function to build a goal-directed short-term collision-avoidance motion model. Bera et al. [29, 30] use reciprocal velocity obstacles and hybrid motion models

to improve the accuracy. All these methods are specifically designed for tracking pedestrian movement.

Vehicle tracking has been studied in computer vision, robotics, and intelligent transportation. Some of the earlier techniques are based on using cameras [31] and laser range finders [32]. The authors of [33] model dynamic and geometric properties of the tracked vehicles and estimate their positions using a stereo rig mounted on a mobile platform. Ess et al. [34] present an approach to detect and track vehicles in highly dynamic environments. Multiple cameras have also been used to perform tracking all surrounding vehicles [35, 36]. Moras et al. [37] use an occupancy grid framework to manage different sources of uncertainty for more efficient vehicle tracking; Wojke et al. [38] use LiDAR for moving vehicle detection and tracking in unstructured environments. Finally, [39] uses a feature-based approach to track the vehicles under varying lighting conditions. Most of these methods focus on vehicle tracking and do not take into account interactions with other road-agents such as pedestrians, two-wheelers, rickshaws etc. in dense urban environments. For an up-to-date review of tracking-by-detection algorithms, we refer the reader to methods submitted to the MOT benchmark [40].

2.2.3 Motion Models in Pedestrian Tracking

Motion models are commonly used in pedestrian tracking algorithms to improve tracking accuracy [41, 42]. [42] presents a variation of MHT and shows that it is at par with the state-of-the-art from the tracking-by-detection paradigm. [43] uses a motion model to combine fragmented pedestrian tracks caused by occlusion. These methods are based on linear constant velocity or acceleration models. Such linear models, however, cannot characterize pedestrian dynamics in

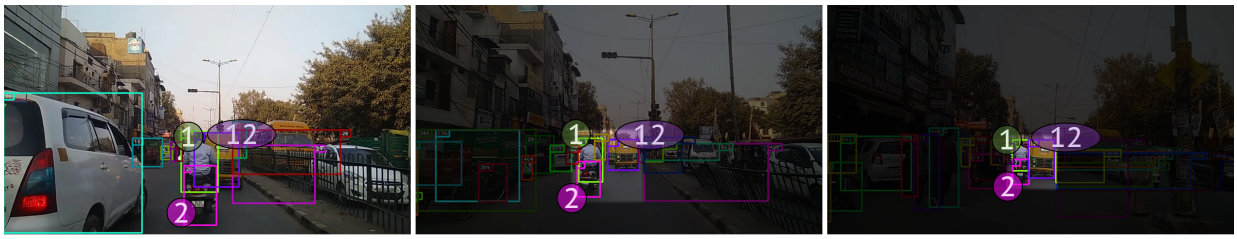


Figure 2.1: Tracking in dense, heterogeneous, and unstructured environments. The circled numbers indicate the agent IDs.

dense crowds [44]. RVO [3] is a non-linear motion model that has been used for pedestrian tracking in dense videos to compute intermediate goal locations, but it only works with top-facing videos and circular pedestrian representations. RVO has been extended to tracking road-agents such as cars, buses, and two-wheelers, in addition to pedestrians, using a linear runtime motion model that considers both collision avoidance and pair-wise heterogeneous interactions between road-agents [45]. Other motion models used in pedestrian tracking are the Social Force model [46], LTA [47], and ATTR [48].

There are also many discrete motion models that represent each individual or pedestrian in a crowd as a particle (or as a 2D circle on a plane) to model the interactions. These include models based on repulsive forces [2] and velocity-based optimization algorithms [49], [3]. More recent discrete approaches are based on short-term planning using a discrete approach [50] and cognitive models [51]. However, these methods are based on circular agent representation and do not work well for front-facing pedestrians in dense crowd videos as they are overly conservative in terms of motion prediction.

2.3 Tracking in dense traffic

My initial contributions as a graduate student included developing new techniques for visual perception in dense and heterogeneous traffic. AVs must perceive the dynamic environment around them and extract useful information. This includes detecting vehicles, static obstacles, and tracking the movement of all road-agents (Figure 2.1). There are two key challenges associated with tracking and detection in dense, heterogeneous, and unstructured traffic environments. First, it is easy to miss occluded agents which results in a large number of false negatives and second, tracks are fragile in that they can be easily fragmented. I propose two algorithms, DensePeds and RoadTrack, that tackle these challenge:

1. **DensePeds** [12] is a tracking-by-detection algorithm where I employ bounding box subtraction to remove background noise from an image patch corresponding to an agent. My background subtraction approach ranked 1st on the state-of-the-art benchmark leader-board at the time in terms of fewest false negatives. This work has been published in **IROS'19**.

2. **RoadTrack** [45] extends DensePeds by incorporating a novel motion model that simultaneously takes into account collision avoidance and inter-agent interactions to bolster current tracks, thereby addressing the second challenge. This work has been published in **ICRA'20**.

2.3.1 Tracking by Detection

Informally, the tracking problem is stated as follows: Given a video, we want to assign an ID to all road-agents in all frames. This is formally equivalent to solving the following sub-problem at each time-step (or frame): At current time t, given the ID labels of all road-agents in

Figure 2.2: Overview of RoadTrack: We use Mask R-CNN on an input frame at time t to generate segmented boxes [12]. We use SimCAI to predict the agent's state at frame $t+1$. We generate features that are invariant to shape, size and scale of heterogeneous road-agents. These features are matched using association algorithms and a tracking ID is assigned to each predicted agent based on feature matching.

the frame, assign labels for road-agents in the next frame (time $t+1$).

We start by using Mask R-CNN to implicitly perform pixel-wise segmentation of the road-agents. This generates a set of segmented boxes [12]. For each detected road-agent, h_j, generated using Mask R-CNN, we extract their corresponding features, f_{h_j}, using the deep learning-based feature extraction architecture proposed in [12]. We do not use the provided pre-trained models and instead, fine-tune the existing feature extraction network on traffic datasets to learn meaningful features pertaining to traffic. We discuss the fine-tuned hyperparameters in the supplementary material.

Next, we predict the next state (state consists of spatial coordinates (p_i) and velocities (ν_i)) for each road-agent for the next time-step using SimCAI. This step is the main contribution of this work and is described in detail in Section 2.3.3. This step results in another set of segmented

boxes for each road-agent at time $t+1$.

Finally, we use these sets of segmented boxes to compute features using a Convolutional Neural Network [1]. The features generated are compared using association algorithms [52] to compute the ID of each agent in the next frame. The features are matched in two ways: the Cosine metric and the IoU overlap [53]. The Cosine metric is computed using the following optimization problem:

$$\min_{h_j}(l(f_{p_i}, f_{h_j})|p_i \in \mathcal{P}, h_j \in \mathcal{H}_i). \tag{2.1}$$

where $\mathcal{H}_i$ is the subset of all detected road-agents in the current frame that are within a circular region around agent p_i that have not been matched to a predicted agent. The IoU overlap metric is used in conjunction with the cosine metric. This metric builds a cost matrix Σ to measure the amount of overlap of each predicted bounding box with all nearby detection bounding box candidates. $\Sigma(i,j)$ stores the IoU overlap of the bounding box of p_i with that of h_j and is calculated as:

$$\Sigma(i,j) = \frac{\mathbb{B}_{p_i} \cap \mathbb{B}_{h_j}}{\mathbb{B}_{p_i} \cup \mathbb{B}_{h_j}}, h_j \in \mathcal{H}_i.$$

If we denote the cosine and the IOU overlap metrics by C and I, respectively, then the combined cost function value is obtained through,

$$\text{Combined Cost} = \lambda_1 C + \lambda_2 I, \lambda_1 + \lambda_2 = 1, \tag{2.2}$$

where λ_1, λ_2 are constants representing the weights for the individual metric costs. Matching a detection to a predicted measurement with maximum overlap thus becomes a max-weight

matching problem and we solve it efficiently using the Hungarian algorithm [52]. The ID of the road-agent at time t is assigned to that road-agent at time $t+1$ whose appearance is most closely associated to the road-agent at time t.

2.3.2 Reduced Probability of Track Loss

We now show how sparse feature vectors generated from segmented boxes reduce the probability of the loss of pedestrian tracks (false negatives).

We define $\mathcal{T}_t = \{\Psi_{1:t}\}$ to be the set of positively identified states for p_i until time t. We denote the time since the last update to a track ID as μ. We denote the ID of p_i as α and we represent the correct assignment of an ID to p_i as $\Gamma(\alpha)$. The threshold for the Cosine metric is $\lambda \underset{\text{i.i.d.}}{\sim} \mathbb{U}[0,1]$. The threshold for the track age, *i.e.*, the number of frames before which track is destroyed, is ξ. We denote the probability of an event that uses Mask R-CNN as the primary object detection algorithm with $\mathbb{P}^M(\cdot)$ and the probability of an event that uses a standard Faster R-CNN [20] as the primary object detection algorithm (*i.e.*, outputs bounding boxes without boundary subtraction) with $\mathbb{P}^F(\cdot)$. Finally, $\mathcal{T}_t \leftarrow \{\emptyset\}$ represents the loss of $\mathcal{T}_t$ by occlusion.

We now state and prove the following lemma.

Lemma 2.3.1. *For every pair of feature vectors $(f^M_{h_j}, f^F_{h_j})$ generated from a segmented box and a bounding box respectively, if $\|f^M_{h_j}\|_0 > \|f^F_{h_j}\|_0$, then $d(f, f^M_{h_j}) < d(f, f^F_{h_j})$ with probability $1 - \frac{B}{A}$, where A and B are positive integers and $A > B$.*

Proof. Using the definition of the Cosine metric, the lemma reduces to proving the following, We pad both $f^M_{h_j}$ and $f^F_{h_j}$ such that $\|f^M_{h_j}\| > ||f^F_{h_j}||_0$.

We reduce f_t, $f^M_{h_j}$, and $f^F_{h_j}$ to binary vectors, *i.e.*, vectors composed of 0s and 1s. Let

$\Delta f = f_{h_j}^M - f_{h_j}^F$. We denote the number of 1s and -1s in Δf as A and B, respectively. Now, let x and y denote the L_0 norm of $f_{h_j}^M$ and $f_{h_j}^F$, respectively. From our padding procedure, we have $x > y$. Then, if $y = B$, then $x = A$ and we trivially have $A > B$. But if $y > B$, then $A = x - (y - B) \implies A - B = x - y$. From $x > y$, it again follows that $A > B$. Thus, $x > y \implies A > B$.

Next, we define a $(1, 1)$ coordinate in an ordered pair of vectors as the coordinate where both vectors contain 1s. Similarly, a $(1, -1)$ coordinate in an ordered pair of vectors is the coordinate where the first vector contains 1 and the second vector contains -1. Then, let p_a and p_b respectively denote the number of $(1, 1)$ coordinates and $(1, -1)$ coordinates in the pair $(f_t, \Delta f)$. By definition, we have $0 < p_a < A$ and $0 < p_b < B$. Thus, if we assume p_a and p_b to be uniformly distributed, it directly follows that $\mathbb{P}(p_a > p_b) = 1 - \frac{B}{A}$.

□

Based on Lemma 2.3.1, we finally prove the following proposition.

Proposition 2.3.1. *With probability $1 - \frac{B}{A}$, sparse feature vectors extracted from segmented boxes decrease the loss of pedestrian tracks, thereby reducing the number of false negatives in comparison to regular bounding boxes.*

Proof. In our approach, we use Mask R-CNN for pedestrian detection, which outputs bounding boxes and their corresponding masks. We use the mask and bounding box pair to generate a segmented box. The correct assignment of an ID depends on successful feature matching between the predicted measurement feature and the optimal detection feature, that is, when the cosine cost between the two features is below a set threshold, λ. But since the correct ID assignment depends on multiple factors including a low cosine cost, we instead exploit the equivalency of the contra-

positive,

$$d(\mathbf{f}, f_{h_j^*}) > \lambda \Leftrightarrow (\alpha = \emptyset) \tag{2.3}$$

Equation 2.3 indicates that when the cosine cost is greater than λ, then feature matching fails and therefore an ID fails to be assigned, or equivalently, set to $\emptyset$. Using Lemma 2.3.1 and the fact that $\lambda \underset{i.i.d.}{\sim} \mathbb{U}[0, 1]$,

$$\mathbb{P}(d(\mathbf{f}, f^M_{h^*_{j,p_i}}) > \lambda) < \mathbb{P}(d(\mathbf{f}, f^F_{h^*_{j,p_i}}) > \lambda)$$

Using the equivalency in Eq. 2.3, we obtain,

$$\mathbb{P}^M(\alpha = \emptyset) < \mathbb{P}^F(\alpha = \emptyset) \tag{2.4}$$

Now, in our approach, we set certain fixed conditions that need to be satisfied for a track to be destroyed or lost. Those conditions are listed as follows:

1. DensePeds updates the ID of every pedestrian after each frame. If an update fails to occur for $\mu > \zeta$ frames, then this leads to loss of the track of that pedestrian.

2. If the first condition is satisfied, and the current ID of a pedestrian is not set, then the track of that pedestrian is lost.

We formalize the two conditions as follows,

$$(\mu > \xi) \wedge (\alpha = \emptyset) \Leftrightarrow \mathcal{T}_t \leftarrow \{\emptyset\}$$

Using Eq. 2.4, it follows that,

$$\mathbb{P}^{M}(\mathcal{T}_t \leftarrow \{\emptyset\}) < \mathbb{P}^{F}(\mathcal{T}_t \leftarrow \{\emptyset\}) \tag{2.5}$$

Informally, equation 2.5 tells us that the probability of losing a track using segmented boxes is less than the probability of losing a track if we were to use regular bounding boxes.

To complete the proof, we now show that equation 2.5 implies that fewer lost tracks leads to fewer false negatives. We start by defining the total number of false negatives (FN) as

$$FN = \sum_{t=1}^{T} \sum_{p_g \in \mathcal{G}} \delta_{\mathcal{T}_t} \tag{2.6}$$

where $p_g \in \mathcal{G}$ denotes a ground truth pedestrian in the set of all ground truth pedestrians at current time t and $\delta_z = 1$ for $z = 0$ and 0 elsewhere. This is a variation of the Kronecker delta function. Using Eq. 2.5 and Eq. 2.6, we can say that fewer lost tracks ($\mathcal{T}_t \leftarrow \{\emptyset\}$) indicate a smaller number of false negatives. We empirically demonstrate this analysis in Section 2.3.4.

□

The upper bound, $\mathbb{P}^{F}(\mathcal{T}_t)$, in Eq. 2.5 depends on the amount of padding done to $\mathbf{f}$ and $\mathbf{f}$. A general observed trend is that a higher amount of padding results in a larger upper bound in Eq. 2.5.

2.3.3 Simultaneous Collision Avoidance and Interactions

One of the major challenges with tracking heterogeneous road-agents in dense traffic is that road-agents such as cars, buses, bicycles, road-agents, etc. have different sizes, geometric shape,

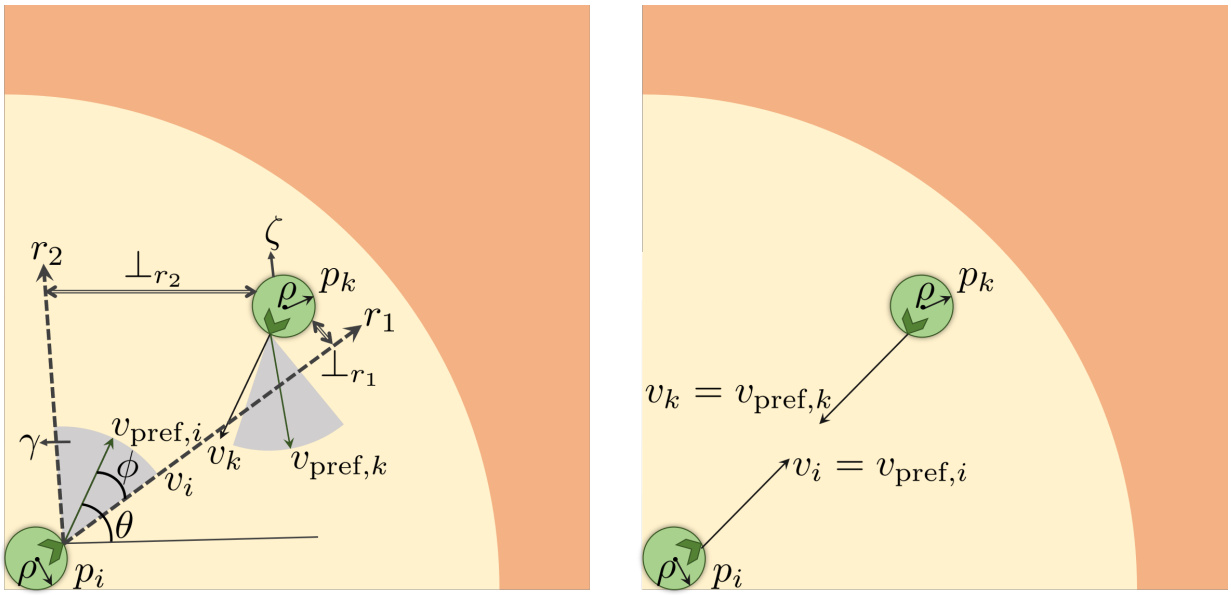


Figure 2.3: Inner yellow circle denotes the social distance and the outer orange area denotes the public region. At time $t \geq \tau$, p_i intends to interact with p_k. Then, (left) p_i determines its ability to interact with p_k. We observe that γ (grey cone) of p_i contains ζ of p_k (green circle around p_k). Thus p_i can interact with p_k. (right) p_i and p_k align their preferred velocities toward each other.

maneuverability, behavior, and dynamics. This often leads to complex inter-agent interactions that have not been taken into account by prior multi-object trackers. Furthermore, road-agents in high-density scenarios are in close-proximity to one another or are almost colliding. So we need an efficient approach for predicting the next state of a road-agent by modeling the collisions and interactions. We thus present SimCAI, that takes into account both,

- Reciprocal collision avoidance [3] with car-like kinematic constraints for trajectory prediction and collision avoidance.
- Heterogeneous road-agent interaction between pedestrians, two-wheelers, rickshaws, buses, cars and so on.

All the notations used in the paper are provided in Table I of full version of this text [45].

2.3.3.1 Velocity Prediction by Modeling Collision Avoidance

Reciprocal Velocity Obstacles (RVO) [3] extends Velocity Obstacles motion model by modeling collision avoidance behavior for multiple engaging agents. RVO can be applied to pedestrians in a crowd and we modify it to work with bounding boxes as our algoritheorem conforms to the tracking-by-detection paradigm.

We represent each agent as, $\Psi_t = [u, v, \dot{u}, \dot{v}, v_{\text{pref}}]$, where $u, v, \dot{u}, \dot{v}$, and v_{pref} represent the top left corner of the bounding box, their velocities, and the preferred velocity of the agent in the absence of obstacles respectively. v_{pref} is computed internally by RVO.

The computation of the new state, Ψ_{t+1}, is expressed as an optimization problem. For each agent, RVO computes a feasible region where it can move without collision. This region is defined according to the RVO collision avoidance constraints (or ORCA constraints [3]). If the ORCA constraints forbid an agent's preferred velocity, that agent chooses the velocity closest to its preferred velocity that lies in the feasible region, as given by the following optimization:

$$v_{\text{new}} = \underset{v \notin ORCA}{\arg\min} ||v - v_{\text{pref}}|| \tag{2.7}$$

The velocity, v_{new}, is then used to calculate the new position of a road-agent.

The difference in shapes, sizes, and aspect ratios of road-agents motivate the need to use appearance-based features. In order to combine object detection with RVO, we modify the state vector, Ψ_t, to include bounding box information by setting the position to the centers of the bounding boxes. Thus, $u = \frac{u+w}{2}$ and $v = \frac{v+h}{2}$, where w, h denote the width and height, respectively, of the corresponding bounding box.

Finally, the original RVO models the motion of agents seen from a top-view. Therefore, to account for front-view traffic as well as top-view, we use the modification proposed by the authors of [12] that allow RVO to model the motion of road-agents in front-view traffic scenes.

2.3.3.2 Velocity Prediction by Modeling Road-Agent Interactions

In a traffic scenario, interactions can occur between different types of road-agents: vehicle-vehicle, pedestrian-pedestrian, vehicle-pedestrian, bicycle-pedestrian, etc. In this section, we present a formulation to model such interactions. Our input is an RGB video captured from a camera with known camera parameters. By using the camera center as the origin, we transform pixel coordinates to scene coordinates for the computations that follow in this section.

Intent of Interaction The idea of using spatial regions to characterize agent behavior was proposed in [54]. The authors introduced the notion of "public" and "social" regions, that are of the form of concentric circles. We show a quadrant of these regions in Figure 2.3, where the yellow area is the social region and the orange area is the public region. Based on this work, Satake et al. [55] proposed a model of approach behavior with which a robot can interact with humans. At the public distance the robot is allowed to approach the human to interact with them, and at the social distance, interaction occurs. In SimCAI, we have set the public and social distances heuristically.

We say that a road-agent, p_i, intends to interact with another agent, p_k, when p_i is within the social distance of p_k for some minimum time τ. When two road-agents intend to interact, they move towards each other and come in close proximity.

Ability to Interact Even when two road-agents want to interact, their movements could be restricted in dense traffic. We determine the ability to interact (Figure 2.3(right)) as follows.

Each agent has a personal space, which we define as a circular region ζ of radius ρ, centered around p_k. Given a road-agent p_i, the slope of its v_{pref} is $\tan\theta$. θ is the angle with the horizontal defined in the world coordinate system. In dense traffic, each agent, p_i has a limited space in which they can steer, or turn. This space is the feasible region determined by the ORCA constraints described in the previous section. We define a 2D cone, γ, of angle ϕ as the ORCA region in which the agent can steer. ϕ is thus the steering angle of the agent. We denote the extreme rays of the cone as r_1 and r_2. $\perp_{\mathbb{G}_{\not\parallel}}^{\mathbb{G}_{\not\perp}}$ denotes the smallest perpendicular distance between any two geometric structures, say, $\mathbb{G}_{\not\parallel}$ and $\mathbb{G}_{\not\perp}$. These parameters are fixed for different agent types and are not learned from data.

If p_i has intended to interact with p_k, the projected cone of p_i, defined by extending r_1 and r_2, is directed towards p_k Then, in order for interaction to take place, it is sufficient to check for either one of two conditions to be true:

1. Condition Ω_1: Intersection of ζ with either r_1 or r_2 (if either ray intersects, then the entire cone intersects ζ).

2. Condition Ω_2: $\zeta \subset \gamma$ (if ζ lies in the interior of the cone, see Figure 2.3).

For these conditions to hold, we require that the cone does not intersect or contain any $p_j \in \mathcal{P}, j \neq i$. We now make these equations more explicit.

We parametrize r_1, r_2 by their slopes $\tan\delta$, where $\delta = \theta_i + \phi_i$ if $\perp_{\zeta}^{r_1} \geq \perp_{\zeta}^{r_2}$, else $\delta = \theta_i - \phi_i$.

The resulting equation of r_1 (or r_2) is $(Y - v_i) = \tan\delta(X - u_i)$ and the equation of ζ is $(X - u_k)^2 + (Y - v_k)^2 = \rho^2$. Solving both equations simultaneously, we obtain an equation

Ω_1. Intersection occurs if the discriminant of $\Omega_1 \geq 0$. This provides us with the first condition necessary for the occurrence of an interaction between p_i and p_k.

Next, we observe that if ζ lies in the interior of γ, then p_k lies on the opposite sides of r_1 and r_2 which is modeled by the following equation:

$$\Omega_2 \equiv r_1(p_k).r_2(p_k) \leq 0 \tag{2.8}$$

Solving Equation 2.8 further provides us with the second condition for the occurrence of an interaction between p_i and p_k, where $\Omega_1, \Omega_2 : \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R} \longmapsto \mathbb{R}$.

Interaction If either Ω_1 or Ω_2 is true, then road-agents p_i, p_k will move towards each other to interact at time $t \geq \tau$. When this happens, we assume that p_i and p_k align their current velocities towards each other. Thus, $v_{\text{new}} = v_{\text{pref}}$. The time taken for the two road-agents to be meet or converge with each other is given by $t = \dfrac{||p_i - p_k||_2}{||v_i - v_k||_2}$. If two road-agents are overlapping (based on the values of Ω_1 and Ω_2), we model them as a new agent with radius 2ϵ.

Our approach can be extended to model multiple interactions. Currently, we restrict an interaction to take place between 2 road-agents. Therefore, in the case of multiple possible interactions with an agent, p_k, we form a set $\mathcal{Q} \subseteq \mathcal{P}$, where $\mathcal{Q}$ is the set of all road-agents p_ω, that are intending to interact with p_k. We determine the road-agent that will interact with p_k as the road-agent that minimizes the distance between p_k and p_ω after a fixed time-step, Δt. Thus, $p_\omega = \arg\min_w \|(p_\omega + v_\omega \Delta t) - p_k\|, p_\omega \in \mathcal{Q}$. road-agents that are not interacting avoid each other and continue moving towards their destination.

2.3.3.3 Analysis

We analyze the accuracy and runtime performance of SimCAI in traffic scenarios with increasing density and heterogeneity.

Accuracy Analysis: We analytically show the advantage of SimCAI over other motion models such as Social Forces [2], RVO [3, 44], and constant velocity [1].

We denote the mutliple object tracking accuracy, $MOTA$ of a system using a particular motion model as $MOTA^{\text{model}}$ and define it as $MOTA^{\text{model}} = \sum_c MOTA_c + \sum_i MOTA_i$ where c and i denote an agent whose motion is being modeled using collision avoidance and interaction, and $MOTA_c$ and $MOTA_i$ denote their individual accuracies, respectively. Let n represent the number of total road-agents in a video, then we have $n = n_c + n_i$, where n_c, n_i correspond to the number of agents that are avoiding collisions and are interacting, respectively.

Increasing n would increase the number of road-agents whose motion is modeled through collision avoidance or heterogeneous interaction formulations. Linear models do not account for either formulation. Standard RVO only accounts for collision avoidance. SimCAI models both. Therefore, we rationalize that,

$$
\begin{aligned}
MOTA_c^{\text{linear}} &\leq MOTA_c^{\text{RVO}} \approx MOTA_c^{\text{SimCAI}} \\
MOTA_i^{\text{linear}} &\leq MOTA_i^{\text{RVO}} \leq MOTA_i^{\text{SimCAI}} \\
\implies MOTA^{\text{linear}} &\leq MOTA^{\text{RVO}} \leq MOTA^{\text{SimCAI}}
\end{aligned}
$$

We validate the analysis presented here in Section 2.3.4.

Runtime Analysis: At approximately 30 fps, we achieve a minimum speed-up of approximately 4×, and upto approximately 30×, over state-of-the-art methods on the MOT dataset (Table 2.3). The state-of-the-art use RNNs to model the motion of road-agents [56, 57], while we use the

modified RVO formulation. We exploit the geometrical formulation of SimCAI to state and prove the following theorem:

Theorem 2.3.1. *Given* $\mathcal{P} = \{p_i | 1 \leq i \leq n\}$*, that represents a set of* n *road-agents in a traffic scene that may assume any shape, size, and agent-type, if state*$_{p_i} \in \{$ stationary, collision avoiding, interacting $\}$*,* $\forall i \in n$*, then SimCAI can track the* n *road-agents in* $\mathcal{O}((n_c + \omega n_i))$*, where* $\omega << n_i$*.*

Proof. RVO is based on linear programming and can perform tracking with a proven runtime complexity of $\mathcal{O}((n))$ [3]. Now, if we assume that agents always assume one of the following states: stationary, avoiding collision, or interacting, then we have $n = n_c + n_i$, where n_c, n_i correspond to the number of agents in collision avoidance states and interacting states, respectively. We ignore stationary road-agents. Following the formulation in Section 2.3.3.2, for each interacting road-agent, SimCAI predicts a new velocity by solving a linear optimization problem over ω road-agents. Thus, the runtime complexity of SimCAI is $\mathcal{O}((n_c + \omega n_i))$, where $\omega << n_i$. □

Our high fps is a consequence of our linear runtime complexity and we validate our theoretical claims in Section 2.3.4. We further hypothesize that prior deep learning-based methods [56, 57] are less optimal in terms of runtime due to the intensive computation requirements by deep neural networks [58, 59]. For example, ResNet [60] needs more than 25 MB for storing the computed model in memory, and more than 4 billion float point operations (FLOPs) to process a single image of size 224×224 [58].

We would like to clarify that by realtime performance, we refer to the realtime computation of the tracking algorithm only. We do not consider the computation time of Mask R-CNN. This is standard practice by tracking-by-detection algorithms [57] that only contribute to the tracking

component, similar to this work. We therefore compare with realtime tracking algorithms.

2.3.4 Results

On Dense Datasets: We provide results on the TRAF dataset using RoadTrack and demonstrate a state-of-the-art average MOTA of 75.8% (Table 2.2). The aim of this experiment is to highlight the advantage of our overall tracking algorithm in dense and heterogeneous traffic. We compare RoadTrack with methods on the dense TRAF dataset in Table 2.2. MOTDT [4] and MDP [5] are the only state-of-the-art methods with available open-source code. All methods are evaluated using a common set of detections obtained using Mask R-CNN. Compared to these methods, we improve upon MOTA by 5.2% on absolute. This is roughly equivalent to a rank difference of 46 on the MOT benchmark.

MOTDT is currently the fastest method on the MOT16 benchmark. Our approach operates at realtime speeds upto approximately 30 fps and is comparable with MOTDT (Table 2.2). Our realtime performance results from the runtime analysis from Section 2.3.3.3 and theorem 2.3.1.

Note that we observe an abnormally high number of identity switches compared to other methods; however, this is because prior methods mostly fail to maintain an agent's track for more than 20% of their total visible time (near 100% ML). Not being able to track road-agents for most of the time excludes those agents as possible candidates for IDS, thereby resulting in lower IDS for prior methods. Interestingly, the low IDS score for prior methods also contributes to their reasonably high MOTA score, despite near-failure to track agents in dense traffic.

On Standard Benchmarks: In the interest of completeness and thorough evaluation, we also evaluate RoadTrack on sparser tracking datasets and present results on both traffic-

Motion Model	FPS↑	MT(%)↑	ML(%)↓	IDS↓	FN↓	MOTP(%)↑	MOTA(%)↑
Const. Vel	30	0.0	100	**11**	247,738(33.3&)	**66.3**	66.7
SF	30	0.1	98.6	147	246,528 (33.1%)	63.8	66.3
RVO	30	0.0	100	38	247,675 (33.2%)	63.8	66.9
SimCAI	**30**	**7.0**	**66.9**	1128	**178,997 (24.0%)**	65.7	**75.8**

Table 2.1: Ablation experiments to show the advantage of SimCAI. We replace SimCAI with a constant velocity (Const Lin Vel) [1], Social Forces (SF) [2], and RVO motion model (RVO)[3]. The rest of the method is identical to the original method. All variations operate at similar fps of approximately 30 fps. Bold is best. Arrows (↑, ↓) indicate the direction of better performance.

only datasets (KITTI-16) in Table 2.3 as well as datasets containing only pedestrians (MOT) in Table 2.4. RoadTrack's main advantage is SimCAI, which is based on modeling collision avoidance and interactions. In the absence of one or both, we do not expect it demonstrate superior performance over prior methods on the sparse KITTI-16 and MOT datasets.

While not conclusive, we believe our low MOTA score on the 2D MOT15 and KITTI-16 may also be attributed to a high number of detections that are incorrectly classified as false positives. For instance, road-agents that are too distant to be manually labeled are not annotated in the ground truth sequence. We observed this to be true for the methods we compared with as well. Therefore, we exclude FP from the calculation of MOTA for all methods in the interest of fair evaluation.

We note, however, that RoadTrack is least 4× faster on the KITTI-16 and 2D MOT15 datasets at approximately 30 fps (Tables 2.3,2.4). To explain the speed-up, we refer to theorem 2.3.1 and the runtime analysis presented in Section 2.3.3.3. We specially point to the 15× and 5× speed-up over learning-based tracking methods, [56, 57] in Table 2.3 which we attribute the linear time computation of SimCAI as opposed to the intensive computation required by deep learning models.

Ablation Experiments: We highlight the advantages of SimCAI through ablation experiments

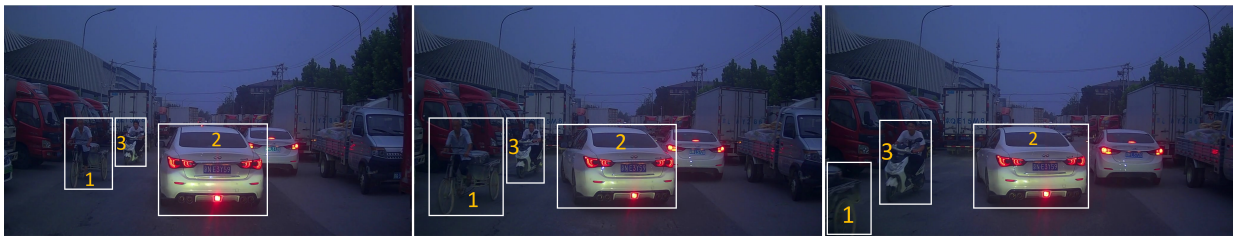


Figure 2.4: Qualitative analysis of RoadTrack on the TRAF dataset at night time consisting of cars, 2-wheelers, 3-wheelers, and trucks. Frames are chosen with a gap of 2 seconds($\sim$ 60 frames). For visual clarity, each road-agent is associated with a unique ID number. The ID is displayed in orange. Note the consistencies in the ID, for example, the 3-wheeler (1), car (2), and 2-wheeler (3).

in Table 2.1. The aim of these experiments is to isolate the benefit of SimCAI. We compare with the following variations of RoadTrack in which we replace our novel motion model SimCAI with standard and state-of-the-art motion models, while keeping the rest of the system untouched:

- Constant Linear Velocity (Const Lin Vel). We replace SimCAI with a constant velocity linear motion model [1].
- Social Forces (SF). We replace SimCAI with the Social Forces motion model [2].
- Reciprocal Velocity Obstacles (RVO) [3]. We replace SimCAI with the RVO motion model.

We compare SimCAI with other motion models (Constant linear velocity, Social Forces, and RVO) on the dense TRAF dataset. These experiments were performed by *only* replacing SimCAI with other motion models, keeping the rest of the system unchanged. We observe that SimCAI outperforms the motion models by at least 8.9% on absolute on MOTA. All the variations used in the ablation experiments operated at the same fps of approximately 30 fps. Additionally, we experimentally verify the analysis of Section 2.3.3.3 by observing that $MOTA^{\text{linear}} \leq MOTA^{\text{RVO}} \leq$

$MOTA^{\text{SimCAI}}$. Once again, we point to our high IDS in Table 2.1, compared to the IDS of other motion models. As mentioned previously, this is due to the near-failure of other motion models (near 100% ML) to track road agents in dense traffic. Not being able to track a road-agent excludes them as a IDS candidate.

Dataset	Tracker	FPS↑	MT(%)↑	ML(%)↓	IDS↓	FN↓	MOTP(%)↑	MOTA(%)↑
TRAF1	MOTDT	37.9	0	98.2	**15 (<0.1%)**	18,764 (33.0%)	**63.3**	67.0
	MDP	9.3	0	98.2	21 (<0.1%)	18,667 (32.8%)	60.1	67.1
	RoadTrack	**43.9**	0	**95.6**	163 (0.3%)	**17,953 (31.6%)**	58.8	**68.1**
TRAF2	MOTDT	**41.6**	0	98.8	17 (<0.1%)	18,201 (32.7%)	60.3	67.3
	MDP	20.9	0	100.0	**7 (<0.1%)**	18,105 (32.5%)	59.6	67.5
	RoadTrack	12.3	0	**92.3**	55 (0.1%)	**17,202 (30.9%)**	**60.8**	**69.0**
TRAF3	MOTDT	50.7	3.3	67.1	64 (<0.1%)	34,883 (27.0%)	69.6	72.9
	MDP	**51.8**	0	100.0	**0 (0.0%)**	43,057 (33.3%)	69.2	66.7
	RoadTrack	36.6	**32.2**	**40.0**	62 (<0.1%)	**19,521 (15.1%)**	**70.1**	**84.8**
TRAF4	MOTDT	36.6	1.2	76.3	123 (0.1%)	54,849 (29.0%)	65.3	70.9
	MDP	9.0	1.2	87.2	**16 (<0.1%)**	59,097 (31.3%)	**66.2**	68.7
	RoadTrack	**40.6**	**6.0**	**54.6**	266 (0.1%)	**47,444 (25.1%)**	65.1	**74.7**
TRAF5	MOTDT	36.0	0.7	75.9	221 (0.2%)	33,774 (28.9%)	63.2	70.9
	MDP	22.5	0	98.4	**6 (<0.1%)**	38,091 (32.6%)	**64.9**	67.3
	RoadTrack	**41.4**	**1.5**	**55.7**	299 (0.3%)	**24,860 (21.3%)**	63.1	**78.4**
TRAF6	MOTDT	**33.0**	0	87.5	161 (0.1%)	58,212 (29.4%)	63.3	70.5
	MDP	4.3	0	99.3	**0 (0.0%)**	65,687 (33.2%)	**68.6**	66.8
	RoadTrack	14.6	**0.7**	**67.8**	283 (0.1%)	**52,017 (26.3%)**	62.8	**73.6**
Summary	MOTDT	**34.7**	0.9	83.6	601 (0.1%)	218,683 (29.3%)	65.5	70.6
	MDP	10.1	0.2	97.0	**50 (<0.1%)**	242,704 (32.6%)	65.3	67.4
	RoadTrack	31.6	**7.0**	**66.9**	1128 (0.2%)	**178,997 (24.0%)**	**65.7**	**75.8**

Table 2.2: Evaluation on the TRAF dataset with MOTDT [4] and MDP [5]. MOTDT is currently the best *online* tracker on the MOT benchmark with open-sourced code. Bold is best. Arrows (↑ ,↓) indicate the direction of better performance. **Observation:** RoadTrack improves the accuracy (MOTA) over the state-of-the-art by 5.2% and precision (MOTP) by 0.2%.

2.4 Trajectory prediction in heterogeneous traffic

For effective planning and navigation in dense and heterogeneous environments, AVs must predict the trajectories of human drivers (Figure 2.5). However, there are several issues associated with such environments that make prediction challenging. Highly dense traffic corresponds to more frequent inter-agent interactions which are hard to model due to the inherent uncertainty

	Tracker	FPS↑	MT(%)↑	ML(%)↓	IDS↓	FN↓	MOTP(%)↑	MOTA(%)↑
KITTI-16	AP_HWDPL_p [61]	6.7	17.6	11.8	18	831	**72.6**	40.7
	RAR_15_pub [57]	5.4	0.0	17.6	18	809	70.9	41.2
	AMIR15 [56]	1.9	11.8	11.8	18	714	71.7	**50.4**
	HybridDAT [62]	4.6	5.9	17.6	**10**	706	**72.6**	46.3
	AM [63]	0.5	5.9	17.6	19	805	70.5	40.6
	RoadTrack	**28.9**	**29.4**	**11.7**	15	**668**	71.3	12.2

Table 2.3: Evaluation on the KITTI-16 dataset from the MOT benchmark with *online methods* that have an average rank higher than ours. RoadTrack is at least approximately 4× faster than prior methods. While we do not outperform on the MOTA metric, we still achieve the highest MT, ML, FN, and MOTP. We analyze our MOTA performance in Section 2.3.4. Bold is best. Arrows (↑, ↓) indicate the direction of better performance. The values for all methods correspond to the KITTI-16 sequence specifically, and not the entire 2D MOT15 dataset.

	Tracker	FPS↑	MT(%)↑	ML(%)↓	IDS↓	FN↓	MOTP(%)↑	MOTA(%)↑
2D MOT15	AMIR15 [56]	1.9	15.8	**26.8**	1026	29,397	71.7	37.6
	HybridDAT [62]	4.6	11.4	42.2	358	31,140	72.6	35.0
	AM [63]	0.5	11.4	43.4	**348**	34,848	70.5	34.3
	AP_HWDPL_p [61]	6.7	8.7	37.4	586	33,203	72.6	**38.5**
	RoadTrack	**28.9**	**18.6**	32.7	429	**27,499**	**75.6**	20.0
MOT16	EAMTT_pub [64]	11.8	7.9	49.1	965	102,452	75.1	38.8
	RAR16pub [57]	0.9	13.2	41.9	648	91,173	74.8	45.9
	STAM16 [63]	0.2	14.6	43.6	**473**	91,117	74.9	46.0
	MOTDT [4]	**20.6**	15.2	38.3	792	85,431	74.8	**47.6**
	AMIR [56]	1.0	14.0	41.6	774	92,856	**75.8**	47.2
	RoadTrack	18.8	**20.3**	**36.1**	722	**78,413**	75.5	40.9

Table 2.4: Evaluation on the full MOT benchmark. The full MOT dataset is sparse and is not a traffic-based dataset. RoadTrack is at least approximately 4× faster than previous methods. While we do not outperform on the MOTA metric, we still achieve the highest MT, ML (MOT16), FN, and MOTP(MOT15). We analyze our MOTA performance in Section 2.3.4. Bold is best. Arrows (↑, ↓) indicate the direction of better performance.

in human behavior. Moreover, prediction algorithms require huge datasets for training, the collection of which is a costly and time-consuming process. Finally, AVs must simultaneously perform low-level trajectory and high-level action prediction for real time navigation as opposed to current state-of-the-art which handles trajectory and action prediction independent of each other. I developed three algorithms that address the limitations described above. In the first

Figure 2.5: Trajectory prediction in dense, heterogeneous, and unstructured environments.

approach, **TraPHic** [65], the key aspect is to selectively focus attention on fewer agents. The algorithm consists of a novel attention mechanism that teaches the ego-vehicle to identify the agents that deserve more importance than others. For instance, a pedestrian in the way of the ego-vehicle requires more attention than, say, a parked car to the side. This work has been published in **CVPR'19**.

2.4.1 Overview

In this section, we give an overview of our prediction algorithm that uses weighted interactions. Our approach is designed for dense and heterogeneous traffic scenarios and is based on two observations. The first observation is based on the idea that road agents in such dense traffic do not react to every road agent around them; rather, they selectively focus attention on key interactions in a semi-elliptical region in the field of view, which we call the "horizon". For example, consider a motorcyclist who suddenly moves in front of a car and the neighborhood of the car consists of other road agents such as three-wheelers and pedestrians (Figure 2.6). The car

must prioritize the motorcyclist interaction over the other interactions to avoid a collision.

The second observation stems from the heterogeneity of different road agents such as cars, buses, rickshaws, pedestrians, bicycles, animals, etc. in the neighborhood of an road agent (Figure 2.6). For instance, the dynamic constraints of a bus-pedestrian interaction differs significantly from a pedestrian-pedestrian or even a car-pedestrian interaction due to the differences in road agent shapes, sizes, and maneuverability. To capture these heterogeneous road agent dynamics, we embed these properties into the state-space representation of the road agents and feed them into our hybrid network. We also implicitly model the behaviors of the road agents. Behavior in our case the different driving and walking styles of different drivers and pedestrians. Some are more aggressive while others more conservative. We model these behaviors as they directly influence the outcome of various interactions, thereby affecting the road agents' navigation.

Given a set of N road agents $\mathcal{A} = \{a_i\}_{i=1\ldots N}$, trajectory history of each road agent a_i over t frames, denoted $\Psi_{i,t} := [(x_{i,1}, y_{i,1}), \ldots, (x_{i,t}, y_{i,t})]^\top$, and the road agent's size l_i, we predict the spatial coordinates of that road agent for the next τ frames. In addition, we introduce a feature called traffic concentration c, motivated by traffic flow theory. Traffic concentration, $c(x, y)$, at the location (x, y) is defined as the number of road agents between (x, y) and $(x, y) + (\delta x, \delta y)$ for some predefined $(\delta x, \delta y) > 0$. This metric is similar to traffic density, but the key difference is that traffic density is a macroscopic property of a traffic video, whereas traffic concentration is a mesoscopic property and is locally defined at a particular location. So we achieve a representation of traffic on several scales.

Finally, we define the state space of each road agent a_i as

$$\Omega_i := \begin{bmatrix} \Psi_{i,t} & \Delta\Psi_{i,t} & c_i & l_i \end{bmatrix}^\top \tag{2.9}$$

where Δ is a derivative operator that is used to compute the velocity of the road agent, and $c_i := [c(x_{i,1}, y_{i,1}), \ldots, c(x_{i,t}, y_{i,t})]^\top$.

2D Image Space to 3D World Coordinate Space: We compute camera parameters from given videos using standard techniques, and use the parameters to estimate the camera homography matrices. The homography matrices are subsequently used to convert the location of road agents in 2D pixels to 3D world coordinates w.r.t. a predetermined frame of reference, similar to approaches in [9, 66]. All state-space representations are subsequently converted to the 3D world space.

Horizon and Neighborhood Agents: Prior trajectory prediction methods have collected neighborhood information using lanes and rectangular grids [67]. Our approach is more generalized in that we pre-process the trajectory data by assuming a lack of lane information. This assumption is especially true in practice in dense and heterogeneous traffic conditions. We formulate a road agent a_i's neighborhood, N_i, using an elliptical region and selecting a fixed number of closest road agents using the nearest-neighbor search algorithm in that region. Similarly, we define the horizon of that agent, H_i, by selecting a smaller threshold in the nearest-neighbor search algorithm, and in a semi-elliptical region in front of a_i.

2.5 Related Work

In this section, we give a brief overview of some important classical prediction algorithms and recent techniques based on deep neural networks.

2.5.1 Prediction Algorithms and Interactions

Trajectory prediction has been researched extensively. Approaches include the Bayesian formulation, the Monte Carlo simulation, Hidden Markov Models (HMMs), and Kalman Filters.

Methods that do not model road-agent interactions are regarded as sub-optimal or as less accurate than methods that model the interactions between road agents in the scene. Examples of methods that explicitly model road-agent interaction include techniques based on social forces, velocity obstacles [3], LTA, etc. Many of these models were designed to account for interactions between pedestrians in a crowd (*i.e.* homogeneous interactions) and improve the prediction accuracy [30]. Techniques based on velocity obstacles have been extended using kinematic constraints to model the interactions between heterogeneous road agents. Our learning approach does not use any explicit pairwise motion model. Rather, we model the heterogeneous interactions between road agents implicitly.

2.5.2 Deep-Learning Based Methods

Approaches based on deep neural networks use variants of Recurrent Neural Networks (RNNs) for sequence modeling. These have been extended to hybrid networks by combining RNNs with other deep learning architectures for motion prediction.

RNN-Based Methods RNNs are natural generalizations of feedforward neural networks to sequence [68].

The benefits of RNNs for sequence modeling makes them a reasonable choice for traffic prediction. Since RNNs are incapable of modeling long-term sequences, many traffic trajectory prediction methods use long short-term memory networks (LSTMs) to model road-agent interactions. These include algorithms to predict trajectories in traffic scenarios with few heterogeneous interactions [67]. These techniques have also been used for trajectory prediction for pedestrians in a crowd [66].

Hybrid Methods Deep-learning-based hybrid methods consist of networks that integrate two or more deep learning architectures. Some examples of deep learning architectures include CNNs, GANs, VAEs, and LSTMs. Each architecture has its own advantages and, for many tasks, the advantages of individual architectures can be combined. There is considerable work on the development of hybrid networks. Generative models have been successfully used for tasks such as super resolution, image-to-image translation, and image synthesis. However, their application in trajectory prediction has been limited because back-propagation during training is non-trivial. In spite of this, generative models such as VAEs and GANs have been used for trajectory prediction of pedestrians in a crowd [9] and in sparse traffic [69]. Alternatively, Convolutional Neural Networks (CNNs or ConvNets) have also been successfully used in many computer vision applications like object recognition. Recently, they have also been used for traffic trajectory prediction [70, 71]. In this paper, we present a new hybrid network that combines LSTMs with CNNs for traffic prediction.

2.5.3 Traffic Datasets

There are several datasets corresponding to traffic scenarios. ApolloScape [72] is a large-scale dataset of street views that contain scenes with higher complexities, 2D/3D annotations

and pose information, lane markings and video frames. However, this dataset does not provide trajectory information. The NGSIM simulation dataset [73] consists of trajectory data for road agents corresponding to cars and trucks, but the traffic scenes are limited to highways with fixed-lane traffic. KITTI [74] dataset has been used in different computer vision applications such as stereo, optical flow, 2D/3D object detection, and tracking. There are some pedestrian trajectory datasets like ETH and UCY, but they are limited to pedestrians in a crowd. Our new dataset, TRAF, corresponds to dense and heterogeneous traffic captured from Asian cities and includes 2D/3D trajectory information.

2.5.4 Advanced Driver Assistance Systems (ADAS)

Passive safety measures (that do not process sensory information) in vehicles include safety belts, brakes, airbags etc. ADAS are active safety measures that collect and process sensory information through sensors such as lidars, radars, stereo cameras, and RGB cameras. Various ADAS process the input information in different ways to implement actions that assist the driver and prevent or reduce the likelihood of traffic accidents due to human error. The development of ADAS began with the Anti-Lock Braking System (ABS) introduced into production in the late 1970s.

As ADAS with various functionality become popular, it is not uncommon for multiple systems to be installed on a vehicle. If each function uses its own sensors and processing unit, it will make installation difficult and raise the cost of the vehicle. As a countermeasure, research integrating multiple functions into a single system has been pursued and is expected to make installation easier, decrease power consumption, and vehicle pricing. RobustTP contributes

towards this research effort by integrating realtime tracking with trajectory prediction.

In addition to to trajectory prediction applications, several other interesting ADAS are currently being used in vehicles on the road. For example, the Adaptive Cruise Control (ACC) automatically adapts speed to maintain a safe distance from vehicles in front. The Blind Spot Detection (BSD) helps drivers when they pull out in order to overtake another road-agent. Emergency Brake Assist (EBA) ensures optimum braking by detecting critical traffic situations. When EBA detects an impending collision, the braking system is put on emergency standby. Intelligent Headlamp Control (IHC) provides optimal night vision. The headlamps are set to provide optimum lighting via a continuous change of the high and low beams of the lights.

2.5.5 Road-Agent Behavior Prediction

Current autonomous vehicles lack social awareness due to their inherent conservative behavior. Overly conservative behavior present new risks in terms of low efficiency and uncomfortable traveling experiences. Real-world examples of problems caused by AVs that are not socially adaptable can be seen in this video[1]. The notion of using driver behavior prediction to make the AVs socially aware is receiving attention [75].

Current driving behavior modeling methods are limited to traffic psychology studies where predictions for driving behavior are made offline, based on either driver responses to questionnaires or data collected over a period of time. Such approaches are not suitable for online behavior prediction. In contrast, our behavior prediction algorithm is the first computationally online approach that does not depend on offline data and manually tunable parameters. In the remainder of this section, we review some of the prior behavior modeling approaches and conclude by

[1] https://www.youtube.com/watch?v=Rm8aPR0aMDE

pointing out the advantages of our approach.

Many studies have been performed behavior modeling by identifying factors that contribute to different driver behaviors classes such as aggressive, conservative, or moderate driving. These factors can be broadly categorized into four categories. The first category of factors that indicate road-agent behavior is driver-related. These include characteristics of drivers such as age, gender, blood pressure, personality, occupation, hearing, and so on [76, 77, 78]. Feng et al. [76] proposed five driver characteristics (age, gender, personality via blood test, and education level). Rong et al. [79] presented a similar study but instead used different features such as blood pressure, hearing, and driving experience to conclude that aggressive drivers tailgate and weave in and out of traffic. Dahlen et al. [80] studied the relationships between driver personality and aggressive driving using the five-factor-model [77]. Social Psychology studies [78, 81] have examined the aggressiveness according to the background of the driver, including age, gender, violation records, power of cars, occupation, etc.

The second category corresponds to environmental factors such as weather or traffic conditions [82, 83]. The study conducted in [83] was designed to investigate the effects of weather-controlled speed limits and signs for slippery road conditions on driver behavior, while other studies [82] correlated changes in traffic density with varying driver behavior.

The third category refers to psychological aspects that affect driving styles. These could include drunk driving, driving under the influence, state of fatigue, and so on [84, 85]. It is shown in [84] that driving under influence induces delayed responses in acceleration and deceleration. Jackson et al. show that a state of fatigue manifests the same characteristics as driving under influence, but without the effect of substance intoxication. Additionally, this category also includes distractions caused by driver activity during driving, such as operating

mobile phones. For example, [85] shows that drivers engaged in mobile phone conversations increase their response time to external stimuli.

The final category of factors contributing to driving behavior are vehicular factors such as positions, acceleration, speed, throttle responses, steering wheel measurements, lane changes, and brake pressure [15, 86, 87, 88, 89].

A recent data-driven behavior prediction approach [15] also models traffic through graphs. The method predicts the driving behavior by training a neural network on the eigenvectors of the DGGs using supervised machine learning. Apart from behavior modeling, several methods have used machine learning to predict the intent of road-agents [90, 91]. The proposed behavior prediction algorithm in this paper extends the approach in [15] by predicting sequences of eigenvectors for future time-steps. Compared to these prior methods, are algorithm is online, computationally tractable and does not depend on any other information other than the vehicle coordinates.

Aljaafreh et al. [86] categorized driving behaviors into four classes: Below normal, Normal, Aggressive, and Very aggressive, in accordance with acceleration data. Murphey et al. [87] conducted an analysis on the aggressiveness of drivers and observed that longitudinal (changing lanes) jerk is more related to aggressiveness than progressive (along the lane) jerk (i.e., rate of change in acceleration). Mohamad et al. [88] detected abnormal driving styles using speed, acceleration, and steering wheel movement, which indicated the direction of vehicles. Qi et al. [92] studied driving styles with respect to speed and acceleration. Shi et al. [93] pointed out that deceleration is not very indicative of the aggressiveness of drivers, but measurements of throttle opening, which are associated with acceleration, were more helpful in identifying aggressive drivers. Wang et al. [94] classified drivers into two categories, aggressive and normal, using speed and throttle opening captured by a simulator. Cheung et al. [89] use speed, acceleration,

lane change information of highway data to derive a linear mapping between vehicular information and driver behavior. Finally, using only the spatial positions of the vehicles, [15] uses spectral graph theory to train a multi-layer perceptron to classify driver behavior.

2.5.6 Traffic Flow and Forecasting

Traffic forecasting has been studied in different contexts in prior literature. From a deep-learning perspective, traffic forecasting is synonymous with trajectory prediction and does not take into account road-agent behavior [11]. However, in a broader sense, traffic forecasting refers to predicting traffic flow [95, 96, 97] or traffic density [98, 99, 100, 101] on a macroscopic scale. Predicting traffic flow is important for applications such as congestion management and vehicle routing. In this paper, we mainly limit ourselves to forecasting low-level trajectories and high-level behaviors of each road-agent.

2.5.7 Hybrid Architecture for Traffic Prediction

In this section, we present our novel network architecture for performing trajectory prediction in dense and heterogeneous environments. In the context of heterogeneous traffic, the goal is to predict trajectories, *i.e.* temporal sequences of spatial coordinates of a road agent. Temporal sequence prediction requires models that can capture temporal dependencies in data, such as LSTMs. However, LSTMs cannot learn dependencies or relationships of various heterogeneous road agents because the parameters of each individual LSTM are independent of one another. In this regard, ConvNets have been used in computer vision applications with greater success because they can learn locally dependent features from images. Thus, in order to leverage the

benefits of both, we combine ConvNets with LSTMs to learn locally useful relationships, both in space and in time, between the heterogeneous road agents. We now describe our model to predict the trajectory for each road agent a_i. A visualization of the model is shown in Figure 2.7.

We start by computing H_i and N_i for the agent a_i. Next, we identify all road agents $a_j \in N_i \cup H_i$. Each a_j has an input state-space Ω_j that is used to create the embeddings e_j, using

$$e_j = \phi(W_l \Omega_i + b_l) \tag{2.10}$$

where W_l and b_l are conventional symbols denoting the weight matrix and bias vector respectively, of the layer l in the network, and ϕ is the non-linear activation on each node.

Our network consists of three layers. The horizon layer (top cyan layer in Figure 2.7) takes in the embedding of each road agent in H_i, and the neighbor layer (middle green layer in Figure 2.7) takes in the embedding of each road agent in N_i. The input embeddings in both these layers are passed through fully connected layers with ELU non-linearities, and then fed into single-layered LSTMs (yellow blocks in Figure 2.7). The outputs of the LSTMs in the two layers are hidden state vectors, $h_j(t)$, that are computed using

$$h_j(t) = \text{LSTM}(e_j, W_l, b_l, h_j^{t-1}) \tag{2.11}$$

where h_j^{t-1} refers to the corresponding road agent's hidden state vector from the previous time-step $t-1$. The hidden state vector of a road agent is a latent representation that contains temporally useful information. In the remainder of the text, we drop the parameter t for the sake of simplicity, *i.e.*, h_j is understood to mean $h_j(t)$ for any j.

The hidden vectors in the horizon layer are passed through an additional fully connected layer with ELU non-linearities. We denote the output of the fully connected layer as h_{jw}. All the h_{jw}'s in the horizon layer are then pooled together in a "horizon map". The hidden vectors in the neighbor layer are directly pooled together in a "neighbor map". These maps are further elaborated in Section 2.5.7.1. Both these maps are then passed through separate ConvNets in the two layers. The ConvNets in both the layers are comprised of two convolution operations followed by a max-pool operation. We denote the output feature vector from the ConvNet in the horizon layer as f_{hz}, and that from the ConvNet in the neighbor layer as f_{nb}.

Finally, the bottom-most layer corresponds to the ego agent a_i. Its input embedding, e_i, passes sequentially through a fully connected with ELU non-linearities, and a single-layered LSTM to compute its hidden vector, h_i. The feature vectors from the horizon and neighbor layers, f_{hz} and f_{nb}, are concatenated with h_i to generate a final vector encoding

$$z := \text{concat}(h_i, f_{\text{hz}}, f_{\text{nb}}) \tag{2.12}$$

Finally, the concatenated encoding z passes through an LSTM to compute the prediction for the next τ seconds.

2.5.7.1 Weighted Interactions

Our model is trained to learn weighted interactions in both the horizon and neighborhood layers. Specifically, it learns to assign appropriate weights to various pairwise interactions based on the shape, dynamic constraints and behaviors of the involved agents. The *horizon-based weighted interactions* takes into account the agents in the horizon of the ego agent, and learns the

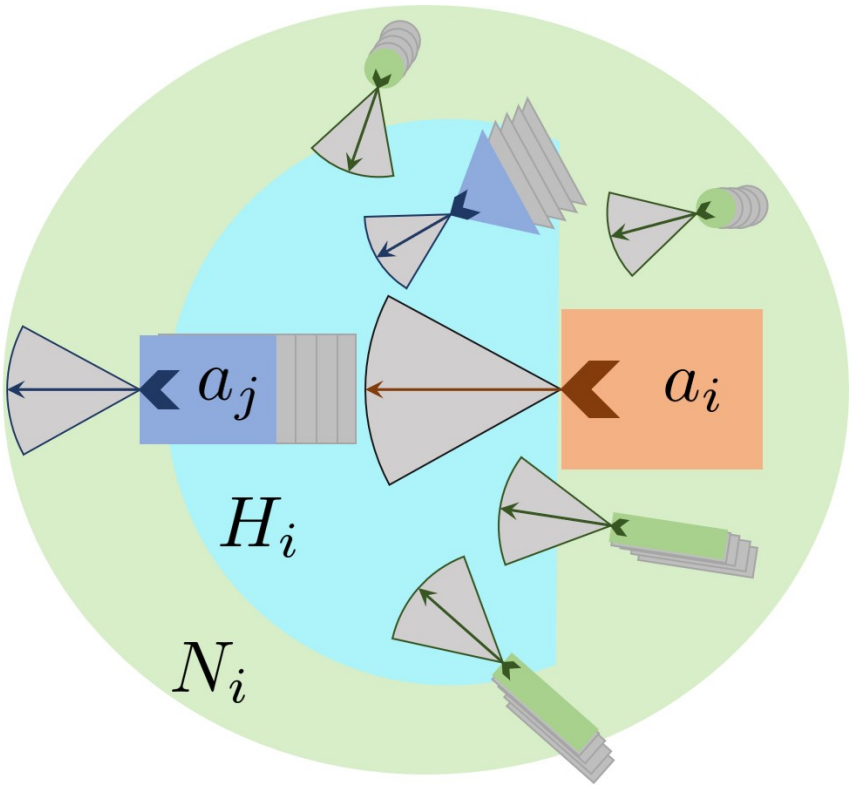


Figure 2.6: **Horizon and Heterogeneous Interactions:** We highlight various interactions for the red car. Horizon-based weighted interactions are in the blue region, containing a car and a rickshaw (both blue). The red car prioritizes the interaction with the blue car and the rickshaw (*i.e.* avoids a collision) over interactions with other road-agents. Heterogeneous-Based weighted interactions are in the green region, containing pedestrians and motorcycles (all in green). We model these interactions as well to improve the prediction accuracy.

"horizon map" $\mathcal{H}_i$, given as

$$\mathcal{H}_i = \{h_{jw} | a_j \in H_i\} \tag{2.13}$$

Similarly, the neighbor or *heterogeneous-based weighted interactions* accounts for all the agents in the neighborhood of the ego agent, and learns the "neighbor map" $\mathcal{N}_i$, given as

$$\mathcal{N}_i = \{h_j | a_j \in N_i\} \tag{2.14}$$

During training, back-propagation optimizes the weights corresponding to these maps by minimizing the loss between predicted output and ground truth labels. Our formulation results in higher weights for prioritized interactions (larger tensors in Horizon Map or blue vehicles in Figure 2.6) and lower weights for less relevant interactions (smaller tensors in Neighbor Map or green vehicles in Figure 2.6).

2.5.7.2 Implicit Constraints

Turning Radius: In addition to constraints such as position, velocity and shape, constraints such as the turning radius of a road agent also affects its maneuverability, especially as it interacts with other road agents within some distance. For example, a car (a non-holonomic agent) cannot alter its orientation in a short time frame to avoid collisions, whereas a bicycle or a pedestrian can.

However, the turning radius of a road agent can be determined by the dimensions of the road agent, *i.e.*, its length and width. Since we include these parameters into our state-space representation, we implicitly take into consideration each agent's turning radius constraints as well.

Driver Behavior: Velocity and acceleration (both relative and average) are clear indicators of driver aggressiveness. For instance, a road agent with a relative velocity (and/or acceleration) much higher than the average velocity (and/or acceleration) of all road agents in a given traffic scenario would be deemed as aggressive. Moreover, given the traffic concentrations at two consecutive spatial coordinates, $c(x, y)$ and $c(x+\delta x, y+\delta y)$, where $c(x, y) >> c(x+\delta x, y+\delta y)$, aggressive drivers move in a "greedy" fashion in an attempt to occupy the empty spots in the subsequent spatial locations. For each road agent, we compute its concentration with respect to

its neighborhood and add this value to its input state-space.

Finally, the relative distance of a road agent from its neighbors is another factor pertaining to how conservative or aggressive a driver is. More conservative drivers tend to maintain a healthy distance while aggressive drivers tend to tail-gate. Hence, we compute the spatial distance of each road agent in the neighborhood and encode this in its state-space representation.

2.5.7.3 Overall Trajectory Prediction

Our algorithm follows a well-known scheme for prediction [66]. We assume that the position of the road agent in the next frame follows a bi-variate Gaussian distribution with parameters $\mu_i^t, \sigma_i^t = [(\mu_x, \mu_y)_i^t, ((\sigma_x, \sigma_y)_i^t)]$, and correlation coefficient ρ_i^t. The spatial coordinates (x_i^t, y_i^t) are thus drawn from $\mathcal{N}(\mu_i^t, \sigma_i^t, \rho_i^t)$. We train the model by minimizing the negative log-likelihood loss function for the i^{th} road agent trajectory,

$$L_i = -\Sigma_{t+1}^{\tau} \log(\mathbf{P}((x_i^t, y_i^t)|(\mu_i^t, \sigma_i^t, \rho_i^t))). \tag{2.15}$$

We jointly back-propagate through all three layers of our network, optimizing the weights for the linear blocks, ConvNets, LSTMs, and Horizon and Neighbor Maps. The optimized parameters learned for the Linear-ELU block in the horizon layer indicates the priority for the interaction in the horizon of an road agent a_i.

2.5.8 Results

We describe our new dataset in Section 2.5.8. In Section 2.5.8, we list all implementation details used in our training process. Next, we list the evaluation metrics and methods that we

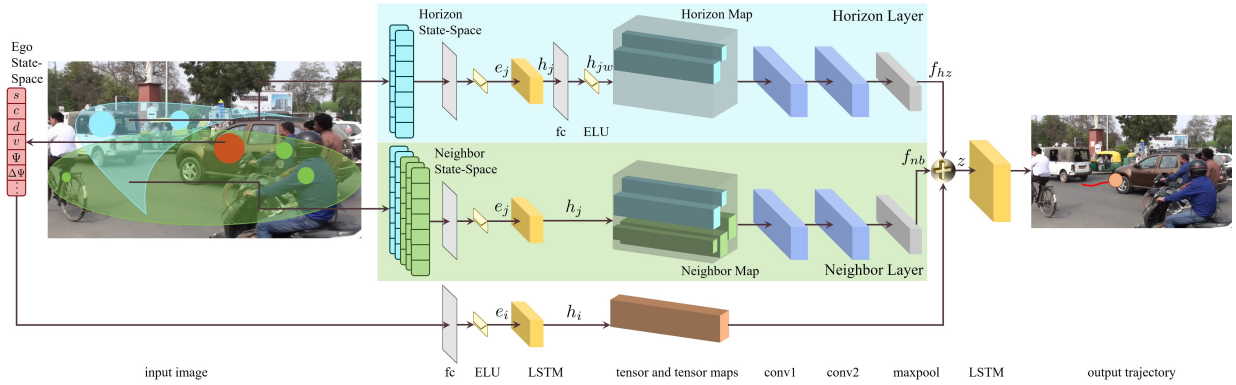


Figure 2.7: **TraPHic Network Architecture:** The ego agent is marked by the red dot. The green elliptical region around it is its neighborhood and the cyan semi-elliptical region in front of it is its horizon. We generate input embeddings for all agents based on trajectory information and heterogeneous dynamic constraints such as agent shape, velocity, and traffic concentration at the agent's spatial coordinates, and other parameters. These embeddings are passed through LSTMs and eventually used to construct the horizon map, the neighbor map and the ego agent's own tensor map. The horizon and neighbor maps are passed through separate ConvNets and then concatenated together with the ego agent tensor to produce latent representations. Finally, these latent representations are passed through an LSTM to generate a trajectory prediction for the ego agent.

compare with, in Section 2.5.8. Finally, we present the evaluation results in Section 2.5.8.

TRAF Dataset: Dense & Heterogeneous Urban Traffic We present a new dataset, currently comprising of 50 videos of dense and heterogeneous traffic. The dataset consists of the following road agent categories: car, bus, truck, rickshaw, pedestrian, scooter, motorcycle, and other road agents such as carts and animals. Overall, the dataset contains approximately 13 motorized vehicles, 5 pedestrians and 2 bicycles per frame. Annotations were performed following a strict protocol and each annotated video file consists of spatial coordinates, an agent ID, and an agent type. The dataset is categorized according to camera viewpoint (front-facing/top-view), motion (moving/static), time of day (day/evening/night), and difficulty level (sparse/moderate/heavy/challenge). All the videos have a resolution of 1280×720. We present a comparison of our dataset with

Dataset	Method				
	RNN-ED	S-LSTM	S-GAN	CS-LSTM	**TraPHic**
NGSIM	6.86/10.02	5.73/9.58	**5.16/9.42**	7.25/10.05	5.63/9.91
Beijing	2.24/8.25	6.70/8.08	4.02/7.30	2.44/8.63	**2.16/6.99**

Table 2.5: **Evaluation** on sparse or homogeneous traffic datasets: The first number is the average RMSE error (ADE) and the second number is final RMSE error (FDE) after 5 seconds (in meters). NGSIM is a standard sparse traffic dataset with few heterogeneous interactions. The Beijing dataset is dense but with relatively low heterogeneity. Lower value is better and bold value represents the most accurate result.

Methods Evaluated on TRAF										
RNN-ED	S-LSTM		S-GAN		CS-LSTM		**TraPHic**			
	Original	Learned	Original	Learned	Original	Learned	B	H_e	H_o	Combined
3.24/5.16	6.43/6.84	3.01/4.89	2.89/4.56	2.76/4.79	2.34/8.01	1.15/3.35	2.73/7.21	2.33/5.75	1.22/3.01	**0.78/2.44**

Table 2.6: **Evaluation** on our new, highly dense and heterogeneous TRAF dataset. The first number is the average RMSE error (ADE) and the second number is final RMSE error (FDE) after 5 seconds (in meters). The original setting for a method indicates that it was tested with default settings. The learned setting indicates that it was trained on our dataset for fair comparison. We present variations of our approach with each weighted interaction and demonstrate the contribution of the method. Lower is better and bold is best result.

Dataset	# Frames ($\times 10^3$)	Agents									Visibility (Km)	Density ($\times 10^3$)	#Diff Agents
		Ped	Bicycle	Car	Bike	Scooter	Bus	Truck	Rick	Total			
NGSIM	10.2	0	0	981.4	3.9	0	0	28.2	0	1013.5	0.548	1.85	3
Beijing	93	1.6	1.9	12.9						16.4	0.005	3.28	3
TRAF	12.4	4.9	1.5	3.6	1.43	5	0.15	0.2	3.1	19.88	0.005	3.97	8

Table 2.7: **Comparison** of our new TRAF dataset with various traffic datasets in terms of heterogeneity and density of traffic agents. Heterogeneity is described in terms of the number of different agents that appear in the overall dataset. Density is the total number of traffic agents per Km in the dataset. The value for each agent type under "Agents" corresponds to the average number of instances of that agent per frame of the dataset. It is computed by taking all the instances of that agent and dividing by the total number of frames. Visibility is a ballpark estimate of the length of road in meters that is visible from the camera. NGSIM data were collected using tower-mounted cameras (bird's eye view), whereas both Beijing and TRAF data presented here were collected with car-mounted cameras (frontal view).

standard traffic datasets in Table 2.7.

Implementation Details We use single-layer LSTMs as our encoders and decoders with hidden state dimensions of 64 and 128, respectively. Each ConvNet is implemented using two convolutional operations each followed by an ELU non-linearity and then max-pooling. We train the network

for 16 epochs using the Adam optimizer with a batch size of 128 and learning rate of 0.001. We use a radius of 2 meters to define the neighborhood and a minor axis length of 1.5 meters to define the horizon, respectively. Our approach uses 3 seconds of history and predicts spatial coordinates of the road agent for up to 5 seconds (4 seconds for KITTI dataset). We do not down-sample on the NGSIM dataset due to its sparsity. However, we use a down-sampling factor of 2 on the Beijing and TRAF datasets due to their high density. Our network is implemented in Pytorch using a single TiTan Xp GPU. Our network does not use batch norm or dropout as they can decrease accuracy. We include the experimental details involving batch norm and dropout in the appendix due to space limitations.

Evaluation Metrics and Comparison Methods We use the following commonly used metrics [9, 66, 67] to measure the performances of the algorithms used for predicting the trajectories of the road agents.

1. Average displacement error (ADE): The root mean square error (RMSE) of all the predicted positions and real positions during the prediction time.

2. Final displacement error (FDE): The RMSE distance between the final predicted positions at the end of the predicted trajectory and the corresponding true location.

We compare our approach with the following methods.

- RNN-ED (Seq2Seq): An RNN encoder-decoder model, which is widely used in motion and trajectory prediction for vehicles.

- Social-LSTM (S-LSTM): An LSTM-based network with social pooling of hidden states to predict pedestrian trajectories in crowds [66].

- Social-GAN (S-GAN): An LSTM-GAN hybrid network to predict trajectories for large human crowds [9].
- Convolutional-Social-LSTM (CS-LSTM): A variant of S-LSTM adding convolutions to the network in [66] in order to predict trajectories in sparse highway traffic [67].

We also perform ablation studies with the following four versions of our approach.

- TraPHic-B: A base version of our approach without using any weighted interactions.
- TraPHic-H_o: A version of our approach without using *Heterogeneous*-Based Weighted interactions, *i.e.*, we do not take into account driver behavior and information such as shape, relative velocity, and concentration.
- TraPHic-H_e: A version of our approach without using *Horizon*-Based Weighted interactions. In this case, we do not explicitly model the horizon, but account for heterogeneous interactions.
- TraPHic: Our main algorithm using both Heterogeneous-Based and Horizon-Based Weighted interactions. We explicitly model the horizon and implicitly account for dynamic constraints and driver behavior.

Results on Traffic Datasets In order to provide a comprehensive evaluation, we compare our method with state-of-the-art methods on several datasets. Table 2.5 shows the results on the standard NGSIM dataset and an additional dataset containing heterogeneous traffic of moderate density. We present results on our new TRAF dataset in Table 2.6.

TraPHic outperforms all prior methods we compared with on our TRAF dataset. For a fairer comparison, we trained these methods on our dataset before testing them on the dataset. However,

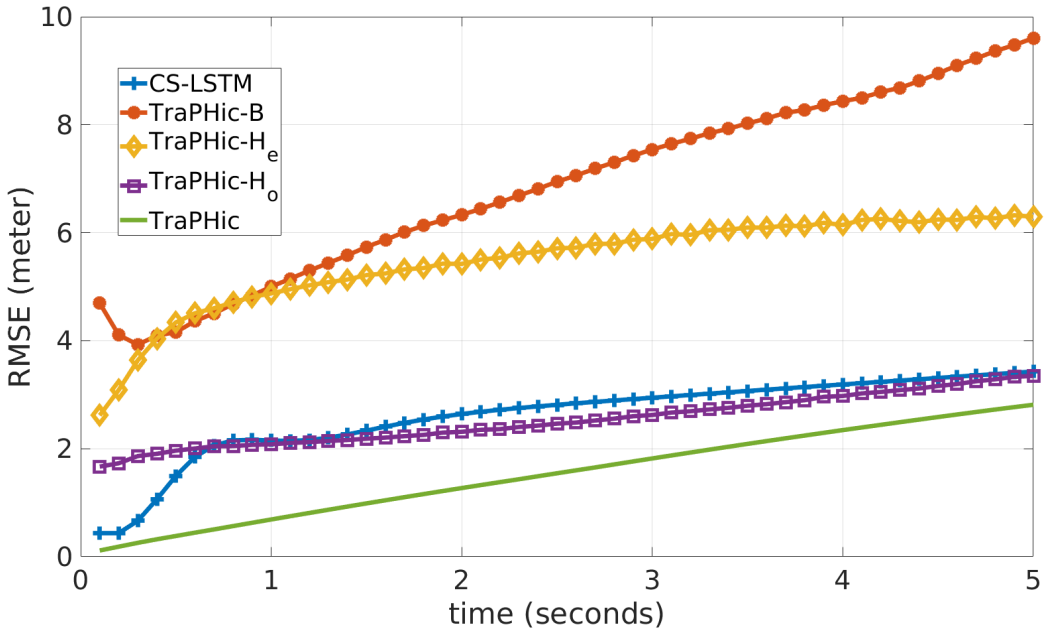


Figure 2.8: **RMSE Curve Plot:** We compare the accuracy of four variants of our algorithm with CS-LSTM and each other based on RMSE values on the TRAF dataset. On the average, using TraPHic-H_e reduces RMSE by 15% relative to TraPHic-B, and using TraPHic-H_o reduces RMSE by 55% relative to TraPHic-B. TraPHic, the combination of TraPhic-H_e and TraPhic-H_o, reduces RMSE by 36% relative to TraPHic-H_o, 66% relative to TraPHic-H_e, and 71% relative to TraPHic-B. Relative to CS-LSTM, TraPHic reduces RMSE by 30%.

the prior methods did not generalize well to dense and heterogeneous traffic videos. One possible explanation for this is that S-LSTM and S-GAN were designed to predict trajectories of humans in top-down crowd videos whereas the TRAF dataset consists of front-view heterogeneous traffic videos with high density. CS-LSTM uses lane information in its model and weight all agent interactions equally. Since the traffic in our dataset does not include the concept of lane-driving, we used the version of CS-LSTM that does not include lane information for a fairer comparison. However, it still led to a poor performance since CS-LSTM does not account for heterogeneous-based interactions. On the other hand, TraPHic considers both heterogeneous-based and horizon-based interactions, and thus produces superior performance on our dense and heterogeneous dataset.

Figure 2.9: **Trajectory Prediction Results:** We highlight the performance of various trajectory prediction methods on our TRAF dataset with different types of road-agents. We showcase six scenarios with different density, heterogeneity, camera position (fixed or moving), time of the day, and weather conditions. We highlight the predicted trajectories (over 5 seconds) of some of the road-agents in each scenario to avoid clutter. The ground truth (GT) trajectory is drawn as a solid green line, and our (TraPHic) prediction results are shown using a solid red line. The prediction results of other methods (RNN-ED, S-LSTM, S-GAN, CS-LSTM) are drawn with different dashed lines. TraPHic predictions are closest to GT in all the scenarios. We observe up to 30% improvement in accuracy over prior methods over this dense, heterogeneous traffic.

We visualize the performance of the various trajectory prediction methods on our TRAF dataset Figure 2.9. Compared to the prior methods, TraPHic produces the least deviation from the ground truth trajectory in all the scenarios. Due to the significantly high density and heterogeneity in these videos, coupled with the unpredictable nature of the involved agents, all the predictions deviate from the ground truth in the long term (after 5 seconds).

We demonstrate that our approach is comparable to prior methods on sparse datasets such as the NGSIM dataset. We do not outperform the current sate-of-the-art in such datasets, since our algorithm tries to account for heterogeneous agents and weighted interactions even when interactions are sparse and mostly homogeneous. Nevertheless, we are at par with the state-of-the-art performance. Lastly, we note that our RMSE value on the NGSIM dataset is quite high,

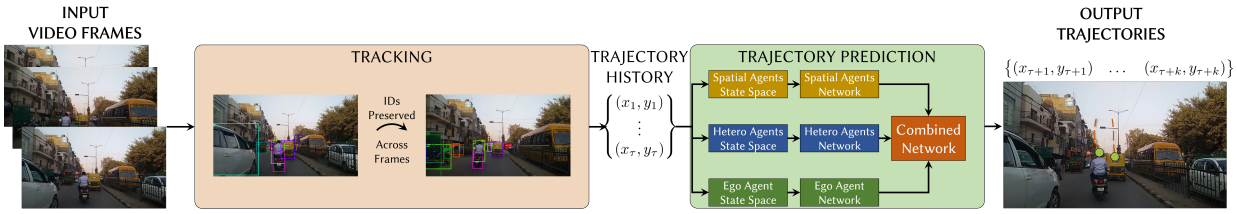


Figure 2.10: Overview of RobustTP: RobustTP is an end-to-end trajectory prediction algorithm that uses sensor input trajectories as training data instead of manually annotated trajectories. The sensor input is an RGB video from a moving or static camera. The first step is to compute trajectories using a tracking algorithm (light orange block). The trajectories generated are the training data for the trajectory prediction algorithm (green block). The model trains on $\tau = 3$ seconds of trajectory history and predicts trajectory for the next $k = 5$ seconds. As an example, the predicted trajectories for two of the agents are shown in the output image at the right end. The green circles denote the positions of the agents at the beginning of prediction, as seen from a top-view in the 3D world. The red-dashed lines denote the predicted trajectories for the next 5 seconds, as seen from the same top-view in the 3D world.

which we attribute to the fact that we used a much higher (2X) sampling rate for averaging than prior methods.

Finally, we perform an ablation study to highlight the contribution of our weighted interaction formulation. We compare the four versions of TraPHic as stated in Section 2.5.8. We find that the Horizon-based formulation contributes more significantly to higher accuracy. TraPHic-H_e reduces ADE by 15% and FDE by 20% over TraPHic-B, whereas TraPHic-H_o reduces ADE by 55% and FDE by 58% over TraPHic-B. Incorporating both formulations results in the highest accuracy, reducing the ADE by 71% and the FDE by 66% over TraPHic-B.

2.6 RobustTP: Improving robustness of prediction in unstructured traffic

We present an end-to-end algorithm for predicting future trajectories of road-agents in dense traffic with noisy sensor input trajectories obtained from RGB cameras (either static or moving) through a tracking algorithm. In this case, we consider noise as the deviation from the

ground truth trajectory. The amount of noise depends on the accuracy of the tracking algorithm. Our approach is designed for dense heterogeneous traffic, where the road agents corresponding to a mixture of buses, cars, scooters, bicycles, or pedestrians. is an approach that first computes trajectories using a combination of a non-linear motion model and a deep learning-based instance segmentation algorithm. Next, these noisy trajectories are trained using an LSTM-CNN neural network architecture that models the interactions between road-agents in dense and heterogeneous traffic. Our trajectory prediction algorithm outperforms state-of-the-art methods for end-to-end trajectory prediction using sensor inputs. We achieve an improvement of upto 18% in average displacement error and an improvement of up to 35.5% in final displacement error at the end of the prediction window (5 seconds) over the next best method. All experiments were set up on an Nvidia TiTan Xp GPU. Additionally, we release a software framework, TrackNPred. The framework consists of implementations of state-of-the-art tracking and trajectory prediction methods and tools to benchmark and evaluate them on real-world dense traffic datasets.

The second approach is called **RobustTP** [102]. This is an end-to-end approach that does not require manually labeled ground-truth trajectories to train the trajectory prediction network. The input to this algorithm consists only of raw traffic videos obtained from commodity sensors such as monocular RGB cameras. The algorithm uses a tracking algorithm to generate noisy trajectories from these videos. These trajectories replace the trajectory input used by TraPHic. This work has been published in **ACM CSCS'19**.

We begin by formally stating the problem and describing the notation. Then we give an overview of our approach to realtime end-to-end trajectory prediction in dense and heterogeneous traffic scenarios.

Given a set of N road agents $\mathcal{R} = \{r_i\}_{i=1...N}$, the trajectory history of each road agent

r_i over τ frames, denoted $\mathcal{T}_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_\tau, y_\tau)\}$, and the road agent's size l, we predict the trajectory, *i.e.*, the spatial coordinates of that road agent for the next k frames.

We define the state space of each road agent r_i as

$$\Omega_i := \begin{bmatrix} \mathcal{T}_i & \Delta\mathcal{T}_i & c & l \end{bmatrix}^\top, \tag{2.16}$$

where Δ is a derivative operator that is used to compute the velocity of the road agent, and $c := [c(x_1, y_1), \ldots, c(x_\tau, y_\tau)]^\top$. The traffic concentration, $c(x, y)$, at the location (x, y), is defined as the number of road agents between (x, y) and $(x, y) + (\delta x, \delta y)$ for some predefined $(\delta x, \delta y) > 0$.

We also compute camera parameters from given videos using standard techniques and use the parameters to estimate the camera homography matrices. The homography matrices are subsequently used to convert the location of road agents in 2D pixels to 3D world coordinates w.r.t. a predetermined frame of reference, similar to approaches in [9, 66]. All state-space representations are subsequently converted to the 3D world space.

Finally, we consider a method to be more robust compared to other methods if the trajectories predicted by it are less affected by noise in the trajectory history (arising due to sensor artifacts, inaccuracies in tracking and similar factors).

2.6.1 TrackNPred: A Software Framework for End-to-End Trajectory Prediction

TrackNPred is a python-based software library [2] for end-to-end realtime trajectory prediction for autonomous road-agents. Our first goal, through TrackNPred, is to enable autonomous road-agents to navigate safely in dense and heterogeneous traffic by estimating how road-agents, that

[2] https://gamma.umd.edu/robusttp

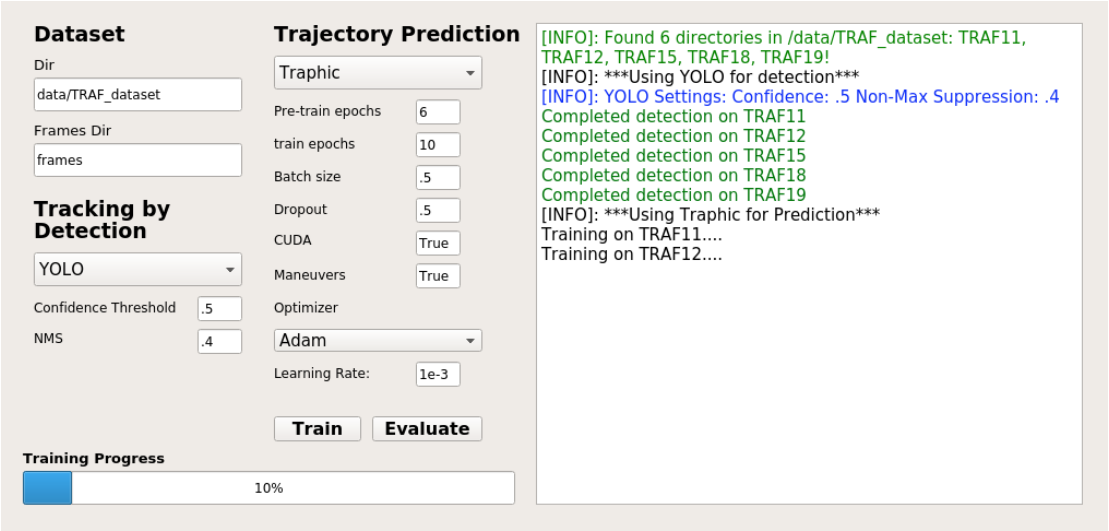


Figure 2.11: TrackNPred is a deep learning-based framework that integrates trajectory prediction methods with tracking by detection algorithms to motivate further research in end-to-end trajectory prediction. In this figure, we show the graphical user interface of TrackNPred where one can select the tracking by detection algorithm as well as choose the trajectory prediction method. The user can also set the hyperparameters for the training and evaluation phases. If the input can be connected to an RGB camera mounted on a road-agent, then TrackNPred can be extended to ADAS applications.

are in close proximity, are going to move in the next few seconds.

The continuous advancement in deep learning has resulted in the development of several state-of-the-art tracking and trajectory prediction algorithms that have shown impressive results on real world dense and heterogeneous traffic datasets. However, there are currently no theoretical guarantees to validate the comparison of performance of different deep learning models. It is only through empirical research that one can evaluate the efficiency of a particular deep learning model.

Our second goal is to equip researchers with a packaged deep learning tool that performs trajectory prediction based on various state-of-the-art neural network architectures, such as Generative Adversarial Networks (GANs [9]), Recurrent Neural Networks (LSTMs [66], and Convolutional Neural Networks (CNNs [67]). Therefore, one of the advantages of TrackNPred is

that it enables researchers to experiment with these different deep learning architectures with minimal difficulty. Researchers need only select hyperparameters for the chosen network. We also provide the ability to modify individual architectures without disrupting the rest of the methods (Figure 2.11).

TrackNPred integrates realtime tracking algorithms with end-to-end trajectory prediction methods to create a robust framework. The input is simply a video (through a moving or static RGB camera). TrackNPred selects a tracking method from the tracking module to first generate a trajectory, $\mathcal{T}_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, for the i^{th} road-agent for n frames, where n is a constant. The trajectories for each agent are then treated as the trajectory history for that agent in the trajectory prediction module. The final output is the future trajectory for the ego-agent, $\mathcal{T}_{\text{ego}} = ((x_{n+1}, y_{n+1}), (x_{n+2}, y_{n+2}), \ldots, (x_{n+k}, y_{n+k}))$, where k is the length of the prediction window. This is a major difference from trajectory prediction methods in the literature [9, 66, 67] that rely on manually annotated input trajectories. TrackNPred, in contrast, does not require any ground truth trajectories.

Finally, TrackNPred evaluates and benchmarks realtime performances of various trajectory prediction methods on a real-world traffic dataset[3] [65]. This dataset contains more than 50 videos of dense and heterogeneous traffic. The dataset consists of the following road agent categories: cars, buses, trucks, rickshaws, pedestrians, scooters, motorcycles, and other road agents such as carts and animals. Overall, the dataset contains approximately 13 motorized vehicles, 5 pedestrians, and 2 bicycles per frame. Annotations consist of spatial coordinates, an agent ID, and an agent type. The dataset is categorized according to camera viewpoint (front-facing/top-view), motion (moving/static), time of day (day/evening/night), and density

[3] https://go.umd.edu/TRAF-Dataset

Table 2.8: The list of algorithms currently implemented in TrackNPred.

	Methods
Tracking by Detection	Mask R-CNN + DeepSORT YOLO + DeepSORT
Trajectory Prediction	RNN- Encoder Decoder [68] Social-GAN [9] Covolutional Social-LSTM [67] TraPHic [65]

level (sparse/moderate/heavy/

challenging). All the videos have a resolution of 1280×720.

2.6.1.1 Methods Implemented in TrackNPred

One of our goals is to motivate research in highly accurate, end-to-end, and realtime trajectory prediction methods. To achieve this goal, we design a common interface for several state-of-the-art methods from both tracking and trajectory prediction literature. Such a design facilitates easy bench-marking of new algorithms with respect to the state-of-the-art . The methods in TrackNPred differ in numerous ways from their original implementations in the literature in order to achieve improved accuracy in tracking and prediction in dense and heterogeneous traffic. Table 2.8 provides a list of algorithms currently implemented in TrackNPred.

Tracking Module For tracking, we mainly focus our attention on tracking by detection approaches. These are approaches that leverage deep learning-based object detection models. This is because tracking methods that do not perform detection require manual, near-optimal initialization of each road-agent's state information in the first video frame. Further, methods that do not utilize

Table 2.9: We evaluate RobustTP with methods that use noisy sensor input, on the TRAF Dataset. The trajectory histories are computed using tracking by two detection methods: Mask R-CNN [6] and YOLO [7]. The results are reported in the following format: ADE/FDE, where ADE is the average displacement RMSE over the k seconds of prediction and FDE is the final displacement RMSE at the end of k seconds. We tested for both short-term ($k = 3$) and longer-term ($k = 5$) predictions. We observe for all the cases that RobustTP is the state-of-the-art.

Prediction length, $k = 3$ secs				
	RNN-ED	S-GAN	CS-LSTM	RobustTP
MRCNN	2.60/4.96	2.11/3.50	1.27/2.01	**1.14/1.90**
YOLO	1.13/2.18	1.29/2.18	1.08/1.55	**0.96/1.53**
Prediction length, $k = 5$ secs				
	RNN-ED	S-GAN	CS-LSTM	RobustTP
MRCNN	3.99/6.55	3.23/5.69	1.91/3.76	**1.75/3.42**
YOLO	2.06/4.26	1.98/3.72	1.52/2.67	**1.29/1.97**

object detection need to know the number of road-agents in each frame a priori so they do not handle cases in which new road-agents enter the scene during the video. Tracking by detection approaches overcome these limitations by employing a detection framework to recognize road-agents entering at any point during the video and initialize their state-space information.

At present, we implement python-based tracking by detection algorithms to facilitate easy integration into TrackNPred. DeepSORT [1] is currently the state-of-the-art realtime tracker implemented in python. Naturally, we use DeepSORT as the base tracker. However, DeepSORT was originally developed using a constant velocity model with the goal of tracking pedestrians in sparse crowds. Consequently, it is not optimized for dense and heterogeneous traffic scenes that may contain cars, buses, pedestrians, two-wheelers, and even animals. Therefore, we replace the constant velocity model with a non-linear RVO motion model [103], which is designed for motion planning in dense environments.

The advantage of using tracking by detection algorithms is that we can combine the unique benefits of different object detection models. For example, we integrate two state-of-the-art object detection models, YOLO and Mask R-CNN. They are state-of-the-art in its own category. The YOLO algorithm is extremely fast as compared to Mask R-CNN wile the latter offers a higher accuracy.

The output of the tracking module is a trajectory file with corresponding ID's. An ID is an integer unique to every agent. Each row of this file corresponds to the following format: $<$ Fid $>$, $<$ Vid $>$, $<$ center-X $>$, $<$ center-Y $>$ which denotes the frame ID, vehicle ID, and the 2D coordinates of the center of the bounding box of the road-agent. This trajectory file is input for the trajectory prediction module.

2.7 Behavior Prediction

In the final algorithm, **SpectralLSTM**, we extend TraPHic to also incorporate action prediction. Architecturally, the network consists of a two-stream approach working in parallel. The first stream is essentially the TraPHic algorithm while the second stream is used to perform action prediction. This work [104] has been published in **RAL/IROS'20**.

2.7.1 Problem Statement

We first present a definition of a vehicle trajectory:

Definition 2.7.1. ***Trajectory:*** *The trajectory for the i^{th} road agent is defined as a sequence $\Psi_i(a,b) \in \{\mathbb{R}^2\}$, where $\Psi_i(a,b) = \left\{[x_t, y_t]^\top \mid t \in [a,b]\right\}$. $[x,y] \in \mathbb{R}^2$ denotes the spatial coordinates of the road-agent in meters according to the world coordinate frame and t denotes*

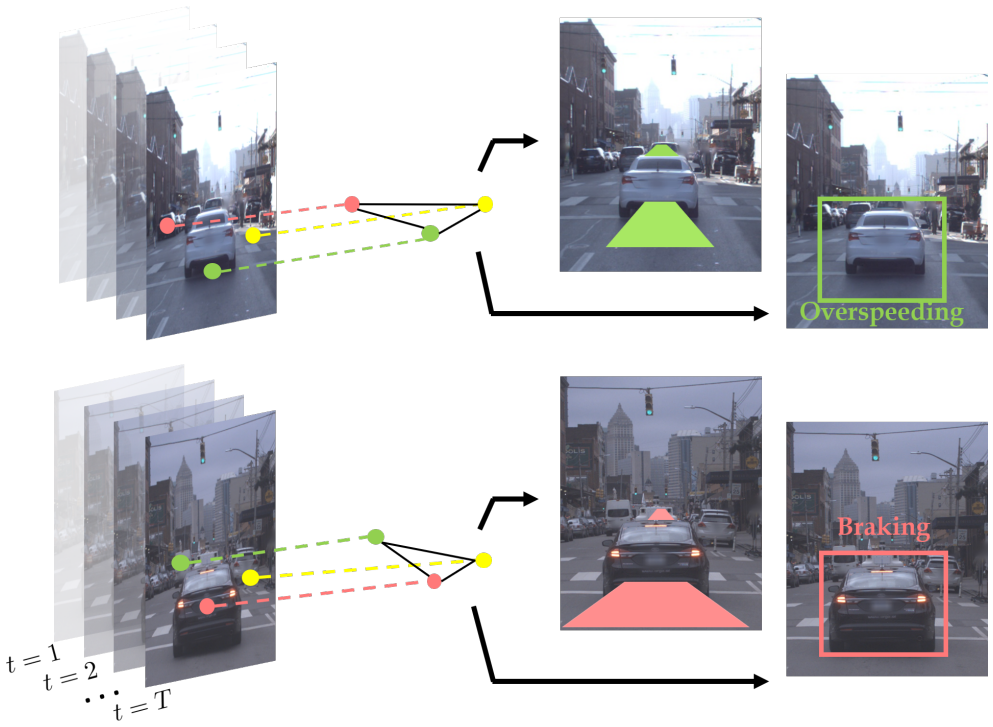


Figure 2.12: **Trajectory and Behavior Prediction:** We predict the long-term (3-5 seconds) trajectories of road-agents, as well as their behavior (e.g. overspeeding, underspeeding, etc.), in urban traffic scenes. Our approach represents the spatial coordinates of road-agents (colored points in the image) as vertices of a DGG to improve long-term prediction using a new regularization method.

the time instance.

We define *traffic forecasting* as solving the following two problem statements, simultaneously, but separately using two separate streams.

Problem 2.7.1. ***Trajectory Prediction:*** *In a traffic video with N road agents, given the trajectory $\Psi_i(0, \tau)$, predict $\Psi_i(\tau^+, T)$ for each road-agent $v_i, i \in [0, N]$.*

Problem 2.7.2. ***Behavior Prediction:*** *In a traffic video with N road agents, given the trajectory, $\Psi_i(0, \tau)$, predict a label from the following set, $\{$ Overspeeding, Neutral, Underspeeding$\}$ for each road-agent $v_i, i \in [0, N]$.*

The overall flow of the approach is as follows:

1. Our input consists of the spatial coordinates over the past τ seconds as well as the eigenvectors of the DGGs corresponding to the first τ DGGs.

2. *Solving Problem 2.7.1:* The first stream accepts the spatial coordinates and uses an LSTM-based sequence model [105] to predict $\Psi_i(\tau^+, T)$ for each $v_i, i \in [0, N]$, where $\tau^+ = \tau + 1$.

3. *Solving Problem 2.7.2:* The second stream accepts the eigenvectors of the input DGGs and predicts the eigenvectors corresponding to the DGGs for the next τ seconds. The predicted eigenvectors form the input to the behavior prediction algorithm in Section 2.7.2.2 to assign a behavior label to the road-agent.

4. Stream 2 is used to regularize stream 1 using a new regularization algorithm presented in Section 2.7.3. We derive the upper bound on the prediction error of the regularized forecasting algorithm in Section 2.7.3.1.

2.7.2 Network Overview

We present an overview of our approach in Figure 2.13 and defer the technical implementation details of our network to the supplementary material. Our approach consists of two parallel LSTM networks (or streams) that operate separately.

Stream 1: The first stream is an LSTM-based encoder-decoder network [105] (yellow layer in Figure 2.13). The input consists of the trajectory history, $\Psi_i(0, \tau)$ and output consists of $\Psi_i(\tau^+, T)$ for each road-agent $v_i, i \in [0, N]$.

Stream 2: The second stream is also an LSTM-based encoder-decoder network (blue layer in Figure 2.13). To prepare the input to this stream, we first form a sequence of DGGs, $\{\mathcal{G}_t | \ t \in [0, \tau]\}$ for each time instance of traffic until time τ. For each DGG, $\mathcal{G}_t$, we first compute

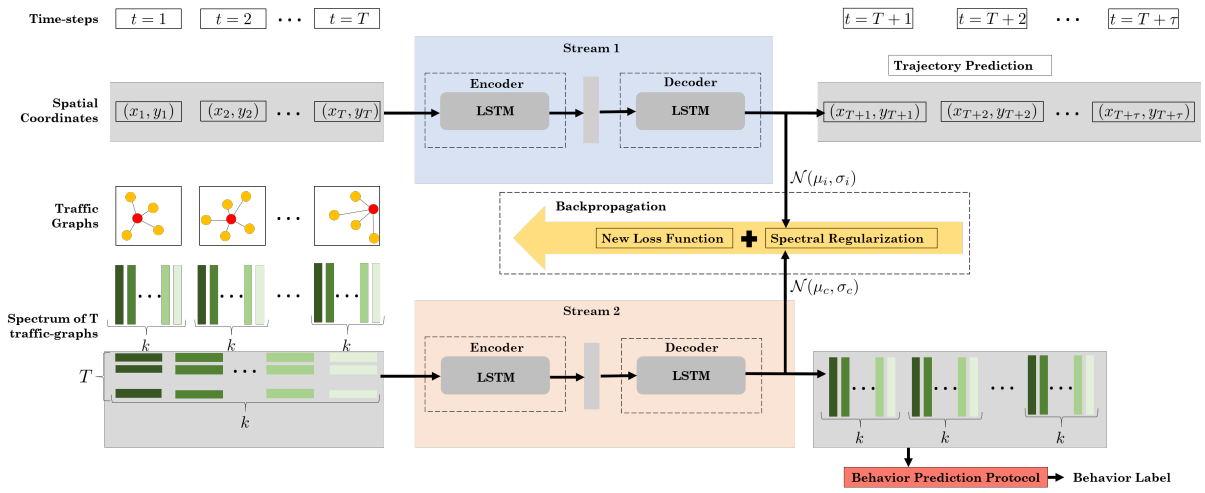


Figure 2.13: **Network Architecture**: We show the trajectory and behavior prediction for the i^{th} road-agent (red circle in the DGGs). The input consists of the spatial coordinates over the past τ seconds as well as the eigenvectors (green rectangles, each shade of green represents the index of the eigenvectors) of the DGGs corresponding to the first τ DGGs. We perform spectral clustering on the predicted eigenvectors from the second stream to regularize the original loss function and perform back-propagation on the new loss function to improve long-term prediction.

its corresponding Laplacian matrix, L_t and use state-of-the-art eigenvalue algorithms to obtain the spectrum, U_t consisting of the top k eigenvectors of length n. We form k different sequences, $\{\mathcal{S}_j |\ j \in [0, k]\}$, where each $\mathcal{S}_j = \{u_j\}$ is the set containing the j^{th} eigenvector from each U_t corresponding to the t^{th} time-step, with $|\mathcal{S}_j| = \tau$.

The second stream then accepts a sequence, $\mathcal{S}_j$, as input to predict the j^{th} eigenvectors for the next $T - \tau$ seconds. This is repeated for each $\mathcal{S}_j$. The resulting sequence of spectrums, $\{\mathcal{U}_t |\ t \in [\tau^+, T]\}$ are used to reconstruct the sequence, $\{\mathcal{L}_t |\ t \in [\tau^+, T]\}$, which is then used to assign a behavior label to a road-agent, as explained below.

2.7.2.1 Trajectory Prediction

The first stream is used to solve Problem 2.7.1. We clarify at this point that stream 1 does not take into account road-agent interactions. We use spectral clustering (discussed later in Section 2.7.3) to model these interactions. It is important to further clarify that the trajectories predicted from stream 1 are not affected by the behavior prediction algorithm (explained in the next Section).

2.7.2.2 Behavior Prediction Algorithm

We define a rule-based behavior algorithm (blue block in Figure 2.13) to solve Problem 2.7.2. This is largely due to the fact that most data-driven behavior prediction approaches require large, well-annotated datasets that contain behavior labels. Our algorithm is based on the predicted eigenvectors of the DGGs of the next τ seconds.

The degree of i^{th} road-agent, ($\theta_i \leq n$), can be computed from the diagonal elements of the Laplacian matrix L_t. θ_i measures the total number of distinct neighbors with which road-agent v_i has shared an edge connection until time t. As L_t is formed by simply adding a row and column to L_{t-1}, the degree of each road-agent monotonically increases. Let the rate of increase of θ_i be denoted as $\theta_i^{'}$. Intuitively, an aggressively overspeeding vehicle will observe new neighbors at a faster rate as compared to a road-agent driving at a uniform speed. Conversely, a conservative road-agent that is often underspeeding at unconventional spots such as green light intersections (Figure 2.12) will observe new neighbors very slowly. This intuition is formalized by noting the change in θ_i across time-steps. In order to make sure that slower vehicles (conservative) did not mistakenly mark faster vehicles as new agents, we set a condition where an observed vehicle is

marked as 'new' if and only if the speed of the observed vehicle is less than the active vehicle (or ego-vehicle). To predict the behavior of the i^{th} road-agent, we follow the following steps:

1. Form the set of predicted spectrums from stream 2, $\{\mathcal{U}_t |\ t \in [\tau^+, T]\}$. We compute the eigenvalue matrix, Λ, of L_t by applying theorem 5.6 of [106] to L_{t-1}. We explain the exact procedure in the supplemental version.

2. For each $U_t \in \mathcal{U}$, compute $L_t = U_t \Lambda U_t^\top$.

3. $\theta_i = i^{\text{th}}$ element of diag(L_t), where "diag" is the diagonal matrix operator.

4. $\theta_i^{'} = \frac{\Delta\theta_i}{\Delta t}$.

where Λ is the eigenvalue matrix of L_t. Based on heuristically pre-determined threshold parameters λ_1 and λ_2, we define the following rules to assign the final behavior label: Overspeeding ($\theta^{'} > \lambda_1$), Neutral ($\lambda_2 \leq \theta^{'} \leq \lambda_1$), and Underspeeding ($\theta^{'} < \lambda_2$).

Note that since human behavior does not change instantly at each time-step, our approach predicts the behavior over time periods spanning several frames.

2.7.3 Spectral Clustering Regularization

The original loss function of stream 1 for the i^{th} road-agent in an LSTM network is given by,

$$F_i = -\sum_{t=1}^{T} \log Pr\big(x_{t+1} \big| \mu_t, \sigma_t, \rho_t\big) \tag{2.17}$$

Our goal is to optimize the parameters, μ_t^*, σ_t^*, that minimize equation 2.17. Then, the next spatial coordinate is sampled from a search space defined by $\mathcal{N}(\mu_t^*, \sigma_t^*)$. The resulting optimization

forces μ_t, σ_t to stay close to the next spatial coordinate. However, in general trajectory prediction models, the predicted trajectory diverges gradually from the ground-truth, causing the error-margin to monotonically increase as the length of the prediction horizon increases ([107], cf. Figure 4 in [65, 67], Figure 3 in [8]). The reason for this may be that while equation 2.17 ensures that μ_t, σ_t stays close to the next spatial coordinate, it does not, however, guarantee the same for $\hat{x}_{t+1} \sim \mathcal{N}(\mu_t, \sigma_t)$. Our solution to this problem involves regularizing equation 2.17 by adding appropriate constraints on the parameters, μ_t, σ_t, such that sampled coordinates from $\mathcal{N}(\mu_t^*, \sigma_t^*)$ are close to the ground-truth trajectory.

We assume the ground-truth trajectory of a road-agent to be equivalent to their "preferred" trajectory, which is defined as the trajectory a road-agent would have taken in the absence of other dynamic road-agents. Preferred trajectories can be obtained by minimizing the Dirichlet energy of the DGG, which in turn can be achieved through spectral clustering on the road-agents [108]. Our regularization algorithm (shown in the yellow arrow in Figure 2.13) is summarized below. For each road-agent, v_i:

1. The second stream computes the spectrum sequence, $\{U_{T+1}, \ldots, U_{T+\tau}\}$.

2. For each U, perform spectral clustering [109] on the eigenvector corresponding to the second smallest eigenvalue.

3. Compute cluster centers from the clusters obtained in the previous step.

4. Identify the cluster to which v_i belongs and retrieve the cluster center, μ_c and deviation, σ_c.

Then for each road-agent, v_i, the regularized loss function, F_i^{reg}, for stream 1 is given by,

$$\sum_{t=1}^{T} \Big(-\log Pr(\hat{y}_{t+1}|\mu_t, \sigma_t, \rho_t \Big) + b_1 \|\mu_t - \mu_c\|_2 + b_2 \|\sigma_t - \sigma_c\|_2 \quad (2.18)$$

where $b_1 = b_2 = 0.5$ are regularization constants. The regularized loss function is used to backpropagate the weights corresponding to μ_t in stream 1. Note that F_i^{reg} resembles a Gaussian kernel. This makes sense as the Gaussian kernel models the Euclidean distance non-linearly – greater the Euclidean distance, smaller the Gaussian kernel value and vice versa. Furthermore, we can use Equation 2.18 to predict multiple modes [67] by computing maneuver probabilities using μ, σ following the approach in Section 4.3 of [67].

2.7.3.1 Upper Bound for Prediction Error

In this section, we derive an upper bound on the prediction error, ϕ_j, of the first stream as a consequence of spectral regularization. We present our main result as follows,

Theorem 2.7.1. $\phi_j \leq \frac{\|\delta_t \delta_t^\top\|_2}{min(\lambda_j, \Lambda)}$, *where* $min(\lambda_j, \Lambda)$ *denotes the minimum distance between* λ_j *and* $\lambda_k \in \Lambda \setminus \lambda_j$.

Proof. At time instance t, the Laplacian matrix, L_t, its block form, $\left[\begin{array}{c|c} L_t & 0 \\ \hline 0 & 1 \end{array}\right]$, denoted as block($L_t$), and the laplacian matrix for the next time-step, L_{t+1} are described by Equation 3.1. We compute the eigenvalue matrix, Λ, of L_t by applying theorem 5.6 of [106] to L_{t-1}.

LSTMs make accurate sequence predictions if elements of the sequence are correlated across time, as opposed to being generated randomly. In a general sequence of eigenvectors,

the eigenvectors may not be correlated across time. Consequently, it is difficult for LSTM networks to predict the sequence of eigenvectors, $\mathcal{U}$ accurately. This may adversely affect the behavior prediction algorithm described in Section 2.7.2.2. Our goal is now to show there exist a correlation between Laplacian matrices across time-steps and that this correlation is lower-bounded, that is, there exist sufficient correlation for accurate sequence modeling of eigenvectors.

Proving a lower-bound for the correlation is equivalent to proving an upper-bound for the noise, or error distance, between the j^{th} eigenvectors of L_t and L_{t+1}. We denote this error distance through the angle ϕ_j. From Theorem 5.4 of [106], the numerator of bound corresponds to the frobenius norm of the error between L_t and L_{t+1}. In our case, the update to the Laplacian matrix is given by Equation 3.1 where the error matrix is $\delta\delta^\top$. □

In Theorem 2.7.1, $\phi_j << 1$ and δ is defined in equation 3.1. λ_j represents the j^{th} eigenvalue and Λ represents all the eigenvalues of L_t. If the maximum component of δ_t is δ_{max}, then $\phi_j = \mathcal{O}(\sqrt{N}\delta_{max})$. Theorem 2.7.1 shows that in a sequence of j^{th} eigenvectors, the maximum angular difference between successive eigenvectors is bounded by $\mathcal{O}(\sqrt{N}\delta_{max})$. By setting $N = 270$ (number of road-agents in Lyft), and $\delta_{max} := e^{-3} = 0.049$ (width of a lane), we observe a theoretical upper bound of 0.8 meters. A smaller value of ϕ_j indicates a greater similarity between successive eigenvectors, thereby implying a greater correlation in the sequence of eigenvectors. This allows sequence prediction models to learn future eigenvectors efficiently.

An alternative approach to computing the spectrums $\{U_{T+1}, \ldots, U_{T+\tau}\}$ is to first form traffic-graphs from the predicted trajectory given as the output from the stream 1. After obtaining the corresponding Laplacian matrices for these traffic-graphs, standard eigenvalue algorithms can be used to compute the spectrum sequence. This is, however, a relatively sub-optimal approach

as in this case, $\phi = \mathcal{O}(NL_{max})$, with $L_{max} \gg \delta_{max}$.

2.7.3.2 Results

2.7.4 Analysis and Discussion

We compare the ADE and FDE scores of our predicted trajectories with prior methods in Table 3.3 and show qualitative results in the supplementary material. We compare with several state-of-the-art trajectory prediction methods and reduce the average RMSE by approximately 75% with respect to the next best method (GRIP).

Ablation Study of Stream 1 (S1 Only) vs. Both Streams (S1 + S2): To highlight the benefit of the spectral cluster regularization on long-term prediction, we remove the second stream and only train the LSTM encoder-decoder model (Stream 1) with the original loss function (equation 2.17). Our results (Table 3.3, last four columns) show that regularizing stream 1 reduces the FDE by up to 70%. This is as expected since stream 1 does not take into account neighbor information. Therefore, it should also be noted that stream 1 performs poorly in dense scenarios but rather well in sparse scenarios. This is evident from Table 3.3 where stream 1 outperforms comparison methods on the sparse NGSIM dataset with ADE less than 1m.

Additionally, Figure 2.14 shows that in the presence of regularization, the RMSE for our spectrally regularized approach ("both streams", purple curve) is much lower than that of stream 1 (red curve) across the entire prediction window.

RMSE depends on traffic density: The upper bound for the increase in RMSE error is a function of the density of the traffic since $\phi = \mathcal{O}(\sqrt{N}\delta_{max})$, where N is the total number of agents in the traffic video and $\delta_{max} = 0.049$ meters for a three-lane wide road system. The

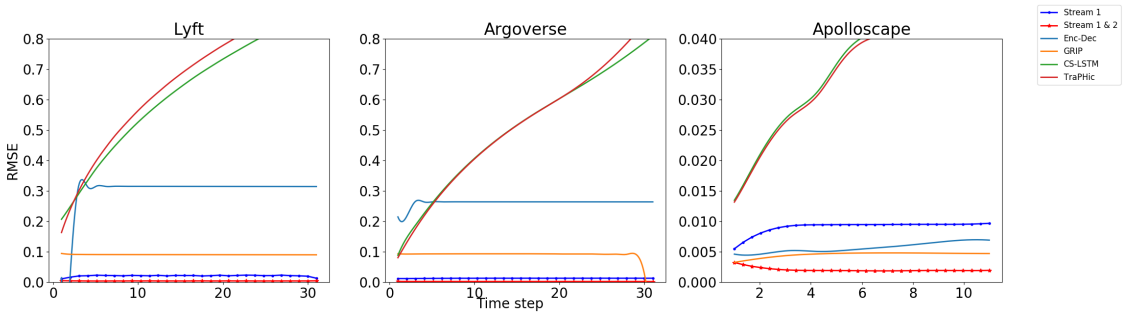


Figure 2.14: **RMSE Curves:** We plot the RMSE values for all methods. The prediction window is 5 seconds corresponding to a frame length of 50 for the NGSIM dataset.

NGSIM dataset contains the sparsest traffic with the lowest value for N and therefore the RMSE values are lower for the NGSIM $(0.40/1.08)$ compared to the other three datasets that contain dense urban traffic.

Comparison with other methods: Our method learns weight parameters for a spectral regularized LSTM network (Figure 2.13), while GRIP learns parameters for a graph-convolutional network (GCN). We outperform GRIP on the NGSIM and Apolloscape datasets, while comparisons on the remaining two datasets are unavailable. TraPHic and CS-LSTM are similar approaches. Both methods require convolutions in a heuristic local neighborhood. The size of the neighborhood is specifically adjusted to the dataset that each method is trained on. We use the default neighborhood parameters provided in the publicly available implementations, and apply them to the NGSIM, Lyft, Argoverse, and Apolloscape datasets. We outperform both methods on all benchmark datasets. Lastly, Social-GAN is trained on the scale of pedestrian trajectories, which differs significantly from the scale of vehicle trajectories. This is primarily the reason behind Social-GAN placing last among all methods.

2.7.5 Long-Term Prediction Analysis

The goal of improved long-term prediction is to achieve a lower FDE, as observed in our results in Table 3.3. We achieve this goal by successfully upper-bounding the worst-case maximum FDE that can theoretically be obtained. These upper bounds are a consequence of the theoretical results in Section 2.7.3.1. We denote the worst-case theoretical FDE by T-FDE. This measure represents the maximum FDE that can be obtained using Theorem 2.7.1 under fixed assumptions. In Table 2.11, we compare the T-FDE with the empirical FDE results obtained in Table 3.3. The T-FDE is computed by,

$$\text{T-FDE} = \frac{\phi}{n} \times (T - \tau) \tag{2.19}$$

The formula for T-FDE is derived as follows. The RMSE error incurred by all vehicles at a current time-step during spectral clustering is bounded by ϕ (Theorem 2.7.1). Let $n = \frac{N}{T} = 10$ be the average number of vehicles per frame in each dataset. Then, at a single instance in the prediction window, the increase in RMSE for a single agent is bounded by $\frac{\phi}{n}$. As $T - \tau$ is the length of the prediction window, the total increase in RMSE over the entire prediction window is given by T-FDE $= \frac{\phi}{n} \times (T - \tau)$. We do not have the data needed to compute ϕ for the NGSIM dataset as the total number of lanes are not known.

We note a $73\%, 82\%, 100\%$ agreement between the theoretical FDE and the empirical FDE on the Apolloscape, Lyft, and Argoverse datasets, respectively. The main cause for disagreements in the first two datasets is the choice for the value of $\delta_{\text{max}} = 0.049$ during the computation of ϕ. This value is obtained for a three-lane wide road system that was observed in majority of the

Table 2.10: **Main Results:** We report the Average Displacement Error (ADE) and Final Displacement Error (FDE) for prior road-agent trajectory prediction methods in meters (m). Lower scores are better and **bold** indicates the SOTA. We used the original implementation and results for GRIP [8] and Social-GAN [9]. '-' indicates that results for that particular dataset are not available. **Conclusion:** Our spectrally regularized method ("S1 + S2") outperforms the next best method (GRIP) by upto 70% as well as the ablated version of our method ("S1 Only") by upto 75%.

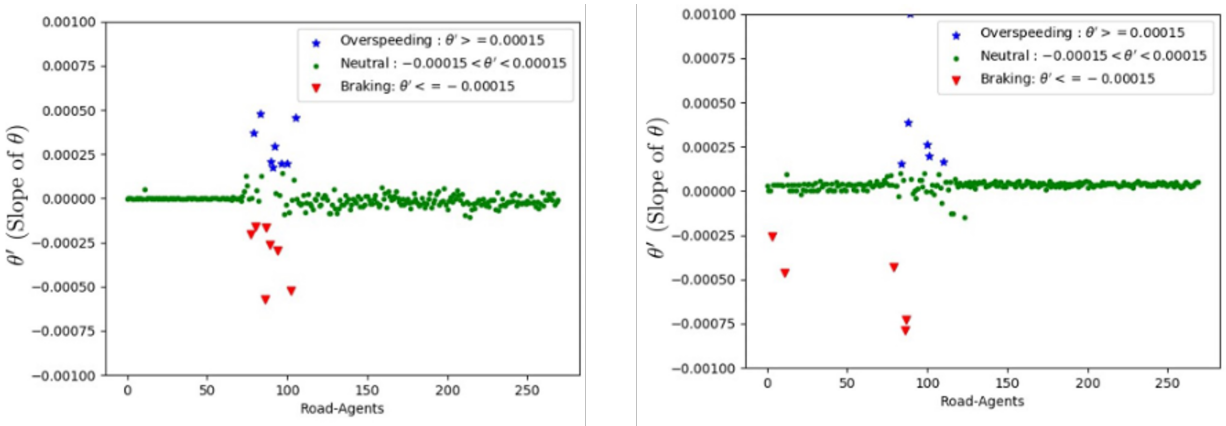
Dataset (Pred. Len.)	Comaprison Methods								Ablation		**Our Approach**	
	CS-LSTM		TraPHic		Social-GAN		GRIP		S1 Only		S1 + S2	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
Lyft (5 sec.)	4.423	8.640	5.031	9.882	7.860	14.340	-	-	5.77	11.20	**2.65**	**2.99**
Argoverse (5 sec.)	1.050	3.085	1.039	3.079	3.610	5.390	-	-	2.40	3.09	**0.99**	**1.87**
Apolloscape (3 sec.)	2.144	11.699	1.283	11.674	3.980	6.750	1.25	2.34	2.14	9.19	**1.12**	**2.05**
NGSIM (5 sec.)	7.250	10.050	5.630	9.910	5.650	10.290	1.61	3.16	1.31	2.98	**0.40**	**1.08**

videos in both datasets. However, it may be the case that several videos contain one- or two-lane traffic. In such cases, the values for $\delta_{\max}$ changes to 0.36 and 0.13, respectively, thereby increasing the upper bound for increase in RMSE.

Note, in Figure 2.14, the increase in RMSE for our approach (purple curve) is much lower than that of other methods, which is due to the upper bound induced by spectral regularization.

2.7.6 Behavior Prediction Results

We follow the behavior prediction algorithm described in Section 2.7.2.2. The values for λ_1 and λ_2 are based on the ground truth labels and are hidden from the test set. We observe a weighted accuracy of 92.96% on the Lyft dataset, 84.11% on the Argoverse dataset, and 96.72% on the Apolloscape dataset. In the case of Lyft, Figure 2.15(top) and Figure 2.15(bottom) show the ground truth and predictions for Lyft, respectively. We plot the value of θ' on the vertical axis and the road-agent I.D.s on the horizontal axis. More similarity across the two plots indicates higher accuracy. For instance, the red (aggressive) and blue (conservative) dotted regions inFigure 2.15(top) and Figure 2.15(bottom) are nearly identical indicating a greater number



(a) Lyft Ground-Truthwith λ=0.00015.

(b) Lyft Behavior Predictions.

Figure 2.15: **Behavior Prediction Results:** We classify the three behaviors– overspeeding(blue), neutral(green), and underspeeding(red), for all road-agents in the Lyft, Argoverse, and Apolloscape datasets, respectively. The y-axis shows $\theta^{'}$ and the x-axis denotes the road-agents. We follow the behavior prediction protocol described in Section 2.7.2.2. Each figure in the top row represents the ground-truth labels, while the bottom row shows the predicted labels. In our experiments, we set $\lambda = \lambda_1 = -\lambda_2$.

of correct classifications. Similar results follow for the Apolloscape and Argoverse datasets, which we show in the supplementary material due to lack of space. Due to the lack of diverse behaviors in the NGSIM dataset, we do not perform behavior prediction on the NGSIM.

An interesting observation is that road-agents towards the end of the x-axis appear late in the traffic video while road-agents at the beginning of the x-axis appear early in the video. The variation in behavior class labels, therefore, decreases towards the end of the x-axis. This intuitively makes sense as $\theta^{'}$ for a road-agent depends on the number of distinct neighbors that it observes. This is difficult for road-agents towards the end of the traffic video.

2.7.6.1 Results

We compare the ADE and FDE scores of our predicted trajectories with prior methods in Table 3.3 and show qualitative results in the supplementary material. We compare with several

state-of-the-art trajectory prediction methods and reduce the average RMSE by approximately 75% with respect to the next best method (GRIP).

Ablation Study of Stream 1 (S1 Only) vs. Both Streams (S1 + S2): To highlight the benefit of the spectral cluster regularization on long-term prediction, we remove the second stream and only train the LSTM encoder-decoder model (Stream 1) with the original loss function (equation 2.17). Our results (Table 3.3, last four columns) show that regularizing stream 1 reduces the FDE by up to 70%. This is as expected since stream 1 does not take into account neighbor information. Therefore, it should also be noted that stream 1 performs poorly in dense scenarios but rather well in sparse scenarios. This is evident from Table 3.3 where stream 1 outperforms comparison methods on the sparse NGSIM dataset with ADE less than 1m.

Additionally, Figure 2.14 shows that in the presence of regularization, the RMSE for our spectrally regularized approach ("both streams", purple curve) is much lower than that of stream 1 (red curve) across the entire prediction window.

RMSE depends on traffic density: The upper bound for the increase in RMSE error is a function of the density of the traffic since $\phi = \mathcal{O}(\sqrt{N}\delta_{max})$, where N is the total number of agents in the traffic video and $\delta_{max} = 0.049$ meters for a three-lane wide road system. The NGSIM dataset contains the sparsest traffic with the lowest value for N and therefore the RMSE values are lower for the NGSIM $(0.40/1.08)$ compared to the other three datasets that contain dense urban traffic.

Comparison with other methods: Our method learns weight parameters for a spectral regularized LSTM network (Figure 2.13), while GRIP learns parameters for a graph-convolutional network (GCN). We outperform GRIP on the NGSIM and Apolloscape datasets, while comparisons on the remaining two datasets are unavailable. TraPHic and CS-LSTM are similar approaches.

Both methods require convolutions in a heuristic local neighborhood. The size of the neighborhood is specifically adjusted to the dataset that each method is trained on. We use the default neighborhood parameters provided in the publicly available implementations, and apply them to the NGSIM, Lyft, Argoverse, and Apolloscape datasets. We outperform both methods on all benchmark datasets. Lastly, Social-GAN is trained on the scale of pedestrian trajectories, which differs significantly from the scale of vehicle trajectories. This is primarily the reason behind Social-GAN placing last among all methods.

2.7.7 Long-Term Prediction Analysis

The goal of improved long-term prediction is to achieve a lower FDE, as observed in our results in Table 3.3. We achieve this goal by successfully upper-bounding the worst-case maximum FDE that can theoretically be obtained. These upper bounds are a consequence of the theoretical results in Section 2.7.3.1. We denote the worst-case theoretical FDE by T-FDE. This measure represents the maximum FDE that can be obtained using Theorem 2.7.1 under fixed assumptions. In Table 2.11, we compare the T-FDE with the empirical FDE results obtained in Table 3.3. The T-FDE is computed by,

$$\text{T-FDE} = \frac{\phi}{n} \times (T - \tau) \tag{2.20}$$

The formula for T-FDE is derived as follows. The RMSE error incurred by all vehicles at a current time-step during spectral clustering is bounded by ϕ (Theorem 2.7.1). Let $n = \frac{N}{T} = 10$ be the average number of vehicles per frame in each dataset. Then, at a single instance in the prediction window, the increase in RMSE for a single agent is bounded by $\frac{\phi}{n}$. As $T - \tau$ is the

Table 2.11: **Upper Bound Analysis:** ϕ is the upper bound on the RMSE for all agents at a time-step. $T - \tau$ is the length of the prediction window. T-FDE (Eq. 2.20) is the theoretical FDE that should be achieved by using spectral regularization. The FDE results are obtained from Table 3.3. The % agreement is the agreement between the T-FDE and FDE computed using $\frac{\text{T-FDE}}{\text{FDE}}$ if T-FDE<FDE, else 100%. *Conclusion:* Theorem 2.7.1 is empirically verified with at least 73% guarantee.

Dataset	ϕ	$(T - \tau)$	T-FDE	FDE	% Agreement
Lyft Level 5	0.80	30	2.46	2.99	82%
Apolloscape	1.50	10	1.50	2.05	73%
Argoverse	0.64	30	1.95	1.87	100%

length of the prediction window, the total increase in RMSE over the entire prediction window is given by T-FDE $= \frac{\phi}{n} \times (T - \tau)$. We do not have the data needed to compute ϕ for the NGSIM dataset as the total number of lanes are not known.

We note a $73\%, 82\%, 100\%$ agreement between the theoretical FDE and the empirical FDE on the Apolloscape, Lyft, and Argoverse datasets, respectively. The main cause for disagreements in the first two datasets is the choice for the value of $\delta_{\text{max}} = 0.049$ during the computation of ϕ. This value is obtained for a three-lane wide road system that was observed in majority of the videos in both datasets. However, it may be the case that several videos contain one- or two-lane traffic. In such cases, the values for δ_{max} changes to 0.36 and 0.13, respectively, thereby increasing the upper bound for increase in RMSE.

Note, in Figure 2.14, the increase in RMSE for our approach (purple curve) is much lower than that of other methods, which is due to the upper bound induced by spectral regularization.

Chapter 3: Online Driver Behavior Modeling

3.1 Overview

Autonomous Vehicles (AVs) are an active area of research, successfully employing tools from machine learning [110], perception [111], planning and driver behavior modeling [75]. Recently, there have been multiple breakthroughs in perception-based tasks in autonomous driving in areas that include object detection [112], tracking [12, 45], trajectory prediction [65, 102, 104], and planning [113, 114]. While these advances have been widely successful, current AVs still lack the ability to interact with multiple human drivers in dense traffic scenarios [115] such as intersections and merging on highways.

As a precautionary measure, AVs are designed to behave conservatively in order to maximize safety [116]. A recent study [117] 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 [118]. For example, in [118], 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 [116] have pointed out

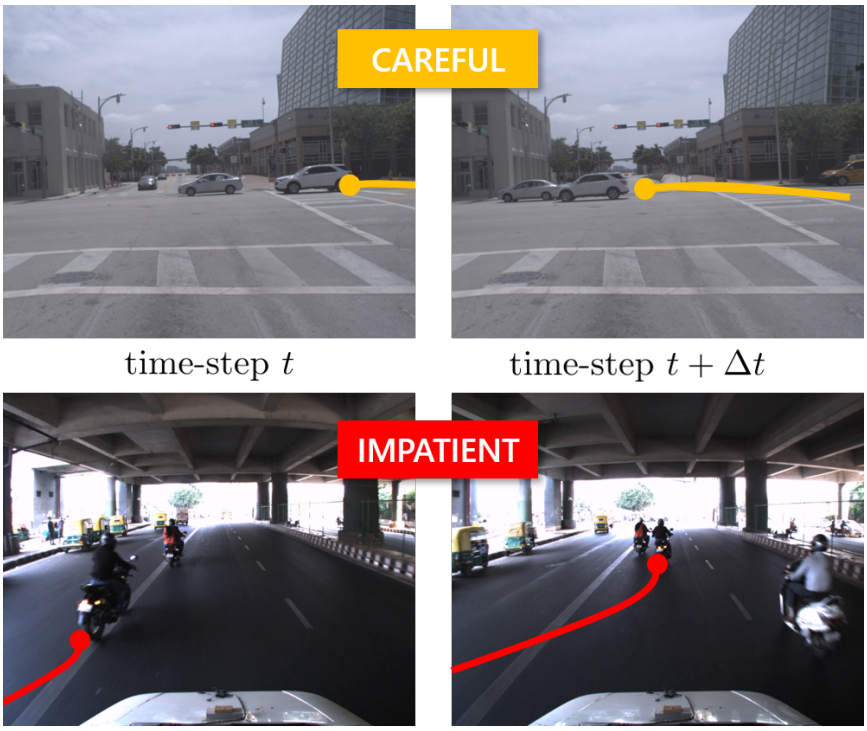


Figure 3.1: GraphRQI predicts the behaviors of road-agents in dense and heterogeneous traffic from one of the following classes–*impatient, reckless, threatening, careful, cautious, timid*. Our approach is up to 25% more accurate than prior behavior prediction methods.

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 [14, 15, 75, 119, 120]. 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 [121] 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 [122], 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 [123]. 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 [124] 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 [122] 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 3.5.1.2, Table 3.3) 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 3.4.4.1 and Table 3.5, we demonstrate robustness of our method to these sensor issues.

3.2 Related Work

3.2.1 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 [125], and convolutional neural networks, remain the same, they are now referred to as graph neural networks (GNNs) [126], Graph-LSTMs [104], and graph convolutional networks (GCNs) [127], respectively. Graph-based machine learning algorithms have been widely used in trajectory prediction, computer vision and natural language processing [128]. 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 [129], multi-layer perceptrons, and support

vector machines [130], which require fewer computational resources than deep learning-based methods.

3.2.2 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 [92, 93, 94]. 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) [75, 119, 131]. 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 [119, 131]. For instance, the approach proposed in [119] requires 32 million data samples for optimum performance. Additionally, IRL-based methods are sensitive to noise in the trajectory data [75, 131]. Consequently, current IRL-based methods are restricted to simple and sparse traffic conditions.

3.2.3 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 [132], probabilistic search-based [133], or use non-linear control optimization [134] approaches.

In addition to vehicular control and motion planning methods, many learning-based techniques are also used [135, 136, 137]. 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. [138] 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 [137]. Our behavior-based formulation can be integrated with these methods. We refer the reader to [122] for a detailed review on recent planning and navigation methods.

3.2.4 Interpretation of Driver Behavior in Social Science

Many studies have attempted to define driver behavior for traffic-agents. Sagberg et al. [10] extract and summarize the common elements from these definitions and propose a unified definition

Table 3.1: A list showing the taxonomy of various aggressive and conservative behaviors. In this work, we focus on modeling the longitudinal and lateral specific styles (except "tailgating" and "responding to pressure"). NA denotes "Not Applicable"

Global Behavior	Specific Style/Indicator	Nature
Aggressive	Overspeeding	Longitudinal
	Tailgating	Longitudinal
	Overtake as much as possible	Lateral
	Weaving	Lateral
	Sudden Lane-Changes	Lateral
	Inappropriate use of horn	NA
	Flashing lights at vehicle in front	NA
Conservative	Uniform speed or under-speeding	Longitudinal
	Conforms to single lane	Lateral
	Responding to pressure from other drivers	NA

for driver behavior. We incorporate this definition in our driver behavior model.

Definition 3.2.1. ***(Sagberg et al. [10]*** *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 3.2.1. *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 3.4.2, we elucidate on "mathematically modeling" a specific style. In Section 3.3, we construct the "traffic-graph" data structure used by our approach. We introduce the ideas of vertex centrality in Section 3.4.1 followed by a presentation of our main approach in Section 3.4.2. We describe the experiments and results in Section 4.4.3.

3.3 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 [139] 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 3.3.1. *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 [140]. 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 j^{th} column corresponding to the i^{th} row of the adjacency matrix stores the relative distance between the i^{th} 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 $t \times t$,

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

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

$$A_{t+1} = \overbrace{\left[\begin{array}{c|c} [A_t]_{\mathbf{t}\times\mathbf{t}} & 0 \\ \hline 0 & 1 \end{array}\right]}^{(\mathbf{t+1})\times(\mathbf{t+1})} + \delta\sigma(\delta)^\top, \tag{3.1}$$

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,

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

Here, σ is a permutation that swaps the two columns in δ. If the j^{th} row of δ is non-zero, then that implies that the j^{th} 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 3.1 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 ($N \times N$).

Table 3.2: Definition and categorization of driving behaviors [10]. 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—

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.

3.4 StylePredict: Mapping Trajectories to Behavior

3.4.1 Centrality Measures

In graph theory and network analysis, centrality measures [124] 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 [141] and key infrastructure nodes in the Internet [142], to rank web-pages in search engines [143], and to discover the

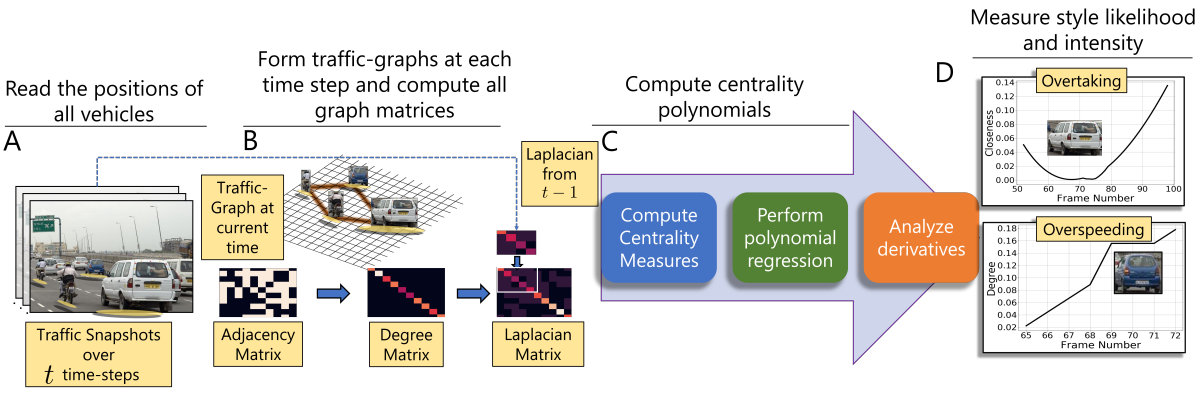


Figure 3.2: **Overview:** The autonomous vehicle reads the positions of all vehicles in realtime. The positions and corresponding spatial distances between vehicles are represented through a traffic-graph $\mathcal{G}_t$ (Section 3.3). We use the centrality functions defined in Section 3.4.1 to model the specific driving style corresponding to the global behaviors as outlined in Table 3.2.

origin of epidemics [144]. 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 [14] (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 [14].

3.4.2 Algorithm

Here, we present the main algorithm, called *StylePredict*, for solving Problem 3.2.1. 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 [10] according to Table 3.2. 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 3.3).

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 3.2. We begin by using the construction described in Section 3.3 to form the traffic-graphs for each frame and use the definitions in [14] 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 3.4.4).

3.4.3 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 3.4.4.

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,

$$\beta = (M^\top M)^{-1} M^\top \zeta^i \tag{3.2}$$

Here, $M \in \mathbb{R}^{T\times(d+1)}$ is the Vandermonde matrix [145]. and is given by,

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

3.4.4 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) [14], which are used to measure the probability and the intensity of a specific style.

[1] A polynomial with degree 2.

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 $\text{SLE}(t) = \left|\frac{\partial \zeta_c(t)}{\partial t}\right|$. The maximum likelihood SLE_{max} can be computed as $\text{SLE}_{\text{max}} = \max_{t \in \Delta t} \text{SLE}(t)$. The SIE of overtaking is computed by simply measuring the second derivative of the closeness centrality using $\text{SIE}(t) = \left|\frac{\partial^2 \zeta_c(t)}{\partial t^2}\right|$. Sudden lane-changes follow a similar maneuver to overtaking and therefore can be modeled using the same equations used to model overtaking.

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 3.1), the degree of the i^{th} vehicle (denoted as θ_i) is calculated by simply counting the number of non-zero entries in the i^{th} 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,

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

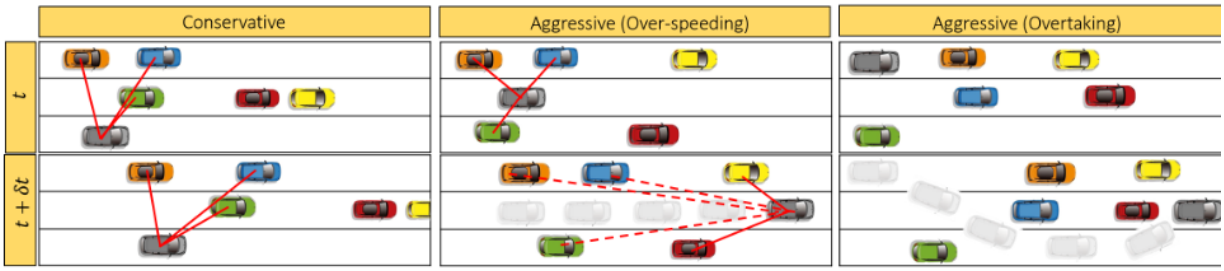
Similar to overtaking, the maximum likelihood estimate is given by $\text{SLE}_{\text{max}} = \max_{t \in \Delta t} \text{SLE}(t)$. Figure 3.3 visualize how the degree centrality can distinguish between an overspeeding vehicle and a vehicle driving at a uniform speed.

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 $\mathcal{T} = \left\{t_c \middle| \frac{\partial \zeta_c(t_c)}{\partial t} = 0\right\}$. 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 [146] of the closeness centrality is non-zero. The set $\mathcal{T}$ is reformulated as follows,

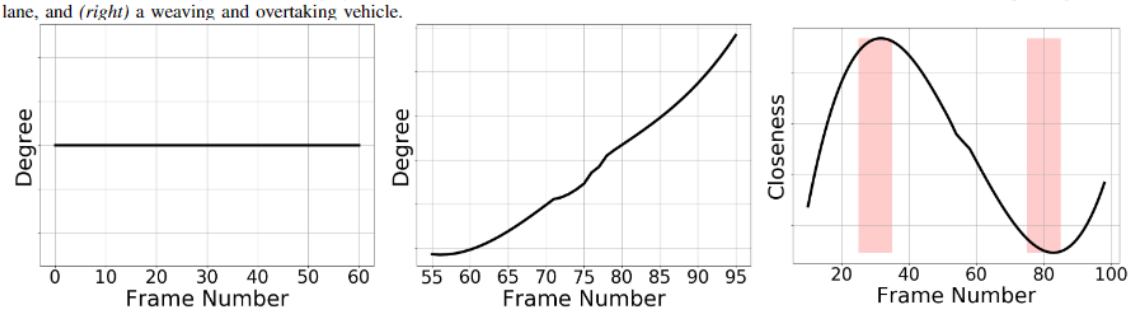
$$\begin{aligned} \mathcal{T} &= \left\{ t_c \middle| \frac{\partial \zeta_c(t_c)}{\partial t} = 0 \right\} \\ \text{s.t.} \max_{t \in \mathcal{B}_\varepsilon(t_c)} & \frac{\partial \zeta_c(t)}{\partial t} \neq \frac{\partial \zeta_c(t_c)}{\partial t} \end{aligned} \tag{3.3}$$

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 3.3 visualizes how the degree centrality can distinguish between an overspeeding vehicle and a vehicle driving at a uniform speed.

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 [147] as much as possible, and drive at a uniform speed [10] 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.



(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 th 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 3.3: **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 part (b) , the derivative of the degree centrality function is 0 because the ego-vehicle does not observe any additional new neighbors (See Section 3.4.4), so the degree centrality is a constant function; therefore, the vehicle is conservative. In part (c), 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 part (d), 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.

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.

3.4.4.1 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 i^{th} agent is given by $\zeta^i \in \mathbb{R}^{T\times 1}$. Therefore, $\zeta^1[2]$ corresponds to the degree centrality value of the 1^{st} 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 3.2. However, in the presence of noise ϵ, the OLS system described in Equation 3.2 is modified as,

$$\tilde{\beta} = (M^\top M)^{-1} M^\top \tilde{\zeta}^i \tag{3.4}$$

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.

3.5 Behavior Classification Using Machine Learning

We treat the centrality polynomials computed in Section 3.4.3 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 4.4.3.

Algorithm 1: Our approach outputs the Style Likelihood Estimate (SLE) and Style Intensity Estimate (SIE) for a vehicle, u, in a given time-period Δt.

Input : $u = v_i \leftarrow [x_i, y_i]^\top \; \forall v_i \in \mathcal{V}(t)$
Output: SLE(t), SIE(t)

$t = 0$
for *each* $v \in \mathcal{V}(t)$ **do**
  while $t \leq T$ **do**
    // Compute Centrality //
    $\zeta_c^i[t] = \frac{N-1}{\sum_{v_j \in \mathcal{V}(t) \setminus \{v_i\}} \mathcal{D}_t(v_i, v_j)}$
    $\zeta_d^i[t] = \left|\{v_j \in \mathcal{N}_i(t)\}\right| + \zeta_d^i[t-1], (v_i, v_j) \notin \mathcal{E}(\tau), \tau = 0, \ldots, t-1$
    $t \leftarrow t + 1$
    // Perform Polynomial Regression w.r.t Time //
    $\beta = \arg\min_\beta \|\zeta - M\beta\|$
    $\zeta(t) = \beta_0 + \beta_1 t + \beta_2 t^2$
    // Compute Likelihood and Intensity //
    for $k = 0, 1, 2$ **do**
      SLE$_k(t) = \left|\frac{\partial \zeta(t)}{\partial t}\right|$
      SIE$_k(t) = \left|\frac{\partial^2 \zeta(t)}{\partial t^2}\right|$
    end
  end
end

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,

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \|y_i - \Phi(\zeta^i(t))\|^2 \tag{3.5}$$

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 4.12. More simply,

Table 3.3: 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 3.5.1.3

Dataset	Styles							
	OS		OT		SLC		W	
	TDE	%	TDE	%	TDE	%	TDE	%
U.S. [11]	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 [148]	0.74s	24	0.44s	32	0.39s	36	0.23s	8
India [65]	0.81s	16	0.38s	40	0.19s	28	0.06s	16

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

3.5.1 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 3.5.1.1. 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 3.5.1.2. We use the TDE to validate our approach and analyze the results in real-world traffic datasets in Section 3.5.1.3. Finally, we analyze the weighted accuracy of StylePredict and compare with state-of-the-art graph classification and behavior prediction methods in Section 3.5.1.4.

3.5.1.1 Evaluation Metrics

1. Time Deviation Error (TDE) [14]: 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,

$$\text{TDE}_{\text{style}} = \left| \frac{t_{\text{SLE}} - \mathbb{E}[T]}{f} \right| \tag{3.6}$$

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 3.4.2, $\mathbb{E}[T]$ is computed using Algorithm 2, 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 .

3.5.1.2 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).

Algorithm 2: 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

$s^* = \min S$

$e^* = \max E$

Initialize a counter $c_t = 0$ for each frame $t \in [s^*, e^*]$

for $t \in [s^*, e^*]$ **do**

  if $t \in [s_m, e_m]$ **then**

    $c_t \leftarrow c_t + 1$

  end

  $\mathcal{P}(T = t) = c_t$

end

$\mathbb{E}[T] = \sum_t t c_t, t = s^*, s^* + 1, \ldots, e^*$

Simulation Environment We use the Highway-Env simulator [149] 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 [150] as the underlying motion model where the linear acceleration model is based on the Intelligent Driver Model (IDM) [151] and the lane changing behavior is based on the MOBIL [152] model. We note here that more sophisticated car models such as the Ackermann steering model may be used. While many popular vehicle simulators [153, 154] 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 [149] 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.

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) [11], New Delhi (India) [65], Beijing (China) [148], 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 2. We directly use the raw trajectory data from these datasets without any pre-processing or filtering step. 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 3.6. Here, $t_{\text{SLE}} = \arg\max_{t \in \Delta t} \text{SLE}(t)$, as explained in Section 3.4.2. We use Algorithm 2 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 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 2 is applied separately for each video in each dataset.

3.5.1.3 Validating StylePredict Using TDE

In Table 3.5, we report the average TDE in seconds (s) in simulation environments. We used Algorithm 2 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 3.4). 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 3.3, 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 3.3 and qualitative results in Figure 3.4 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 3.4, 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 are color heat maps that correspond to $\mathcal{P}(T)$ (line 8, Algorithm 2). 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

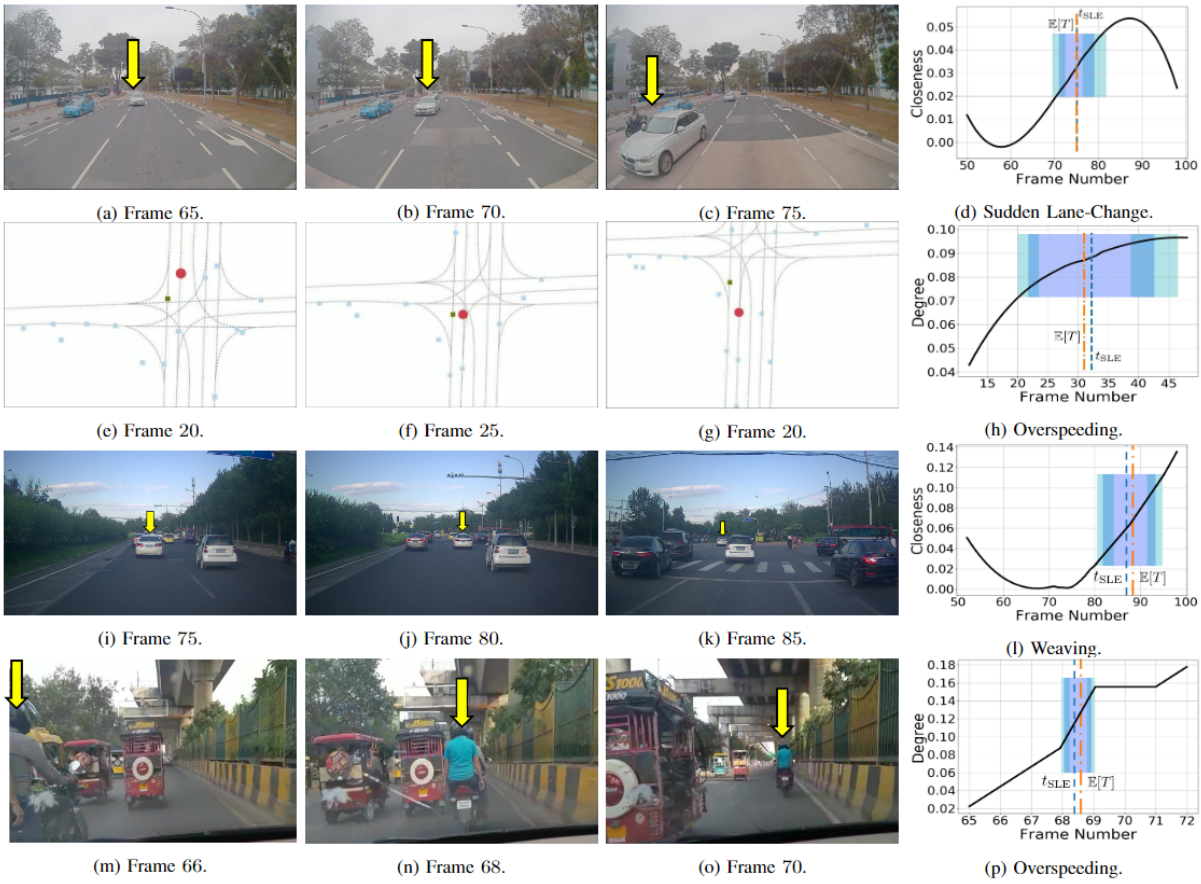


Figure 3.4: **Driver Behavior Modeling in Singapore *(top row)*, U.S. *(second row)*, China *(third row)*, and India *(bottom row)*:** In each row, the first three figures demonstrate the trajectory of a vehicle executing an aggressive driving style (sudden lane change, overspeeding, weaving, and overspeeding, 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 2).

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 75^{th} frame (blue dashed line), 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 30^{th} frame with a TDE of 0.25

Table 3.4: We compare the weighted classification accuracy of StylePredict versus supervised learning-based SOTA methods on the Argoverse dataset [11]. Additionally, we compare the accuracy of different supervised learning machine learning and deep learning algorithms.

Dataset	Method	Weighted Accuracy
Argoverse	DANE [155]	65.50%
	Cheung et al. [89]	62.50%
	StylePredict *w. LR*	69.90%
	StylePredict *w. RNN*	70.80%
	StylePredict *w. SVM*	75.00%
	StylePredict *w. MLP*	**89.90%**

seconds. Note that in both cases the TDE (the distance between the blue and orange dashed vertical lines) is < 1 second.

3.5.1.4 Analyzing Behavior Prediction Using Weighted Accuracy

We compare our approach with Dynamic Attributed Network Embedding (DANE) [155] and Cheung et al. [89]. 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 3.4 where we show an improvement of up to 25%.

StylePredict uses a multi-layer perceptron (MLP) [156] for the classification task. However, other classifiers in the machine learning literature such as logistic regression (LR), support vector

Table 3.5: **Analysing Simulation Results using TDE:** We analyze StylePredict by varying the traffic density, number of lanes, and the noise parameter ϵ (Equation 3.4). We observe that TDE increases as these parameters increase in value. We discuss these results in detail in Section 3.5.1.3.

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

machines (SVM)), and deep neural networks (RNNs) may be used. In Table 3.4, we compare the results of replacing the MLP with different classifiers and benchmark the different output accuracies against the MLP.

3.6 Speeding up the eigenvector centrality

3.6.1 Eigenvalue Algorithm

Our eigenvalue algorithm is built upon the classical RQI algorithm [157] that computes an eigenvector u that corresponds to an approximation of a given eigenvalue μ of a given matrix. However, the dominant step of the RQI consists of matrix inversion that generally requires $\mathcal{O}(d^3)$ operations, and so the applicability of RQI to a sequence of dynamic (or time-varying) matrices largely depends on two factors — computational complexity of matrix inversion, and the length of the sequence. For a sequence of dynamic Laplacian matrices, $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_T\}$, the main advantage of the GraphRQI approach is to be able to compute the eigenvector matrix, U, very efficiently by combining the following optimizations:

- Recursively exploit sub-k matrix information.
- Exploit the sparsity and symmetry of Laplacian matrices to compute inverse Laplacian matrices efficiently.

At each time-step t, we compute k eigenvectors of $\mathcal{L}_t$. For each eigenvector, we perform an iterative process. We begin by initializing a random vector. Next, we iteratively perform the following update rule until it converges to an eigenvector. For the j^{th} eigenvector, the update rule is given as:

$$x_{\text{new}} \leftarrow \textbf{SM}\left(\delta\delta^\top, \left[\begin{array}{c|c} (1-\mu_j)\left(\textbf{SM}(\sigma\sigma^\top, \mathcal{L}_{t-1})\right) & 0 \\ \hline 0 & 1-\mu_j \end{array}\right]\right) x_{\text{old}} \tag{3.7}$$

where SM refers to the Sherman-Morrison formula [158] that computes the inverse of a sum of a matrix and an outer product, $xx^\top$, where $x \in \mathbb{R}^{d\times 1}$ is a sparse vector, μ_j is the approximate eigenvalue corresponding to which we compute an eigenvector, u_j. $\mathcal{L}_{t-1}, U_{t-1}$, and Λ_{t-1} are the Laplacian, spectrum, and corresponding diagonal eigenvalue matrix of the previous time-step. δ is a sparse $\mathbb{R}^d$ vector where, if the k^{th} entry of δ is denoted by $\delta(k)$, then the k^{th} road-agent observes a new neighbor if $\delta(k) = 1$.

3.6.2 Graph Spectrum Analysis

We compute the spectrum of the graph that describes the topologies of traffic at various time-step to classify aggressive and conservative behaviors.

Table 3.6: Analytical Comparison of GraphRQI with classical and state-of-the-art eigenvector algorithms. In practice, our observed runtime is 10 milliseconds which is a speed of up to 2.348 seconds over prior works for calculating the spectrums of dynamic traffic graph.

Metric	SVD	IncrementalSVD [159]	RestartSVD [160]	TruncatedSVD	CG [161]	**GraphRQI**
Time	$\mathcal{O}(d^3)$	$\mathcal{O}(\lvert\mathcal{L}_t\rvert r)$	$\mathcal{O}\Big(\lvert f(\mathcal{L}_t)\rvert + \lvert g(\mathcal{L}_t)\rvert k + \lvert g(\mathcal{L}_t)_{\bar{\emptyset}}\rvert k^2\Big)$	$\mathcal{O}(dk^2)$	$\mathcal{O}(d\sqrt{\kappa})$	$\mathcal{O}(\lvert\mathcal{L}_t^{-1}\rvert k)$
Space	$\mathcal{O}(d^2)$	$\mathcal{O}(2\ dr)$	-	$\mathcal{O}(dk)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$

In Table 3.6, we compare the time and space complexities of several state-of-the-art eigenvalue algorithms. For each of these methods, the input is a Laplacian matrix. We now state the theoretical guarantees for the running time and storage cost of our eigenvalue algorithm in the following theorems. All proofs are provided in the supplementary material[2].

Theorem 3.6.1. *Given a sequence, $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_T\}$, our eigenvalue algorithm for $\mathcal{L}_t$ converges cubically with a running time complexity of $\mathcal{O}(\lvert\mathcal{L}_t^{-1}\rvert k)$ for each Laplacian at time-step t, where $k \ll \dim(\mathcal{L}_t)$, with a storage cost of $\mathcal{O}(d)$.*

Proof. Our eigenvalue algorithm is motivated from the Rayleigh Quotient Iteration algorithm [157] (RQI). In RQI, to compute the eigenvector of any given matrix, A, corresponding to an approximation to a given eigenvalue, say η, the standard update formula for the RQI method is given as $x_{\text{new}} = (A - \eta I)^{-1} x_{\text{old}}$, where $x_{\text{old}} \in \mathbb{R}^{d \times 1}$ is initialized as a random vector. The efficiency of the update step depends on the efficiency of computing $(A - \eta I)^{-1}$. In GraphRQI, denoting $m = \lvert\mathcal{L}_t^{-1}\rvert$ as the number of non-zero elements of $\mathcal{L}_t^{-1}$, and $d = \dim\mathcal{L}_t$, for each of the k eigenvectors, the iterative update formula is: $x_{\text{new}} = (\mathcal{L}_t - \mu I)^{-1} x_{\text{old}}$. At first glance, it seems as if this update can be performed by applying the Sherman-Morrison (SM) update on the following decomposition:

$$(\mathcal{L}_t - \mu I)^{-1} = [\mathcal{L}_t - \mu\,(I_1 + I_2 + \ldots + I_d)]^{-1} \tag{3.8}$$

[2] `https://gamma.umd.edu/graphrqi`

That is, by expressing μI as a sum of d rank-1 matrices, I_j, where each I_j is a diagonal matrix with $I(j,j) = 1$ and 0 elsewhere. However, SM on $\mathcal{L}_t$ in the above equation requires $\mathcal{O}(d^3)$ flops (d^2 flops for each SM operation applied d times. Therefore, we look towards a better solution.

Observe that we can decompose $\mathcal{L}_t = \mathcal{L}_t^{'} + rr^\top$, where $\mathcal{L}_t^{'} = \left[\begin{array}{c|c} \mathcal{L}_{t-1}^r & 0 \\ \hline 0 & 1 \end{array}\right]$ and $rr^\top$ is an outer product of rank 2 with $r \in \mathbb{R}^{d\times 2}$ is a vector of 1's and 0's, where $r(j) = 1$ represents the addition of a new neighbor by the j^{th} road-agent. $\mathcal{L}_{t-1}^r$ denotes the laplacian matrix at the previous time-step plus a diagonal update on account of observing r new neighbors at the current time-step (This is further explained in lemma IV.2). So,

$$(\mathcal{L}_t - \mu I)^{-1} = (\mathcal{L}_t^{'} + rr^\top - \mu I)^{-1},$$

where $rr^\top$ is a sum of two rank-1 matrices. Therefore, if we can compute $(\mathcal{L}_t^{'} - \mu I)^{-1}$ in less than or equal to $\mathcal{O}(m)$, we are done. It is easy to verify that for any matrix A,

$$\left[\begin{array}{c|c} A & 0 \\ \hline 0 & 1 \end{array}\right]^{-1} = \left[\begin{array}{c|c} A^{-1} & 0 \\ \hline 0 & 1 \end{array}\right].$$

Therefore,

$$\left(\mathcal{L}_t^{'}\right)^{-1} = \left[\begin{array}{c|c} \mathcal{L}_{t-1}^r & 0 \\ \hline 0 & 1 \end{array}\right]^{-1} = \left[\begin{array}{c|c} \left(\mathcal{L}_{t-1}^r\right)^{-1} & 0 \\ \hline 0 & 1 \end{array}\right]$$

and thus,

$$(\mathcal{L}_t^{'} - \mu I)^{-1} = \left[\begin{array}{c|c} \left(\mathcal{L}_{t-1}^r - \mu I\right)^{-1} & 0 \\ \hline 0 & (1-\mu) \end{array}\right].$$

We can now use Lemma IV.2 to show that computing the inverse of the laplacian, $\left(\mathcal{L}_{t-1}^{r} - \mu I\right)$ can be performed in $\mathcal{O}(m)$. The entire update can be summarized as follows,

$$x_{\text{new}} \leftarrow \textbf{SM}\left(rr^{\top}, \left[\begin{array}{c|c} (1-\mu_j)\left(\textbf{SM}(\sigma\sigma^{\top}, \mathcal{L}_{t-1})\right) & 0 \\ \hline 0 & 1-\mu_j \end{array}\right]\right) x_{\text{old}} \tag{3.9}$$

$\sigma\sigma^{\top} = \Sigma$.

At each time-step, we add a new column and a row to the laplacian matrix of the previous time-step. Therefore, the space complexity grows linearly with the dimension of the laplacian matrix. Finally, the cubic convergence follows from the classic RQI algorithm [157].

□

Informally, Theorem 3.6.1 states that for a Laplacian matrix $\mathcal{L}_t$, the compuatation for each eigenvector requires $\mathcal{O}(|\mathcal{L}_t^{-1}|)$ operations, which is fewer than quadratic runtime algorithms that require $\mathcal{O}(d^2)$ operations, since $\mathcal{O}(|\mathcal{L}_t^{-1}|) \ll d^2$. For a random intial iterate, x, our eigenvalue algorithm converges cubically, that is, at each iteration, $|x_{\text{new}} - x_{\text{old}}| = \mathcal{O}(\epsilon^3)$.

Note that some eigenvalue algorithms make several assumptions in terms of running time complexity analysis. For example, the conjugate gradient method [161] requires $\mathcal{O}(d\sqrt{\kappa})$ operations. The success of conjugate gradient assumes the matrix is either well-conditioned or pre-conditioned by a preconditioning matrix, for which the running time becomes $\mathcal{O}(d^{1.5})$. IncrementalSVD [159] requires the matrix to be low-rank. In the general case, IncrementalSVD requires $\mathcal{O}(|\mathcal{L}_t|d)$ which becomes $\mathcal{O}(|\mathcal{L}_t|r)$ for low rank matrices with rank r. Our eigenvalue algorithm makes no such

assumptions. We now state a lemma, related to our Laplacian matrix, that is used to prove Theorem 3.6.1.

Lemma 3.6.1. *The complexity of computing the inverse of a Laplacian at time-step t, $\mathcal{L}_t^{-1}$, grows as $\mathcal{O}(|\mathcal{L}_t^{-1}|)$.*

Proof. Note that $\mathcal{L}_{t-1}^r \neq \mathcal{L}_{t-1} \implies \left(\mathcal{L}_{t-1}^r\right)^{-1} \neq \mathcal{L}_{t-1}^{-1}$ since the diagonal elements (the degree of each node (vehicle)) are now different due the entry of new nodes at current time t. Let the number of new nodes be r. For example, if at the current time-step, 10 cars from the previous time-step observed a new neighbor, then $r = 10$. Since we heuristically set the time-step and the KNN parameters, we constrain each vehicle observing at most 1 new neighbor for each time-step. Therefore,

$$\mathcal{L}_{t-1}^r = \mathcal{L}_{t-1} + \Sigma,$$

where $\Sigma = I_1 + I_2 + \ldots + I_r$ is a low-rank matrix with rank $r \ll d$, where we treat r as a constant due to our earlier assumption. Each I_j is a diagonal matrix with $I(j,j) = 1$ and 0 elsewhere. Hence,

$$\begin{aligned} \left(\mathcal{L}_{t-1}^r - \mu I\right) &= \mathcal{L}_{t-1} - \mu I + \Sigma \\ \left(\mathcal{L}_{t-1}^r - \mu I\right)^{-1} &= \left(\mathcal{L}_{t-1} - \mu I + I_1 + I_2 + \ldots + I_r\right)^{-1}, \end{aligned} \tag{3.10}$$

where $(\mathcal{L}_{t-1} - \mu I)^{-1} = U_{t-1}(\Lambda - \mu I)^{-1} U_{t-1}^\top$. Using the Sherman-Morrison formula r times on $(\mathcal{L}_{t-1} - \mu I)$, we get $\left(\mathcal{L}_{t-1}^r - \mu I\right)^{-1}$ in $\mathcal{O}(m)$.

□

The current best-case running-time complexity for inverting a $d \times d$) matrix is $\mathcal{O}(d^{2.373})$ using a variation of the Coppersmith–Winograd algorithm [162]. We show that we can achieve an even lower runtime complexity for inverting our matrix in $\mathcal{O}(|\mathcal{L}_t^{-1}|)$, where $\mathcal{O}(|\mathcal{L}_t^{-1}|) \ll d^2$.

3.6.3 Behavior Classification

The graphs are constructed from the road-agent trajectories as described in Section 3.3. We then pass the spectrum of the traffic graphs as inputs to train a Multi-Layer Perceptron(MLP) [156] for behavior classification. The MLP is parametrized by a weight vector, w. Our goal is to learn the optimal parameters that minimize the squared error loss,

$$f(w) = \frac{1}{2} \sum_{k=1}^{d} \left(w^\top y_k - z_k \right)^2 \tag{3.11}$$

where d is the number of vehicles, y_k is the k^{th} row of the eigenvector matrix, U and corresponds to the feature vector of the k^{th} vehicle. The corresponding label of the k^{th} vehicle is denoted as z_k. In our experiments, d is around 100 for each traffic video.

3.6.4 Running Time Evaluation

Table 3.7: Ablation study on TRAF and ARGO datasets. We perform a running time analysis of several eigenvalue algorithms. All experiments were performed on an 8 Core Intel Xeon(R) W2123 CPU clocked at 3.60GHz with 32 GB RAM to compute the eigenvectors for a $d \times d$ matrix, where d is the number of road-agents. We also compare the accuracy of different supervised learning machine learning models and report the weighted classification accuracy.

	Eigenvalue Algorithms (Runtime)						GraphRQI + ML model (Accuracy)			
Dataset	SVD	IncrementalSVD [159]	RestartSVD [160]	TruncatedSVD	CG	**GraphRQI**	LogReg	SVM	LSTM	**GraphRQI**
TRAF	34.2ms	67ms	1,644ms	37ms	2,365ms	**16.9ms**	69.1%	74.2%	71.4%	**78.3%**
ARGO	16.9ms	45ms	1,091ms	35ms	2,328ms	**10ms**	69.9%	75.0%	70.8%	**89.9%**

For fair evaluation, we use the same programming platform (Python 3.7) and the same

processor for running all the different methods (8 Core Intel Xeon(R) W2123 CPU clocked at 3.60GHz with 32 GB RAM). We empirically validate the theoretical guarantees of Theorem 3.6.1 by replacing our eigenvalue algorithm with several standard eigenvector algorithms, Singular Value Decomposition (SVD), Incremental SVD[159], Restart SVD [160], Truncated SVD, and the state-of-the-art iterative method Conjugate Gradient Descent [161]. For SVD, and TruncatedSVD, we directly used the library routine implemented in the SCIPY package, and for RestartSVD, we used the authors' original implementation. In practice, our method outperforms all these standard methods for Driver Behavior Classification. These results are shown in Table 3.7.

3.7 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.

Chapter 4: Behaviorally Compliant Planning in Human Environments

4.1 Overview

The navigation problem for autonomous vehicles (AVs) corresponds to computing the optimal actions that enable the AV to begin from starting point A and reach destination B via a smooth trajectory while avoiding collisions with dynamic obstacles or traffic agents. A key aspect of navigation is safety because the AVs are expected to keep a safe distance from other vehicles while also making driving more fuel- and time-efficient. Navigation is a central task in autonomous driving, and navigation problems have also been studied extensively in the contexts of motion planning and mobile robots.

There is considerable research on designing prediction and navigation algorithms for autonomous driving, but these algorithms are currently primarily deployed in specific driving scenarios [163] or low-density traffic [164]. Some of these algorithms are intentionally designed to excel in unique cases [165], while others are inevitably limited because they are based on data-driven methods trained on selected datasets with specific driving conditions [67]. Modern prediction and navigation algorithms must be able to handle various driving scenarios to comply with real-world situations. One of these scenarios is dense traffic, which is commonly observed in city centers or in the vicinity of frequent or popular destinations. There are many challenges in terms of handling dense traffic environments, including computing safe and collision-free trajectories, and

modeling the interactions between the traffic-agents. Some key issues are related to evaluating the driving behaviors of human drivers and ensuring that the driving pattern of the AV is consistent with traffic norms. It has been observed that current AVs tend to drive hyper-cautiously or in ways that can frustrate other human drivers [118], potentially leading to fender-benders.

Moreover, it is important to handle the unpredictability or aggressive nature of human drivers. For example, human drivers may act irrationally and move in front of other vehicles by suddenly changing lanes or aggressively overtaking. In 2016, Google's self-driving car had a collision with an oncoming bus during a lane change [123]. In this case, the ego-vehicle assumed that the bus driver was going to yield; instead, the bus driver accelerated. Overall, we need better prediction and navigation methods that can account for such behaviors and more diverse datasets enriched with these behaviors so that learning-based methods can produce results that are more applicable to real-world scenarios.

There is considerable work on classifying driver behaviors or styles in traffic psychology [75, 166, 167]. In most cases, driving style is defined with respect to either aggressiveness [168] or fuel consumption [169]. Many recent methods have been proposed to classify driving behaviors based on past trajectories [14, 79, 81] and use them for predicting the vehicle's future trajectory [104, 138]. On the other hand, recent techniques for prediction and navigation have been based on reinforcement learning, and they tend to learn optimal policies with suitable rewards for collision handling, lane changing, and traffic rule adherence. Our goal is to extend these learning methods to account for driver behaviors.

Table 4.1: **Summary of prior work:** We list methods for navigating unsignaled intersections, roundabouts, and merging based on multi-agent planning (MAP), action space (AS), and incentive compatibility (IC). ✓* corresponding to a method indicates that optimality does not hold for human drivers with varying social preferences.

Approach	Methods	Optimality	MAP	AS	IC	Real world
DRL	Capasso et al. [170]	✗	✓	C	-	✗
	Isele et al. [171]	✗	✗	D	-	✗
	Kai et al. [172]	✗	✗	D	-	✓
	Liu et al. [173]	✗	✗	D	-	✗
Game Theory	Li et al. [174]	✗	✓	C	-	✗
	Tian et al. [175]	✗	✓	C	-	✗
RNN	Roh et al. [176]	✗	✓	C	-	✗
Auctions	FIFO	✗	✓	D	-	✗
	Buckman et al. [177]	✗	✓	D	-	✗
	Vasirani and Ossowski [178]	✗	✓	D	✗	✗
	Censi et al. [179]	✗	✓	D	✗	✗
	Lin and Jabari [180]	✗	✓	D	✗	✗
	Carlino et al. [181]	✓*	✓	D	✓	✗
	Rey et al. [182]	✓*	✓	D	✓	✗
	Sayin et al. [183]	✓*	✓	D	✓	✗
	This work	✓	✓	D	✓	✓

4.2 Prior Work

In Table 4.1, we compare our approach with the current state-of-the-art in navigating unsignaled intersections, roundabouts, and merging scenarios on the basis of optimality guarantees, multi-agent versus single-agent planning (MAP), action space (AS), incentive compatibility (IC), and real-world applicability.

4.2.1 Deep reinforcement learning (DRL)

DRL-based methods [114, 170, 171, 172, 173] learn a navigation policy using the notion of expected reward received by an agent from taking a particular action in a particular state. This policy is learned from trajectories obtained via traffic simulators using Q-learning and is very hard as well as expensive to train. In practice, DRL-based planning methods often do not generalize well to different environments and it is hard to provide any guarantees. Furthermore, these methods discussed so far are intended for single-agent navigation. However, Capasso et al. [170] use additional signals such as traffic signs (stop, yield, none) to regulate the movement and actions of multiple agents. In terms of real world applications, Kai et al. [172] learn a unified policy for multiple tasks and also demonstrate their approach on a real robot.

4.2.2 Game theory

Game-theoretic approaches [174, 175], by their very nature, perform multi-agent planning. However, these methods restrict the actions and objectives of active agents. For example, Li et al. [174] formulate traffic intersection planning as a stackelberg leader-follower game in which one agent is assumed to act first (leader) and the other agent (follower) will react accordingly.

The leader is decided by a first-in first-out (FIFO) principle. Tian et al. [175] use a recursive $k-$level game-theoretic approach in which complex strategies for agents at each level are derived from previous levels. However, all agents except the ego-agent at the first level are assumed to be static.

4.2.3 Recurrent neural networks (RNNs)

Deep learning-based methods [176] train a recurrent neural network for trajectory prediction and are also susceptible to complex environments or different behavior of drivers. In contrast to all of the above, GamePlan do not require an objective function; instead, successful application of this approach requires minimizing a loss function that depends on the distribution of the data.

4.2.4 Auctions

A very basic principle to regulate traffic at intersections is the FIFO principle which basically states that agents move in the order in which they arrive at the intersection. The main concern with such an approach is that aggressive or impatient drivers are incentivised to break form and move out of turn. Additionally, in situations when drivers arrive at an intersection at the same time, oftentimes deadlocks occur as a result of ensuing confusion among the drivers. Therefore, FIFO is not guaranteed to be optimal. Buckman et al. [177] integrate a driver behavior model [75] within the basic FIFO framework to incorporate human social preference to address some of the above limitations of FIFO. But the model does not estimate the social preferences in real time; instead, it chooses a fixed preference parameter for each agent.

Vasirani and Ossowski [178] proposed a combinatorial auction for assigning turns at intersections.

Censi et al. [179] introduced a karma-based auction and Lin and Jabari [180] proposed a mechanism for pricing intersection priority based on transferable utility games. However, these auctions are not incentive-compatible. Incentive-compatible auctions such as Sayin et al. [183] propose a mechanism in which agents are assigned turns based on their distance from the intersection and the number of passengers in the vehicle. Carlino et al. [181] and Rey et al. [182] propose a similar mechanism but use a monetary-based bidding strategy. These methods are, however, biased towards wealthier agents disregarding human preference, limiting their use in the real-world applications.

4.3 Planning at Unsignaled Intersections, Roundabouts, and Merging

In this section, we begin by defining several key terms and stating the problem statement and assumptions made in our approach (Section 4.3.1). We briefly summarize the CMetric algorithm [14] for behavior modeling and prediction that we use in GamePlan in Section 4.3.2. Finally, we provide an overview on sponsored search auctions [184] in Section 4.3.3 which forms the basis of the auction used in GamePlan.

4.3.1 Problem Formulation

We frame navigating unsignaled intersections, roundabouts, and merging scenarios as a multi-agent game-theoretic non-sequential decision-making problem. We consider three unsignaled and uncontrolled traffic scenarios or environments – intersections, merging, and roundabouts with multiple non-communicating human drivers and AVs. In each of the three scenarios, we refer to a traffic-agent as "active" if they participate in the decision-making process for navigating the

scenario. For example, the lead traffic-agents in each service lane of a four-way intersection are active vehicles, while agents waiting in line behind their respective lead vehicle are non-active agents. We denote the active agents as a set $\mathcal{A} = \{a_1, a_2, \ldots a_n\}$ where each a_i participates in the GamePlan auction. The action space for each active agent is a discrete set consisting of finite acceleration values. The state space consists of the position $x_t \in \mathcal{X}$ and velocity v_t of each agent measured in a global coordinate frame (*e.g.* GPS coordinates or 3D points obtained using GPS and lidars).

Problem Statement: To prevent collisions and deadlocks, it is necessary to determine the order in which agents take turns to navigate the traffic scenario [178, 180, 181, 183, 185, 186]. Thus, the goal of this paper is to compute an optimal turn-based ordering which is defined as follows, There may be multiple turn-based orderings that may apply in a given scenario. However, many of them could be sub-optimal and may result in collisions and deadlocks. An optimal turn-based ordering prevents collisions and resolves deadlock conflicts.

We are now ready to formally state our goal in this paper.

Assumptions: We assume agents (AVs and human drivers) are rational and strive to maximize their own utility which can be independent of other agents. Further, agents are *non-ideal* (in contrast to ideal agents in [13, 187, 188]) in that they do not have access to the actions, objectives, or utility functions of other agents. We make no assumptions on the motion or dynamics of traffic-agents. The state-space is fully observable by all agents (active and non-active). We assume the availability of a behavior modeling and prediction algorithm such as CMetric [14] that models driving behavior as being either aggressive or conservative. This information is provided to an auction program. We assume that agents navigate through the environment one at a time and do not use traffic signals, employ right-of-way rules, or communicate with other agents.

4.3.2 Modeling human driver behavior

Each active agent is characterized by a behavior profile, $\boldsymbol{\zeta}_i$, which can be obtained via recent behavior modeling algorithms such as the SVO [75] and CMetric [14]. In this work, we use CMetric to quantify the aggressiveness or conservativeness of a traffic-agent. This model provides an objective measure of aggressiveness based on driving maneuvers such as overspeeding, overtaking and so on. We briefly summarize the CMetric algorithm. To determine if an agent is aggressive or conservative, the algorithm begins by reading the trajectories of the ego-agent and surrounding vehicles via cameras or lidars during any time-period Δt. The trajectory of an agent is represented by

$$\Xi_{\Delta t} = \{x_t \mid t = t_0, t_1, \ldots, t_0 + \Delta t\}.$$

In CMetric, these trajectories are represented via weighted undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which the vertices denote the positions for each agent and the edges correspond to the distances between agents. The algorithm proceeds by using these graphs to model the likelihood and intensity of driving behavior indicators like overspeeding, overtaking, sudden "zig-zagging" and lane-changes via centrality functions [124] represented by $\Phi : \mathcal{G} \rightarrow \mathbb{R}$. These behavior indicators determine whether an agent is aggressive or conservative. The behavior profile for a_i is denoted by $\boldsymbol{\zeta}_i$ and is computed as,

$$\boldsymbol{\zeta}_i(\Xi_{\Delta t}) = \Phi^i(\mathcal{G})[t], \tag{4.1}$$

where $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is constructed using $\Xi_{\Delta t}$. The original definition of centrality, however, does not

take into account temporal memory since the the centrality value changes with time. In order to model driver behavior during a time period ΔT, we must keep track of all the neighbors the vehicle has interacted with during that period. $\Phi(\mathcal{G})[t]$ is defined as follows,

Definition 4.3.1. ***Temporal memory for*** $\boldsymbol{\Phi(\mathcal{G})}$*: In a connected traffic-graph* $\mathcal{G}$ *at time* t *with associated adjacency matrix* A_t*, let* $\mathcal{N}_i(t) = \{v_j \in \mathcal{V}(t),\ A_t(i,j) \neq 0, \nu_j \leq \nu_i\}$ *denote the set of vehicles in the neighborhood of the* i^{th} *vehicle with radius* μ*, then the discrete degree centrality function of the* i^{th} *vehicle at time* t *is defined as,*

$$\begin{gathered}\Phi^i[t] = \Phi^i\left(\{v_j \in \mathcal{N}_i(t)\}\right) + \Phi^i[t-1] \\ \textit{such that } (v_i, v_j) \notin \mathcal{E}(\tau), \tau = 0, \ldots, t-1\end{gathered} \tag{4.2}$$

where $|\cdot|$ *denotes the cardinality of a set and* ν_i, ν_j *denote the velocities of the* i^{th} *and* j^{th} *vehicles, respectively.*

In this work, we use CMetric to compute the behavior profiles of traffic-agents, represented by a $n-$dimensional vector $\boldsymbol{\zeta}$, that is provided to the auction program.

4.3.3 Sponsored search auctions (SSAs)

Sponsored search auctions (SSAs) are a game-theoretic mechanism that are used extensively in internet search engines for the purpose of internet advertising [184]. In an SSA, there are K items to be allocated among n agents. Each agent a_i has a private valuation $\boldsymbol{v}_i$ and submits a bid $\boldsymbol{b}_i$ to receive at most one item of value $\boldsymbol{\alpha}_i$. A strategy is defined as an n dimensional vector, $\boldsymbol{b} = (\boldsymbol{b}_i \cup \boldsymbol{b}_{-i})$, representing the bids made by every agent. $\boldsymbol{b}_{-i})$ denotes the bids made by all agents except a_i. Furthermore, let $\boldsymbol{b}_1 > \boldsymbol{b}_2 > \ldots > \boldsymbol{b}_K$ and $\boldsymbol{\alpha}_1 > \boldsymbol{\alpha}_2 > \ldots > \boldsymbol{\alpha}_K$. The

allocation rule is that the agent with the i^{th} highest bid is allocated the i^{th} most valuable item, $\boldsymbol{\alpha}_i$. The utility $\boldsymbol{u}_i$ [184] incurred by a_i is given as follows,

$$\boldsymbol{u}_i(\boldsymbol{b}_i) = \boldsymbol{v}_i\boldsymbol{\alpha}_i - \sum_{j=i}^{k} \boldsymbol{b}_{j+1}\left(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_{j+1}\right). \tag{4.3}$$

In the equation above, the quantity on the left represents the total utility for a_i which is equal to value of the allocated goods $\boldsymbol{\alpha}_i$ minus a payment term. The first term on the right is the value of the item obtained by a_i. The second term on the right is the payment made by a_i as a function of bids $\boldsymbol{b}_{j>i}$ and their allocated item values $\boldsymbol{\alpha}_j$. We refer the reader to Chapter 3 in [184] for a derivation and detailed analysis of Equation 4.3.

In our approach, we re-cast Equation 4.3 through the lens of a human driver. More specifically, the term $\boldsymbol{v}_i\boldsymbol{\alpha}_i$ denotes the time reward gained by driver a_i by moving on her turn. The payment term represents a notion of risk [13] associated with moving on that turn. It follows that an allocation of a conservative agent to a later turn (smaller $\boldsymbol{\alpha}$) also presents the lowest risk and vice-versa.

Choosing an optimal ordering, in which agents navigate unsignaled and uncontrolled traffic scenarios, can be cast as an allocation problem where the goal is to allocate each agent, a_i, a position in the optimal turn-based ordering ($\boldsymbol{\sigma}_i$). Deciding such an allocation depends heavily on the incentives of the agents which, in the case of non-ideal agents, is a hard problem. Prior planning methods model non-ideal agents by estimating the objective functions of the agents from noisy data using statistical methods [122, 175, 189] or by assuming a fixed behavior for surrounding agents (static or constant velocity) [174, 175]. These methods are not guaranteed to be optimal and result in collisions and deadlocks in unsignaled traffic scenarios, as shown in

Table 3.3.

Auction-based methods, on the other hand, model non-ideal agents in unsignaled traffic scenarios effectively albeit using a monetary-based bidding strategy that is not realizable in real-world scenarios [178, 180, 181, 183]. Our formulation, GamePlan, differs in this regard wherein we use a novel online driving behavior-based bidding strategy using the CMetric model [14]. In the rest of this section, we present the main algorithm followed by an analysis of its optimality.

4.3.4 Algorithm

Our goal is to solve Problem 3.2.1 and compute the optimal turn-based ordering $\boldsymbol{\sigma}_{\text{OPT}} = \boldsymbol{\sigma}_1\boldsymbol{\sigma}_2 \ldots \boldsymbol{\sigma}_n$, which shall determine the order in which agents will navigate unsignaled intersections, roundabouts, or merging. Our GamePlan algorithm proceeds in two stages: the behavior modeling phase and the planning phase.

During the behavior modeling phase, we use CMetric to compute the behavior profiles $\boldsymbol{\zeta}_i$ for every agent (active or non-active) using Equation 4.1 during an observation period of 5 seconds. However, alternative behavior models such as SVO [75] may be used. This is followed by the planning phase which runs a sponsored search auction (SSA) scheme. In the auction scheme, each active agent has a private valuation $\boldsymbol{v}_i$. Each agent a_i submits a bid $\boldsymbol{b}_i \in \mathbb{R}^{\geq 0}$ and obtains a time reward of $\frac{1}{t_i}$ for completing the navigation task in t_i seconds, measured from the time the first agent begins to move. Note that moving earlier corresponds to a higher time reward.

To summarize the algorithm, the agent with the highest bid *i.e.* most aggressive behavior is allocated the highest priority and is allowed to navigate the scenario first, followed by the second-most aggressive, and so on. Therefore,

$$(\boldsymbol{\sigma}_{\text{OPT}})_i = j^\star, \tag{4.4}$$

where $j^\star$ is the index of $\boldsymbol{\zeta}_{j^\star}$ in the sequence $\boldsymbol{\zeta}_1 > \boldsymbol{\zeta}_2 > \ldots > \boldsymbol{\zeta}_{j^\star} > \ldots > \boldsymbol{\zeta}_K$.

4.3.5 Game-theoretic optimality and efficiency analysis

In this section, we show that our approach is incentive compatible, welfare maximizing, and can be computed in polynomial time.

4.3.5.1 Incentive compatibility

The goal of any optimal auction should be such that that no agent is incentivised to "cheat" or, more simply, when the dominant strategy for each agent is to bid their true valuation $\boldsymbol{v}_i$. We define a dominant strategy as,

Definition 4.3.2. ***Dominant Strategy:*** *Bidding $\mathbf{b}_i$ is a dominant strategy for a_i if $\boldsymbol{u}_i(\boldsymbol{b}_i, \boldsymbol{b}_{-i}) > \boldsymbol{u}_i(\bar{\boldsymbol{b}}_i, \boldsymbol{b}_{-i})$ for all $\bar{\mathbf{b}}_i \neq \mathbf{b}_i$.*

Ensuring fair allocations is crucial for auctions applied to traffic scenarios since unfair allocations could result in collisions and deadlocks. Incentivising traffic-agents to bid their true value as a dominant strategy is known as incentive compatibility [181, 182, 183, 184] which is defined as follows,

Definition 4.3.3. ***Incentive Compatibility:*** *An auction is said to be incentive compatible if for each agent, bidding $\mathbf{b}_i = \boldsymbol{v}_i$ is a dominant strategy.*

We want to show that σ_{OPT} is incentive compatible, maximizes welfare, and can be computed

in polynomial time. Incentive compatibility ensures that the best action

In our formulation, we set the true valuation ($\boldsymbol{v}_i$) for a traffic-agent to be equal to its behavior profile $\boldsymbol{\zeta}_i$. Hence,

$$\boldsymbol{v}_i = \boldsymbol{\zeta}_i.$$

And so to show that our auction is incentive-compatible, we show the following, We verify these properties through the following analysis.

Theorem 4.3.1. *For each active agent $a_i \in \mathcal{A}$ at a traffic intersection, roundabout, or during merging, bidding $\boldsymbol{b}_i = \boldsymbol{\zeta}_i$ is the dominant strategy.*

We defer the proof to the supplementary material.

Proof. Recall that the k^{th} highest bidder (k^{th} most aggressive agent) receives a time reward $\boldsymbol{\alpha}_k = \frac{1}{t_k}$. Then according to Equation 4.3, the overall utility achieved by the k^{th} most aggressive traffic-agent is,

$$\boldsymbol{u}_k(\boldsymbol{b}_k) = \boldsymbol{\zeta}_k \left(\frac{1}{t_k}\right) - \sum_{j=k}^{K} \boldsymbol{b}_{j+1} \left(\frac{1}{t_j} - \frac{1}{t_{j+1}}\right).$$

We sort the K highest bids received in the following order: $\boldsymbol{b}_1 > \boldsymbol{b}_2 > \ldots > \boldsymbol{b}_K$. In order to show that $\boldsymbol{b}_k = \boldsymbol{\zeta}_k$ is the dominant strategy, it is sufficient to show that over-bidding ($\bar{\boldsymbol{b}}_k > \boldsymbol{\zeta}_k$) and under-bidding ($\bar{\boldsymbol{b}}_k < \boldsymbol{\zeta}_k$) both result in a lower utility than $\boldsymbol{u}_k$. We proceed by analyzing both cases.

Case 1: Over-bidding ($\bar{\boldsymbol{b}}_k = \boldsymbol{b}_{k-1} > \boldsymbol{b}_k$): In this case, the new utility for a_k is $\bar{\boldsymbol{u}}_k(\bar{\boldsymbol{b}}_k)$ which is

equal to,

$$\boldsymbol{\zeta}_k \left(\frac{1}{t_{k-1}} \right) - \boldsymbol{b}_k \left(\frac{1}{t_{k-1}} - \frac{1}{t_k} \right) - \sum_{j=k}^{K} \boldsymbol{b}_{j+1} \left(\frac{1}{t_j} - \frac{1}{t_{j+1}} \right). \tag{4.5}$$

From Equation 4.3 and Equation 4.5, the net increase in utility is,

$$\bar{\boldsymbol{u}}_k(\bar{\boldsymbol{b}}_k) - \boldsymbol{u}_k(\boldsymbol{b}_k) = (\boldsymbol{\zeta}_k - \boldsymbol{b}_k) \left(\frac{1}{t_{k-1}} - \frac{1}{t_k} \right). \tag{4.6}$$

Therefore, bidding $\bar{\boldsymbol{b}}_k > \boldsymbol{\zeta}_k \implies \bar{\boldsymbol{u}}_k(\bar{\boldsymbol{b}}_k) - \boldsymbol{u}_k(\boldsymbol{b}_k) < 0$ since $t_{k-1} < t_k$. In other words, overbidding yields negative utility for agent a_k.

Case 2: Under-bidding ($\bar{\boldsymbol{b}}_k = \boldsymbol{b}_{k+1} < \boldsymbol{b}_i$): The new utility in this case is given by,

$$\bar{\boldsymbol{u}}_k(\bar{\boldsymbol{\zeta}}_k) = \boldsymbol{\zeta}_k \left(\frac{1}{t_{k+1}} \right) - \sum_{j=k+1}^{K} \boldsymbol{b}_{j+1} \left(\frac{1}{t_j} - \frac{1}{t_{j+1}} \right) \tag{4.7}$$

From Equation 4.3 and Equation 4.7, the net decrease in utility is,

$$\boldsymbol{u}_k(\boldsymbol{b}_k) - \bar{\boldsymbol{u}}_k(\bar{\boldsymbol{b}}_k) = (\boldsymbol{\zeta}_k + \boldsymbol{b}_{k+1}) \left(\frac{1}{t_k} - \frac{1}{t_{k+1}} \right). \tag{4.8}$$

Note that Equation 4.8 is always positive since $\boldsymbol{\zeta}, \boldsymbol{b}_{k+1} > 0$ and $t_k < t_{k+1}$. This implies that under-bidding always results in a decrease in utility as well.

□

4.3.5.2 Welfare maximization

The next desired property in an optimal auction is welfare maximization [183, 184] which maximizes the total utility earned by every active agent.

Theorem 4.3.2. ***Welfare maximization:*** *Social welfare of an auction is defined as* $\sum_i \boldsymbol{v}_i \boldsymbol{\alpha}_i$. *Welfare maximization involves finding the strategy* $\boldsymbol{b}$ *that maximizes* $\sum_i \boldsymbol{v}_i \boldsymbol{\alpha}_i$. *For each active agent* $a_i \in \mathcal{A}$, *bidding* $\boldsymbol{b}_i = \boldsymbol{\zeta}_i$ *maximizes social welfare.*

We defer the proof to the supplementary material.

Proof. Our proof is based on induction. We begin with the base case with the most aggressive agent (highest bidder). Recall that after sorting, we have agents in decreasing order of aggressiveness *i.e.* $\boldsymbol{\zeta}_1 > \boldsymbol{\zeta}_2 > \ldots > \boldsymbol{\zeta}_n$ and $\frac{1}{t_1} > \frac{1}{t_2} > \ldots > \frac{1}{t_k}$. Therefore, we have that $\frac{\boldsymbol{\zeta}_1}{t_1}$ is maximum. Next, consider the hypothesis that the sum $\sum_{j=1}^{k} \frac{\boldsymbol{\zeta}_j}{t_j}$ is maximum up to the k^{th} highest bidder. Then the inductive step is to prove that $\sum_{j=1}^{k+1} \frac{\boldsymbol{\zeta}_j}{t_j}$ is maximum. Observe that,

$$\sum_{j=1}^{k+1} \left(\frac{\boldsymbol{\zeta}_j}{t_j} \right) = \sum_{j=1}^{k} \left(\frac{\boldsymbol{\zeta}_j}{t_j} \right) + \frac{\boldsymbol{\zeta}_{k+1}}{t_{k+1}}$$

Note that the first term on the RHS is maximum from hypothesis. Then,

$$\begin{gathered} \boldsymbol{\zeta}_{k+1} > \boldsymbol{\zeta}_{k+2} > \boldsymbol{\zeta}_{k+3} > \ldots > \boldsymbol{\zeta}_n \\ \text{and} \\ \frac{1}{t_{k+1}} > \frac{1}{t_{k+2}} > \frac{1}{t_{k+3}} > \ldots > \frac{1}{t_K} \end{gathered} \tag{4.9}$$

implies that $\frac{\zeta_{k+1}}{t_{k+1}}$ is maximum.

□

4.3.5.3 Polynomial time computation

Finally, in terms of planning and auction design [184], it is important to show that the underlying auction is computationally efficient and can handle a large number of agents. We show that our approach runs in polynomial time via the following theorem,

Theorem 4.3.3. ***Polynomial Runtime:*** *GamePlan runs in polynomial time.*

Proof. The main computation in our algorithm is dominated by sorting the agent's CMetric values; it is known that sorting algorithms run in polynomial time [190]. □

4.3.6 Using σ_{OPT} for collision prevention and deadlock resolution

We identify a deadlock as a situation when two or more active traffic-agents remain stationary for an extended period of time due to the uncertainty in the actions of other active traffic-agents. Deadlocks may arise in traffic scenarios consisting of multiple conservative and/or aggressive agents and are resolved when one of the agents opts to move based on some heuristic. Via $\boldsymbol{\sigma}_{\text{OPT}}$, agents automatically know when each agent is supposed to move thereby eliminating any confusion or uncertainty in the actions of other agents.

GamePlan can also prevent collisions in a similar manner. The number of collisions, or the likelihood thereof, increases when two or more aggressive agents decide to `MOVE` first, simultaneously, despite the uncertainty in the actions of the other agents. $\boldsymbol{\sigma}_{\text{OPT}}$ can break ties between multiple aggressive drivers since $\boldsymbol{\zeta}_i \neq \boldsymbol{\zeta}_j$ for $i, j \in [1, n]$. Using the turn-based ordering

determined by $\boldsymbol{\sigma}_{\text{OPT}}$, less aggressive agents can let more aggressive agents pass first.

4.3.7 Conclusion, Limitations, and Future Work

We present a novel multi-agent game-theoretic planning algorithm called GamePlan in intersections, roundabouts, and during merging with human drivers and autonomous vehicles. GamePlan uses the behavior profiles of all traffic-agents, combines with sponsored search auctions, and produces an optimal turn-based ordering. We show that GamePlan is incentive compatible, welfare maximizing, and operates in polynomial time. We reduce the number of collisions and deadlocks by at least $10 - 20\%$ on average over prior methods. Moreover, we demonstrate GamePlan in two merging scenarios involving real human drivers and show that our game-theoretic model is applicable in real-world scenarios.

There are a few limitations of our work. Our approach is primarily designed for moderately to highly dense traffic as CMetric [14] may not work as well in sparse traffic conditions. In such cases, data-driven behavior models such as SVO [75] may be used. There are many interesting directions for future work. For example, our method currently does not plan beyond computing turn-based orderings, i.e. local navigation. Next steps may include integrating GamePlan with global motion planning methods to achieve an end-to-end navigation approach for non-communicating multi-agent traffic scenarios. In addition, we have currently demonstrated real world application with $2 - 3$ vehicles. In the future, we plan to conduct further evaluation in denser and more comprehensive real world settings with more vehicles.

4.4 Risk-Aware Planning

Risk-aware planning involves sequential decision-making in dynamic and uncertain environments, where agents must consider the risks associated with their actions and corresponding costs and rewards [191]. *Risk-seeking* agents are willing to take lower expected reward in exchange for a higher reward variance (more risk), while *risk-averse* agents are willing to take a lower expected reward in exchange for lower reward variance (less risk). Agents that are risk-averse or risk-seeking are collectively referred to as risk-aware. Human drivers are risk-aware by nature [192, 193, 194]. For example, aggressive drivers frequently speed, overtake, and perform sharp cut-ins, whereas conservative drivers drive more cautiously. To navigate successfully among human drivers, autonomous vehicles (AVs) must identify the risk preferences of human drivers online, and predict and plan future motion with the risk preferences of all agents in mind, including the AV's own risk preferences.

The most common risk measures utilized in risk-sensitive planning are entropic risk [195] and conditional value at risk (CVaR) [196]. A popular approach for risk-aware planning in multi-agent traffic scenarios is to model risk-aware agent interactions via dynamic games [13] wherein agents act while considering their impact on other agents as well as the intentions of the other agents. In [13], the authors compute the Nash equilibrium solution of the game by iteratively solving a set of LEQ equations [197, 198]. The main benefits of this approach compared to prior risk-aware planning methods include improved time-to-goal and, more importantly, generation of emergent behaviors. For instance, risk-averse agents learn to maintain a greater distance from risk-seeking agents and generally yield more frequently to risk-seeking agents at intersections, at roundabouts, and during merging.

Despite its performance and benefits, the main drawback of the approach proposed by [13] is that it does not model the risk tolerance of human drivers, as it assumes the AV knows the synthetically chosen risk tolerances for all other driving agents). Extending game-theoretic risk-aware planning to human drivers will allow AVs to act more confidently around human drivers and reduce time-to-goal via more efficient and safer navigation. Estimating the risk tolerances of human drivers, however, requires computationally tractable human driver behavior models that can characterize drivers.

Some of the state-of-the-art approaches for modeling human driver behavior [75, 119, 131] are data-driven and require a large volume of clean training data. These methods classify behaviors as aggressive and conservative [119] or selfish and altruistic [75]. In contrast, deterministic models [14] do not require data and assign a real-valued score to each agent to quantify its behavior. These approaches can be integrated with risk-aware planning frameworks to incorporate planning for human agents.

Main Contributions:

We propose a novel approach for risk-aware planning in multi-agent traffic scenarios that takes into account human driver behaviors. We extend an existing risk-aware planner [13] by incorporating interactions with human drivers using a data-driven human driver behavior model [14]. We derive a linear mapping between the driver behavior and risk tolerance, which serves as the key component of our proposed approach.

To evaluate our approach, we validate the mapping between driver behavior and risk tolerance by measuring the number of lane changes, and test the accuracy of this model via K-Means clustering. Our results show that aggressive human driving results in more frequent lane changing. We confirm that the final trajectories obtained from the risk-aware planner generate emergent

behaviors. We measure the yield % and minimum distance between human drivers at intersections, at roundabouts, and during merging where we observe that conservative drivers generally yield to aggressive drivers while maintaining a greater distance from them. We also conduct a user study in which we show that users are able to distinguish between aggressive and conservative trajectories generated by the planner.

Finally, we compare our modified risk-aware planner with existing planners that do not model human drivers and show that modeling human drivers results in safer navigation. Specifically, [13] (and similar planners) assign a fixed neutral risk tolerance to human drivers and the ego-vehicle generates to the human driver accordingly. However, when the human driver is, in fact, either aggressive or conservative, then we show that the error (absolute value of minimum relative distance between the agents) increases by 10%.

4.4.1 Related Work

4.4.1.1 Risk-Aware Planning

Risk sensitivity-based planning [199, 200, 201, 202, 203] considers the risk associated with the actions of agents to avoid unsafe situations. The most common risk measures utilized in risk-sensitive planning are entropic risk [195] and conditional value at risk (CVaR) [196]. Entropic risk has been widely used in optimal control due to its simplicity and tractability [204], while CVaR has recently been incorporated in trajectory optimization due to its interpretability [201]. Risk-aware planning has been used extensively in autonomous underwater vehicles [205], ground vehicles [206, 207], and unmanned aerial vehicles (UAVs) [208]. While the CVaR risk model has been used in the latter two cases, [205] used the entropic measure of risk. In addition to CVaR

and the entropic models, several other models are also used in various applications such as the dynamic risk density function for collision avoidance [209] and semantic maps for simultaneous localization and mapping (SLAM) [210, 211]

4.4.1.2 Data-Driven Methods for Driver Behavior Prediction

Data-driven methods broadly follow two approaches. In the first approach, various machine learning algorithms such as clustering, regression, and classification predict or classify driver behavior as either aggressive or conservative. These methods have been studied in traffic psychology and the social sciences [76, 78, 81, 86, 87, 87, 88, 92, 93, 94, 212, 213, 214, 215, 216, 217, 218, 219]. So far, there has been relatively little work to improve the robustness and ability to generalize to different traffic scenarios, steps that require ideas from computer vision and robotics.

The second approach uses trajectories to learn reward functions for human behavior using inverse reinforcement learning (IRL) [75, 119, 131]. 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 [119, 131]. For instance, [119] requires 32 million data samples for optimum performance. Additionally, IRL-based methods are sensitive to noise in the trajectory data [75, 131]. Consequently, current IRL-based methods are restricted to simple and sparse traffic conditions.

4.4.2 Algorithm

We consider N agents consisting of a mixture of human drivers and AVs with the system dynamics defined by Equation 4.11, and we define the cost function for each agent by Equation 4.13. A human driver is simulated using a user-controlled keyboard with the following features: acceleration, braking, and lane changing. For simplicity, we test with one human driver, but our approach can work with more than one human driver. We further assume that agents are non-ideal in that agents are not provided the risk tolerance of other agents. The input to our approach consists of the state and control signals of every agent at time t. Then, our goal is to compute the Nash equilibrium trajectories for all agents. The trajectories for the human agents are predictions, while the trajectories for the AVs can be executed in a receding horizon planning loop. Finally, none of the agents are assumed to follow constant velocity models.

We describe our algorithm (Figure 3.2). The first step is to read the trajectories for every agent over a finite horizon T, denoted by Ξ_T. These trajectories correspond to human agents. The second step is to compute the CMetric value, ζ, for each agent during T; the CMetric value encodes the aggressive (or conservative) nature of the driver via certain indicators such as speeding, overtaking, and zigzagging. The third step consists of mapping an agent's CMetric to their risk sensitivity. This is performed using a linear transformation obtained by simple linear regression. This is discussed in detail in Section 4.4.2.1. Finally, based on the risk sensitivity, we perform game-theoretic risk-aware planning using the planner developed by Wang et al. [13].

4.4.2.1 CMetric to Risk Sensitivity

We denote the risk sensitivity parameter by θ. We first compute a linear mapping $\mathcal{M} : Z \longrightarrow \Theta$. Since both $\zeta \in Z$ and $\theta \in \Theta$ are scalars, we can use simple one dimensional linear regression to estimate $\mathcal{M}$. We create a training dataset by first generating trajectories corresponding to a fixed array of risk sensitivity values ranging from -5.0 (risk-seeking) to $+5.0$ (risk-averse). We denote these risk sensitivity values as $\hat{\theta}$ to indicate they are training values. We then evaluate the CMetric values, also represented using $\hat{\zeta}$, corresponding to each of these trajectories using the algorithm described in the previous section. The risk sensitivity and CMetric pair constitute the training dataset on which we apply linear regression to estimate linear coefficients β_0 and β_1. $\mathcal{M}$ is then defined as follows,

$$\mathcal{M}(\zeta) = \beta_1 \zeta + \beta_0, \tag{4.10}$$

where ζ is the CMetric value of a human agent at test time.

4.4.2.2 Risk-Aware Planning

The system dynamics for each agent are given by,

$$x_{t+1} = A_t x_t + B_t^1 u_t^1 + B_t^2 u_t^2 + w_t. \tag{4.11}$$

To simplify notation, we describe a two-player system, although our approach can easily generalize to n agents. $x_t = [x_t^1, x_t^2] \in X$ represents the system state and $x_t^i = [p_x^i, p_y^i, v_x^i, v_y^i]$ denotes the position (in meters) and velocity (in meters/second) of an agent. $u_t^1 = a_1, u_t^2 = a_2 \in U$ are the

control inputs for both agents denoting the acceleration of both agents, $w_t \sim \mathcal{N}(0, W_t)$ is the system noise, and A_t, B_t^1, B_t^2 are fixed matrices of appropriate dimensions. An agent incurs the following cost during a finite horizon T:

$$\Psi^i = \sum_{t=0}^{T-1} \left[\frac{1}{2} x_t^T Q_t^i x_t + l_t^{iT} x_t + \frac{1}{2} \sum_j u_t^{jT} R_t^{ij} u_t^j \right] + \frac{1}{2} x_T^T Q_T^i x_T + l_T^{iT} x_T, \tag{4.12}$$

where $Q_t \succeq 0$ and $R_t \succ 0$. To model risk, we use the exponential risk cost function used in [13],

$$J(\Psi) = \frac{1}{\mathcal{M}(\zeta)} \log \mathbb{E}\left[e^{(\mathcal{M}(\zeta)\Psi)}\right] = R_{\mathcal{M}(\zeta)}(\Psi), \tag{4.13}$$

where $\mathcal{M}(\zeta)$ is the risk tolerance of a human driver.

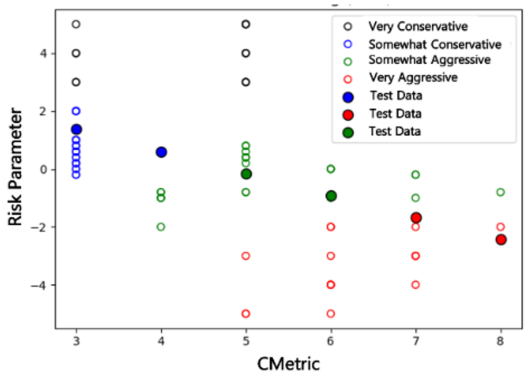
Remark 1*: The difference between Equation 4.13 and the risk cost function described in [13] is that the risk parameter in the latter work includes a fixed value for every agent, whereas in this work, we automatically generate the risk parameter for human agents in a data-driven fashion.*

The optimal strategies for each player can be obtained by minimizing $J(\Psi^i)$ for each agent i and obtaining the Nash equilibrium using Riccati recursion [220, Chap. 6]. However, Equation 4.13 is constrained by the fact that $\mathcal{M}(\zeta)$ is bounded. If the $\mathcal{M}(\zeta)$ is too low or too high, then the cost function value approaches ∞, also known as "neurotic breakdown" [13]. Due to the data-driven nature of Equation 4.13, in order to ensure optimality, certain traffic parameters such as traffic density is assumed, since they affect the CMetric value [14], and by Equation 4.10, the risk sensitivity of the human agent used in Equation 4.13.

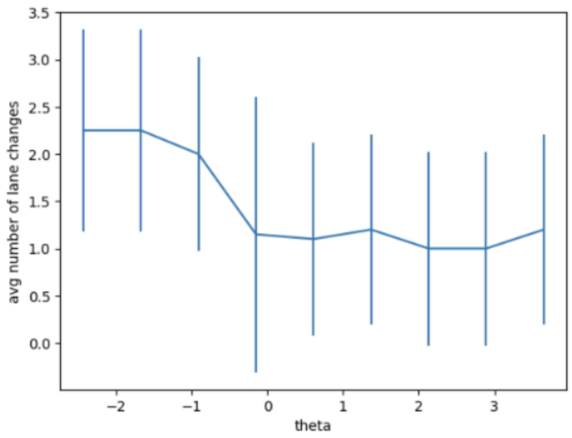
4.4.3 Experiments and Results

In this section, we present the results of extensive experiments testing the accuracy of the linear mapping between human driver aggressiveness and risk tolerance. We then evaluate the emergent behaviors associated with the final trajectories generated by the iterative risk sensitive game theoretic solver, compare with [13], which is chosen as the baseline (in which human driver behavior is ignored), and finally, discuss using alternative human driver behavior models. All experiments are performed using a 12-core 2.60GHz Intel i7 processor. We conduct open-loop tests that follow the pipeline outlined in Figure 3.2. We use the OpenAI traffic simulator [149] to compute the CMetric values representing human driver behavior and the python-based controller provided by Wang et al. [13] to generate the final trajectories based on the risk tolerances obtained from the corresponding CMetric values. The configurations of both the simulator and the controller (which include the dynamics of the vehicles, traffic density, number of lanes etc.) are kept identical so that all vehicles generated using the controller are tracked in the simulator.

We compute the CMetric values of the human driver in a highway scenario since we require a fixed duration of time (5s) during which we must observe the vehicle's trajectory and its interaction with other vehicles. For the risk-aware trajectory controller, we consider a merging scenario where a human agent must merge onto a highway with another human agent in the target merging lane. Here, the human agent is equivalent to an agent whose risk sensitivity value is obtained from the CMetric value. We assume vehicles follow the center line in their current driving lanes and only consider the vehicle's speed to finish the merging maneuver. In other words, we assume a steering controller will be executed separately for each car to remain in its lane.



(a) **KMeans clustering:** As human driver behavior becomes more aggressive (higher CMetric value), the risk parameter θ tends to decrease. This trend is consistent with the definition of risk sensitivity in [45]. Furthermore, our approach can effectively categorize the risk sensitivity of new human drivers (solid colored points) regardless of traffic density, number of lanes, etc.

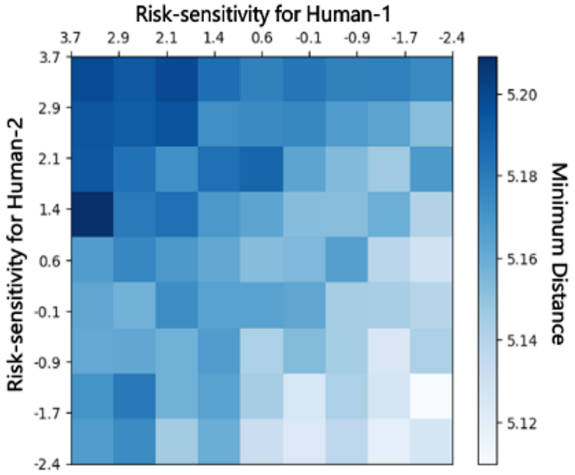


(b) **Number of lane changes:** Aggressive drivers ($\theta < 0$) yield a greater number of lane changes than conservative drivers ($\theta > 0$).

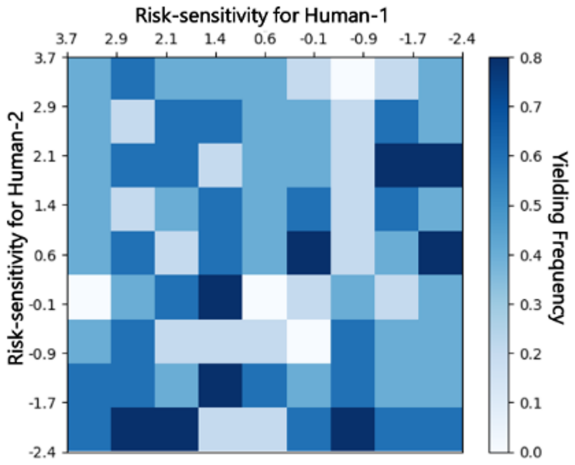
Figure 4.1: We highlight the relationship between the CMetric value and the risk parameter θ. Refer to Section 4.4.3.1 for further details.

4.4.3.1 Verifying the accuracy of $\mathcal{M}$

In Figure 4.1, we plot the risk parameter θ (y-axis) obtained from a given CMetric value (x-axis) via the linear mapping. When computing the risk parameters corresponding to each CMetric value, we vary the simulation configuration (traffic density, number of lanes etc.) to include a range of environments. This results in θ belonging to a range (as opposed to a fixed value). This is desirable since, in practice, the traffic will vary according to place and time. The risk parameters are clustered into four categories: "very conservative", "conservative", "aggressive", and "very aggressive". Each cluster is identified by a color. The empty circles are training data. The goal of this experiment is to cluster a test set of CMetric values (solid-colored points) based on their risk sensitivity. The test data are generated by a human driving the OpenAI simulator [149] in a randomly selected environment consisting of eleven vehicles and four lanes. The results (Figure 4.1) demonstrate that given the CMetric value, the linear regression mapping can accurately identify the risk sensitivity among a wide range of traffic configurations.



(a) **Minimum distance:** Darker colors (indicating larger minimum distance) corresponding to two risk-averse human agents and lighter colors (indicating smaller minimum distance) corresponding to two risk-seeking human agents.



(b) **Yielding behaviors:** Darker colors (indicating higher likelihood of yielding) corresponding to interactions between a risk-seeking agent and risk-averse agent. As the risk tolerances for both the human drivers are data-driven, and therefore noisy, both agents are adapting to the other. As a result, when either agent is risk-averse, we see higher yielding likelihood (darker colors).

Figure 4.2: **Yielding behaviors:** Darker colors (indicating higher likelihood of yielding) corresponding to interactions between a risk-seeking agent and risk-averse agent. As the risk tolerances for both the human drivers are data-driven, and therefore noisy, both agents are adapting to the other. As a result, when either agent is risk-averse, we see higher yielding likelihood (darker colors).

Another metric we use to validate the correlation between the CMetric and corresponding risk sensitivity is the average number of lane changes. Based on the final trajectories generated from the risk sensitivity parameter (obtained from corresponding CMetric values) and using the controller provided by [13], we measure the average number of lane changes made by the ego-vehicle. The reason for using average number of lane changes as a metric is that aggressive drivers change lanes more frequently than non-aggressive and conservative drivers. The aim of the experiment, therefore, is to check if an aggressive human-driven vehicle (modeled using the keyboard of the OpenAI simulator) results in more lane changes by the final simulated ego-vehicle (simulated using the python controller) and conversely, if a conservative human driver results in fewer simulated lane changes. In Figure 4.1, we confirm this is indeed the case; aggressive drivers ($\theta < 0$) yield a greater number of lane changes than conservative drivers ($\theta > 0$).

4.4.3.2 Emergent behaviors

We evaluate the final trajectories generated using the learned risk sensitivity of human drivers in a merging scenario where a human agent attempts to merge onto the highway. Different risk sensitivities yield a range of emergent behaviors. For example, in [13], Wang et al. showed that two risk-averse agents maintain a larger minimum distance between them, while risk-seeking agents may allow a smaller gap. Further, in an interaction between a risk-averse and a risk-seeking agent, there is a higher likelihood of the risk-averse agent yielding to the risk-seeking agent.

The experiments conducted by Wang et al. modeled synthetic agents for which the risk

sensitivity must be manually chosen. Here, we run the same set of experiments for human agents. In Figure 4.2, we can observe darker colors (indicating larger minimum distance) corresponding to two risk-averse human agents and lighter colors (indicating smaller minimum distance) corresponding to two risk-seeking human agents. In Figure 4.2, we can observe darker colors (indicating a higher likelihood of yielding for the risk-averse agent) corresponding to interactions between a risk-seeking agent and a risk-averse agent.

4.4.3.3 User studies

We recruited 27 participants to respond to a user study consisting of two questions. The first question involved showing two video clips of final trajectories. The first clip (top) consisted of a risk seeking trajectory ($\theta = -2.429$), while the second (bottom) consisted of a risk averse trajectory ($\theta = 3.651$). Participants were not told the risk preferences that generated the trajectories, and were asked to identify which trajectory corresponded to an aggressive driver. The goal of this question is to qualitatively assess the emergent nature of the final trajectories. That is, based on simply observing the nature of the trajectory, can a human distinguish between the generated trajectories? We answer in the affirmative; 26 out of the 27 participants correctly identified the risk seeking driver as the aggressive driver.

4.4.3.4 Comparing with the baseline

We compare our modified risk-aware planner with existing planners that do not model human drivers and show that modeling human drivers results in safer navigation. Specifically, [13] (and similar planners) assign a fixed neutral risk tolerance to human drivers and the ego-vehicle

generates to the human driver accordingly. There are two outcomes:

1. Suppose the human driver is, in fact, aggressive. Then, by modeling the driver with a neutral risk tolerance, the ego-vehicle may stray close to the aggressive driver as opposed to keeping a safe distance from them.

2. Conversely, suppose the human driver is conservative. Then, by modeling the driver with a neutral risk tolerance, the ego-vehicle may enter a brief deadlock during which both agents wait to see who moves first.

We aim to capture these inefficiencies via a single error metric, which is the absolute value of the minimum relative distance between the two agents. This metric is ideal since in both cases, it measures the discrepancy between the expected distance and the actual observed distance. For example, we show that in the first case, the expected minimum relative distance between both agents is more than the observed distance while in the second case, we show the observed minimum distance is more than the actual distance. In both cases, the error is positive by virtue of the absolute value. Empirically, the maximum RMSE observed is 0.0425m or 10% as shown in Figure 4.3.

4.4.3.5 Using alternative human driver behavior models

Thus far, we have successfully demonstrated that CMetric can be effectively integrated with risk-aware planning to generate game-theoretic behavior-rich trajectories. Alternative models for human driver behavior such as the SVO can theoretically be used. However, there are practical issues when it comes to integrating these alternative models in risk-aware planning. Here, we discuss some of these challenges. SVO is an offline technique that requires a large volume

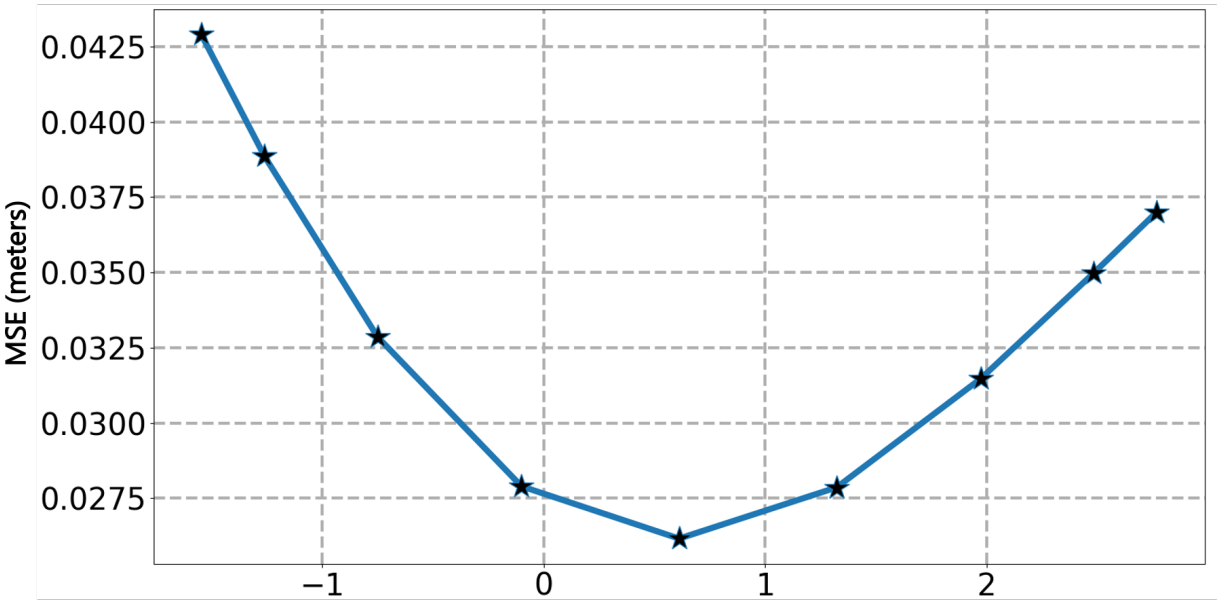


Figure 4.3: **Comparison with [13]:** The approach by Wang et al. assumes a neutral risk sensitivity for human agents. However, when an ego-agent interacts with a human driver who may be aggressive or conservative (indicated by the negative and positive values on the x axis, respectively), then the assumption of neutral risk tolerance results in an error in terms of absolute value of minimum relative distance between the two agents.

of training data to learn a data-driven reward function via inverse reinforcement learning. Our technique is meant to be deployed in realtime and, as such, we test in an open-loop simulation and use *active* metrics such as yield %, frequency of lane changes, and minimum distance between agents. The SVO approach, on the other hand, uses RMSE to measure the deviation of the prediction trajectories from the ground-truth trajectories. We do not assume the availability of ground-truth data. In future, we will conduct experiments comparing CMetric with SVO once the source code for SVO is public.

4.4.4 Conclusion, Limitations, and Future Work

We presented an approach for risk-aware planning in multi-agent traffic with human agents. The basic intuition of our approach is that aggressiveness of a driver is linearly correlated with

risk preference. That is, aggressive drivers are more risk-seeking while conservative drivers are more risk-averse. Accordingly, we integrate a human driver behavior model [14] with the risk-aware dynamic game solver in [13] via simple linear regression to derive a mapping between driver behavior and risk tolerance. Our results show that aggressive human driving results in more frequent lane-changing. We show that conservative drivers generally yield to aggressive drivers while maintaining a greater distance from them. Finally, we confirm that the final trajectories obtained from the risk-aware planner generate emergent behaviors though a comprehensive user study in which participants were able to distinguish between aggressive and conservative drivers.

There are some limitations to our method. Currently, we have tested our approach in an open-loop simulation where we use two different simulators for the human behavior model and the trajectory planner. To use both simulators in open-loop simulation effectively, the environment configuration must be kept identical, which is cumbersome and a hindrance. In the future, we will explore a closed-loop simulator that combines the human behavior model and the risk-aware trajectory planner.

Chapter 5: Software and Datasets

We present a new traffic dataset, METEOR, which captures traffic patterns and multi-agent driving behaviors in unstructured scenarios. METEOR consists of more than 1000 one-minute videos, over 2 million annotated frames with bounding boxes and GPS trajectories for 16 unique agent categories, and more than 13 million bounding boxes for traffic agents. METEOR is a dataset for rare and interesting, multi-agent driving behaviors that are grouped into traffic violations, atypical interactions, and diverse scenarios. Every video in METEOR is tagged using a diverse range of factors corresponding to weather, time of the day, road conditions, and traffic density. We use METEOR to benchmark perception methods for object detection and multi-agent behavior prediction. Our key finding is that state-of-the-art models for object detection and behavior prediction, which otherwise succeed on existing datasets such as Waymo, fail on the METEOR dataset. METEOR marks the first step towards the development of more sophisticated perception models for dense, heterogeneous, and unstructured scenarios.

5.1 Overview

Recent research in learning-based techniques for robotics, computer vision, and autonomous driving has been driven by the availability of datasets and benchmarks. Several traffic datasets have been collected from different parts of the world to stimulate research in autonomous driving,

driver assistants, and intelligent traffic systems. These datasets correspond to highway or urban traffic, and are widely used in the development and evaluation of new methods for perception [45], prediction [66], behavior analysis [120], and navigation [221].

Many initial autonomous driving datasets were motivated by computer vision or perception tasks such as object recognition, semantic segmentation or 3D scene understanding. Recently, many other datasets have been released that consist of point-cloud representations of objects captured using LiDAR, pose information, 3D track information, stereo imagery or detailed map information for applications related to 3D object recognition and motion forecasting. Many large-scale motion forecasting datasets such as Argoverse [222], and Waymo Open Motion Dataset [223], among others, have been used extensively by researchers and engineers to develop robust prediction models that can forecast vehicle trajectories. However, existing datasets do not capture the rare behaviors or heterogeneous patterns. Therefore, prediction models trained on these existing datasets are not very robust in terms of handling challenging traffic scenarios that arise in the real world.

A major challenge currently faced by research in autonomous driving is the *heavy tail problem* [222, 223], which refers to the challenge of dealing with rare and interesting instances. There are several ways in which existing datasets currently address the heavy tail problem:

1. **Mining:** The Argoverse and Waymo datasets use a mining procedure that includes scoring each trajectory based on its "interestingness" to explicitly search for difficult and unusual scenarios [222, 223].

2. **Diversifying the taxonomy:** Train the prediction and forecasting models to identify the unknown agents at the time of testing. This approach necessitates annotating a diverse

taxonomy of class labels. Argoverse and nuScenes [224] contain 15 and 23 classes, respectively.

3. **Increasing dataset size:** This approach is to simply collect more data with the premise that collecting more traffic data will likely also increase the number of such scenarios in the dataset.

In spite of many efforts along these lines, existing datasets manage to collect only a handful of such instances, due to the infrequent nature of their occurrence. For example, the Waymo Open Motion dataset [223] contains only atypical interactions and diverse scenarios while the Argoverse dataset [222] contains only atypical interactions. There is clearly a need for a different approach to addressing the heavy tail problem. Our solution is to build a traffic dataset from videos collected in India, where the inherent nature of the traffic is dense, heterogeneous, and unstructured. The traffic patterns and surrounding environment in parts of India are more challenging. than those in other parts of the world. This includes high congestion and traffic density. Some of these roads are unmarked or unpaved. Moreover, the traffic agents moving on these roads correspond to vehicles, buses, trucks, bicycles, pedestrians, auto-rickshaws, two-wheelers such as scooters and motorcycles, etc.

5.1.1 Main Contributions

1. We present a novel dataset, METEOR, corresponding to the dense, heterogeneous, and unstructured traffic in India. METEOR is the first large-scale dataset containing annotated scenes for rare and interesting instances and multi-agent driving behaviors, broadly grouped into:

   (a) Traffic violations—running traffic signals, driving in the wrong lanes, taking wrong

turns).

(b) Atypical interactions—cut-ins, yielding, overtaking, overspeeding, zigzagging, lane changing.

(c) Diverse scenarios—intersections, roundabouts, and traffic signals.

2. METEOR has more than 2 million labeled frames and 13 million annotated bounding boxes for 16 unique traffic agents, and GPS trajectories for the ego-agent.

3. Every video in METEOR is tagged using a diverse range of factors including weather, time of the day, road conditions, and traffic density.

4. We evaluate state-of-the-art methods for object detection and multi-agent behavior prediction on METEOR.

5. We present a novel, fine-grained analysis on the relationship between traffic environments and perception. Specifically we study the effect of 2D object detection in varying traffic density, mixture of agents, area, time of the day, and weather conditions.

5.1.2 Applications and Benefits

- **Towards Risk-Aware Planning and Control:** Our multi-agent behavior prediction benchmark can aid the development of risk-aware motion planners by predicting the behaviors of surrounding agents. Motion planners can compute controls that guarantee safety around aggressive drivers who are prone to overtaking and overspeeding.

- **Towards Robust Perception:** We observe that these models fail in challenging Indian traffic scenarios, compared to their performance on existing datasets captured in the US,

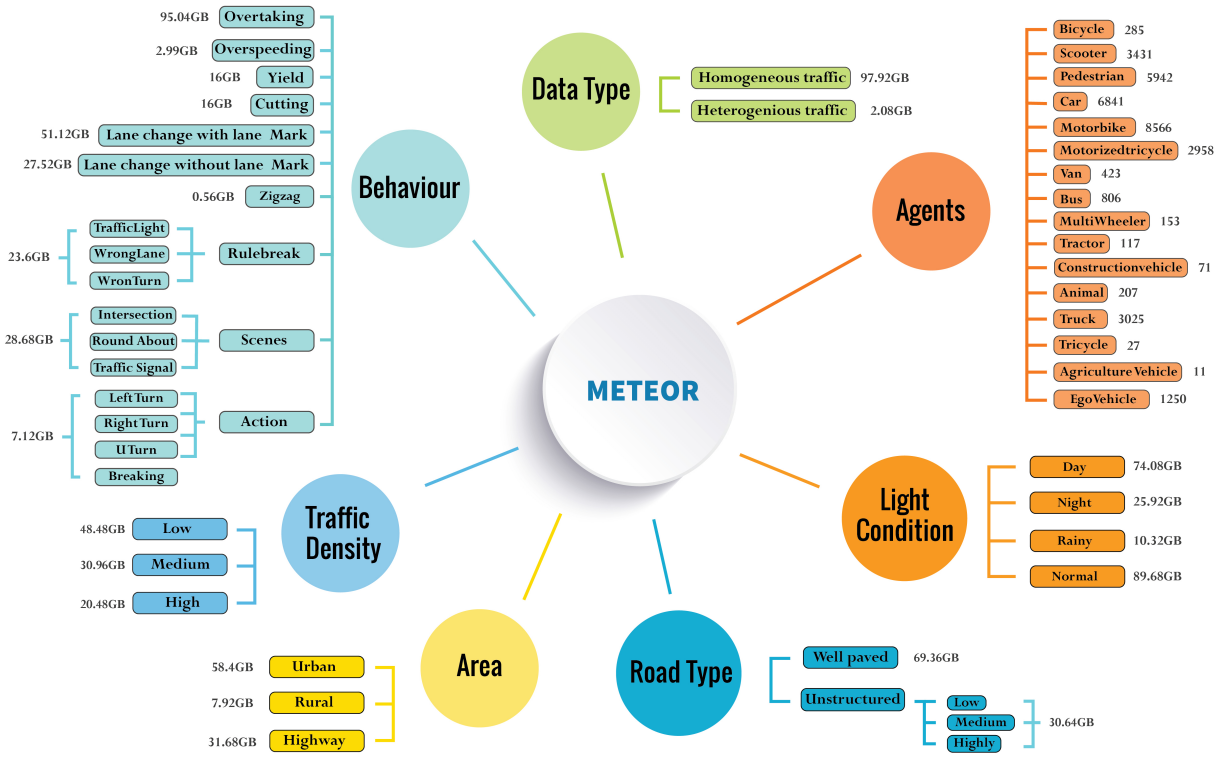


Figure 5.1: **METEOR:** We summarize various characteristics of our dataset in terms of scene: traffic density, road type, lighting conditions, agents (we indicate the total count of each agent across 1250 videos), and behaviors, along with their size distribution (in GB). The total size of the current version of the dataset is around 100GB, and it will continue to expand. Our dataset can be used to evaluate the performance of current and new methods for perception, prediction, behavior analysis, and navigation based on some or all of these characteristics. Details of the organization of our dataset are given at **`https://gamma.umd.edu/meteor`**.

Europe, and other developed nations. As a result, METEOR can be a useful benchmark for research in perception in unstructured traffic environments and developing nations.

- **Towards Fine-grained Traffic Analysis:** Our novel analysis studying the relationship between traffic patterns and 2D object detection can lead to more informed research in perception for autonomous driving.

5.2 Comparison with Existing Datasets

5.2.1 Tracking and Trajectory Prediction Datasets

Datasets such as the Argoverse [222], Lyft Level 5 [225], Waymo Open Dataset [223], ApolloScape [72], nuScenes dataset [224] are used for trajectory forecasting [65, 102, 104, 226, 227] and tracking [45]. Several of these datasets use mining procedure [222, 223] that heuristically searches the dataset for rare and interesting scenarios. The resulting collection of such scenarios and behaviors, however, is only a fraction of the entire dataset. METEOR, by comparison, exclusively contains such scenarios due to the inherent nature of the unstructured traffic in India.

METEOR has many additional characteristics with respect to these datasets. For instance, METEOR's 2.02 million annotated frames are more than $10\times$ the current highest number of annotated frames with respect to other dataset with high density traffic (ApolloScape). Furthermore, METEOR consists of 16 different traffic-agents that include only on-road moving entities (and not static obstacles). This is by far, the most diverse in terms of class labels. In comparison, Argoverse and nuScenes both contain 10 and 13 traffic-agents, respectively. METEOR is the first motion forecasting and behavior prediction dataset with traffic patterns from rural and urban areas that consist of unmarked roads and high-density traffic. In contrast, traffic scenarios in Argoverse, Waymo, Lyft, and nuScenes have been captured on sparse to medium density traffic with well-marked structured roads in urban areas.

Table 5.1: **Characteristics of Traffic Datasets:** We compare METEOR with state-of-the-art autonomous driving datasets that have been used for trajectory tracking, motion forecasting, semantic segmentation, prediction, and behavior classification. METEOR is the largest (in terms of number of annotated frames) and most diverse in terms of heterogeneity, scenarios, varying behaviors, densities, and rare instances. Darker shades represent a richer collection in that category. Best viewed in color.

										Rare and Interesting Behaviors[‡]		
Datasets	Location	Bad weather	Night	Road type	Het.[⋆]	Size	Density	Lidar	HD Maps	Traffic Violations	Atypical Interactions	Diverse Scenarios
Argoverse [222]	USA	✓	✓	urban	10	22K	Medium	✓	✓	✗	✓	✗
Lyft Level 5 [225]	USA	✗	✗	urban	9	46K	Low	✓	✓	✗	✗	✗
Waymo [223]	USA	✓		urban	4	200K	Medium	✓	✓	✗	✓	✓
ApolloScape [72]	China	✗	✓	urban, rural	5	144K	High	✓	✓	✗	✗	✗
nuScenes [224]	USA/Sg.	✓	✓	urban	13	40K	Low	✓	✓	✗	✓	✓
INTERACTION [228]	International	✗	✗	urban	1	–	Medium	✓	✓	✗	✗	✗
CityScapes [229]	Europe	✗	✗	urban	10	25K	Low	✗	✗	✗	✗	✗
IDD [230]	India	✗	✗	urban, rural	12	10K	High	✗	✗	✗	✗	✗
HDD [231]	USA	✗	✗	urban	–	275K	Medium	✓	✗	✗	✓	✓
Brain4cars [232]	USA	✗	✗	urban	–	2000K	Low	✗	✓	✗	✗	✗
D2-City [233]	China	✓	✗	urban	12	700K	Medium	✗	✗	✗	✗	✓
TRAF [65]	India	✗	✓	urban, rural	8	72K	High	✗	✗	✗	✗	✗
BDD [234]	USA	✓	✓	urban	8	**3000K**	Low	✗	✗	✗	✗	✓
METEOR	**India**	✓	✓	**urban, rural**[†]	**16**[††]	2027K	**High**[§]	✗	✗	✓	✓	✓

[‡] Rare instances can be broadly grouped into (i) traffic violations, (ii) atypical interactions, and (iii) difficult scenarios.
[†] Includes roads without lane markings. Roads in other datasets with rural roads may contain lane markings.
[⋆] Heterogeneity. We indicate the classes corresponding to moving traffic agents only, excluding static objects such as poles, traffic lights, etc.
[§] Up to 40 agents per frame.
[††] Up to 9 unique agents per frame.

5.2.2 Semantic Segmentation Datasets

CityScapes [229] is widely used for several tasks, primarily semantic segmentation. It is based on urban traffic data collected from European cities with structured roads and low traffic density. In contrast, the Indian Driving Dataset (IDD) [230] is collected in India with both urban and rural areas with high-density traffic. A common aspect of both these datasets (CityScapes and IDD), however, is the relatively low annotated frame count (25K and 10K, respectively). This is probably due to the effort involved with annotating every pixel in each image. IDD also contains high-density traffic scenarios in rural areas, similar to METEOR. However, our dataset has $200\times$ the number of annotated frames and $1.6\times$ the number of traffic-agent classes. Similar to TRAF, the IDD does not contain the behavior data that is provided by METEOR.

5.2.3 Behavior Prediction

Behavior prediction corresponds to the task of predicting turns (right, U-turn, or left), acceleration, merging, and braking in addition to driver-intrinsic behaviors such as over-speeding, overtaking, cut-ins, yielding, and rule-breaking. The two most prominent datasets for action prediction include the Honda Driving Dataset (HDD) [231] and the BDD dataset [234]. Some of the major distinctions between METEOR and the HDD in terms of size (approximately $10\times$), the availability of scenes with night driving and rainy weather, and the inclusion of unstructured environments in low-density traffic. The BDD dataset [234] contains more annotated samples than METEOR, however, the BDD dataset contains 100K videos while METEOR contains 1K videos. So the number of annotated samples per video is $66\times$ higher for METEOR. The annotations in prior datasets are limited to actions and do not contain the rare and interesting behaviors contained in METEOR.

5.3 METEOR dataset

Our dataset is visually shown in Figure 5.1. Below, we present some details of the data collection process and discuss some of the salient features and characteristics of METEOR.

5.3.1 Dataset Collection

The data was collected in and around the city of Hyderabad, India within a radius of 42 to 62 miles. Several outskirts were chosen to cover rural and unstructured roads. Our hardware capture setup consists of two wide-angle Thinkware F800 dashcams mounted on an MG Hector and Maruti Ciaz. The camera sensor has 2.3 megapixel resolution with a 140 degrees field of

view. The video is captured in full high definition with a resolution of 1920×1080 pixels at a frame rate of 30 frames per second. The dashcam is embedded with an accurate positioning system that stores the GPS coordinates, which were processed into the world frame coordinates. The sensor synchronizes between the camera and the GPS. Recordings from the dashcam are streamed continuously and are clipped into 1 minute video segments.

5.3.2 Dataset organization

The dataset is organized as 1250 one-minute video clips. Each clip contains static and dynamic XML files. Each static file summarizes the meta-data of the entire video clip including the behaviors, road type, scene structure etc. Each dynamic file describes frame-level information such as bounding boxes, GPS coordinates, and agent behaviors. Our dataset can be searched using helpful filters that sort the data according to the road type, traffic density, area, weather, and behaviors. We also provide many scripts to easily load the data after downloading.

5.3.3 Annotations

We provide the following annotations in our dataset: (i) bounding boxes for every agent, (ii) agent class IDs, (iii) GPS trajectories for the ego-vehicle, (iv) environment conditions including weather, time of the day, traffic density, and heterogeneity, (v) road conditions with urban, rural, lane markings, (vi) road network including intersections, roundabouts, traffic signal, (vii) actions corresponding to left/right turns, U-turns, accelerate, brake, $(viii)$ rare and interesting behaviors (See Section 5.3.4), and (ix) the camera intrinsic matrix for depth estimation to generate trajectories of the surrounding vehicles. This set of annotations is the most diverse and extensive

compared prior datasets.

A diverse and rich taxonomy of agent categories is necessary to ensure that autonomous driving systems can detect different types of agents in any given scenario. Towards that goal, datasets for autonomous driving are designed or captured to achieve two goals: (a) capture as many different types of agent categories as possible; (b) capture as many instances of each category as possible. In both these aspects, METEOR outperforms all prior datasets. We annotate 16 types of moving traffic entities, not including static obstacles listed in Figure 5.1 along with their distribution. Note specifically that the percentages of pedestrians, motorbikes, and bicycles are higher than the percentage of passenger vehicles. This is particularly useful as the former categories are known as "vulnerable road users" (VRUs) [235], and it is important for autonomous driving systems to be able to detect them–necessitating many instances of these VRUs in any dataset.

5.3.4 Rare and Interesting Behaviors

We provide a total of 17 different types of rich collection of rare and interesting cases that are unique to our dataset. They can be summarized in terms of the following groups:

5.3.4.1 Atypical Interactions

Atypical interactions correspond to pairwise interactions among traffic agents that are not often observed in regular traffic scenarios. Some examples of atypical interactions include yielding to, and cutting across, pedestrians, zigzagging through traffic, pedestrian jaywalking, overtaking, sudden lane changing, and overspeeding. We describe these in more detail below:

Figure 5.2: **Annotations for rare instances:** One of the unique aspects of METEOR is the availability of explicit labels for rare and interesting instances including atypical interactions, traffic violations, and diverse scenarios. These annotations can be used to benchmark new methods for object detection and multi-agent behavior prediction.

- *Overtaking (OT)*: When an agent overtakes another agent with sudden or aggressive movement.
- *Overspeeding (OS)*: If the vehicle over-speeds (based on speed limits) due to any reason.
- *Yield (Y)*: A pedestrian, bicycle, or any slow-moving agent trying to cross the road in front of another agent. If the latter slows down or stops, letting them cross the road then such behavior is labeled as yield.
- *Cutting (C)*: When pedestrians, bicycles, or any slow-moving agents trying to cross the road is interrupted by another agent. Yielding and cutting can also be re-labeled as instances of jaywalking. In a majority of these cases, one of the agents involved is a pedestrian crossing

the road in the middle of traffic.

- *Lane change w. lane markings (LC(m))*: Agents aggressively change lanes on roads with clear lane markings.

- *Lane change w/o. lane markings (LC)*: Agents aggressively change lanes on roads without lane markings.

  The above two annotations can be used to identify videos in the dataset that contain roads without lane markings for relevant applications.

- *Zigzagging (ZM)*: If any of the agent of interest undergoes a zigzag movement in the traffic, the agent behavior is classified as zigzagging.

5.3.4.2 Traffic Violations

In addition to the above driving behaviors, we also annotate traffic agents breaking traffic rules. These are particularly unique since rule breaking scenarios are rare.

- *Running a traffic light (RB TL)*: Passing through an intersection even though the traffic signal is red.

- *Wrong Lane (RB WL)*: A road may not be divided for inbound and outbound traffic by a physical barrier, making it possible for the motorists to use the inbound lane for the outbound traffic and vice versa. This behavior identifies all such cases.

- *Wrong Turn (RB WT)*: When an agent makes an illegal turn (including U-turns).

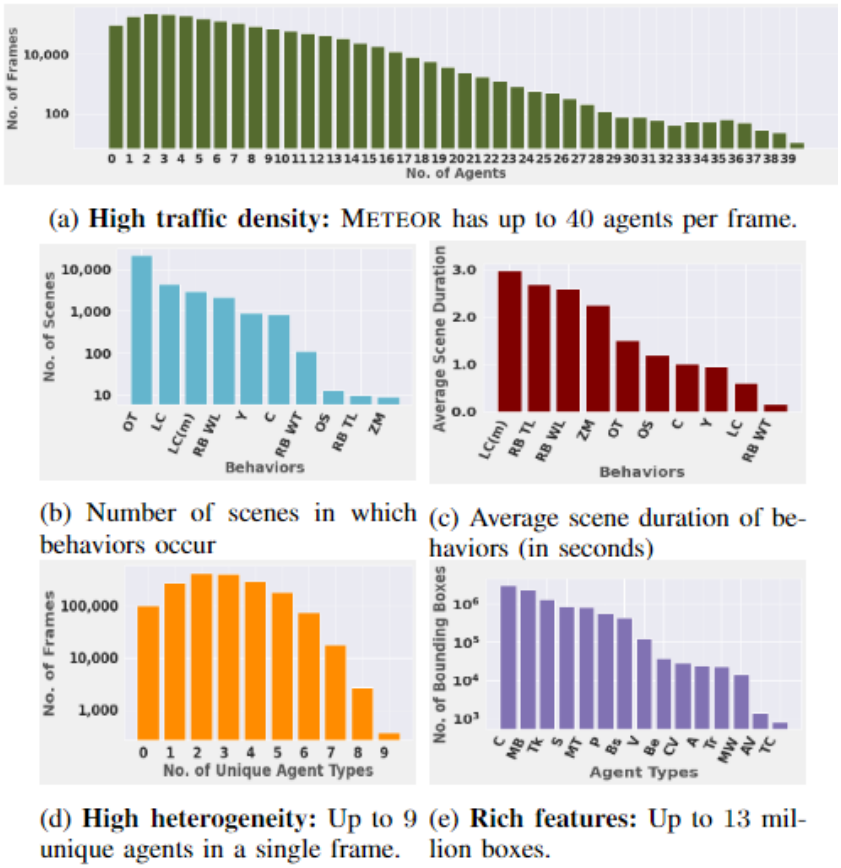


Figure 5.3: We highlight the high traffic density, heterogeneity, and the richness of behavior information in METEOR. Abbreviations correspond to various behavior categories and are explained in Section 5.3.4.

5.3.4.3 Diverse Scenarios

Finally, we provide annotations for challenging scenarios that include intersections, roundabouts, traffic signals, executing left turns, right turns, and U-turns.

5.3.5 Dataset statistics

We analyze the dataset statistics and distribution of agents and their behaviors in terms of total count, uniqueness, and duration (in seconds). Figure 5.3 show that METEOR is very dense and highly heterogeneous, respectively; the total number of agents in a single frame can

Table 5.2: **Effect of meta features on object detection:** We analyze how meta features such as traffic density, type of agents, location, time of the day, and weather play a role in 2D object detection using the DETR, Deformable DETR, YOLOv3 and CenterNet object detectors. **Bold** indicates the type of meta feature that is the most effective for object detection.

DETR and Deformable DETR (in parentheses)											
	Density			Agents		Environment		Time		Weather	
	Low	**Medium**	High	**Mixed**	Uniform	**Urban**	Rural	**Day**	Night	**Normal**	Rainy
mAP	19.00 (22.70)	27.00 (38.30)	19.30 (28.10)	27.00 (38.30)	14.80 (31.30)	27.00 (38.30)	14.20 (25.70)	27.00 (38.30)	12.00 (20.60)	27.00 (38.30)	12.00 (20.90)
mAP_{50}	33.33 (36.80)	48.40 (61.80)	32.40 (41.40)	48.40 (61.80)	31.80 (44.30)	48.40 (61.80)	23.40 (34.90)	48.40 (61.80)	22.70 (36.10)	48.40 (61.80)	21.90 (32.70)
mAP_{75}	21.50 (22.10)	28.10 (41.50)	20.40 (31.30)	28.10 (41.50)	11.70 (37.00)	21.80 (41.50)	16.30 (28.40)	28.10 (41.50)	12.20 (20.50)	28.10 (41.50)	12.60 (22.90)
mAP_S	2.60 (7.10)	1.20 (12.10)	0.20 (2.50)	1.20 (12.10)	0.30 (12.80)	1.20 (12.10)	2.00 (10.30)	1.20 (12.10)	0.10 (0.30)	1.20 (12.10)	1.80 (9.50)
mAP_M	7.40 (25.20)	8.30 (22.50)	10.50 (16.90)	8.30 (22.50)	7.20 (34.30)	8.30 (22.50)	11.70 (28.10)	8.30 (22.50)	3.30 (12.50)	8.30 (22.50)	6.20 (19.90)
mAP_L	25.60 (24.90)	45.90 (54.10)	24.70 (35.60)	45.90 (54.10)	40.30 (57.80)	45.90 (54.10)	26.30 (35.60)	45.90 (54.10)	16.70 (27.80)	45.90 (54.10)	15.10 (23.80)
YOLOv3 and CenterNet (in parentheses)											
	Density			Agents		Environment		Time		Weather	
	Low	**Medium**	High	**Mixed**	Uniform	**Urban**	Rural	**Day**	Night	**Normal**	Rainy
mAP	19.20 (22.90)	30.40 (32.90)	21.10 (23.30)	30.40 (32.90)	19.10 (30.20)	30.40 (32.90)	13.80 (13.60)	30.40 (32.90)	13.30 (15.90)	30.40 (32.90)	13.40 (14.00)
mAP_{50}	36.90 (34.80)	52.50 (55.40)	36.30 (32.50)	52.50 (55.40)	35.10 (43.40)	52.50 (55.40)	22.00 (22.70)	52.50 (55.40)	25.00 (25.70)	52.50 (55.40)	25.00 (22.50)
mAP_{75}	16.10 (28.10)	32.30 (33.40)	23.20 (26.70)	32.30 (33.40)	19.70 (37.30)	32.30 (33.40)	15.70 (13.20)	32.30 (33.40)	13.40 (27.00)	32.30 (33.40)	13.60 (15.50)
mAP_S	2.70 (8.40)	2.40 (13.10)	0.60 (2.90)	2.40 (13.10)	7.90 (19.30)	2.40 (13.10)	5.20 (5.40)	2.40 (13.10)	0.00 (0.90)	2.40 (13.10)	1.30 (10.90)
mAP_M	14.10 (26.20)	13.10 (30.50)	11.70 (17.60)	13.10 (30.50)	19.10 (38.80)	13.10 (30.50)	22.50 (25.80)	13.10 (30.50)	7.50 (11.60)	13.10 (30.50)	11.60 (17.40)
mAP_L	23.70 (29.50)	48.70 (44.60)	27.30 (27.90)	48.70 (44.60)	38.90 (40.00)	48.70 (44.60)	21.20 (21.40)	48.70 (44.60)	18.50 (21.70)	48.70 (44.60)	16.40 (14.30)

reach up to 40 and up to 9 unique agents can exist in a single frame. Figure 5.3 represents the distribution of behaviors across videos and Figure 5.3 shows the distribution of each behavior's average duration. In particular, we note that the average duration can reach up to 3 seconds which, at 30 frames per second, corresponds to approximately 90 frames that contain visual, contextual, and semantic information that can inform behavior prediction algorithms for more accurate perception and prediction.

5.4 Experiments and Analysis

We provide the pre-trained models for object detection and behavior prediction at **`https://gamma.umd.edu/meteor`**.

5.4.1 Analyzing Object Detection in Unstructured Scenarios

Existing datasets have helped develop sophisticated and robust 2D detection methods. We use the MMDetection [239] toolbox to train the following 2D object detection models—

Table 5.3: **Training Details for Object Detection** (BS: Batch size, Mom: Momentum, WD: Weight decay, MGN: Max Gradient Norm)

Method	Backbone	BS	Opt.	LR	Mom.	WD (L_2)	MGN
DETR [236]	ResNet-50	2	AdamW	1e−4	–	1e−4	0.1
Def. DETR [237]	ResNet-50	2	AdamW	2e−4	–	1e−4	0.1
YOLOv3 [7]	Darknet-53	8	SGD	1e−3	0.9	5e−4	35
CenterNet [238]	ResNet-18	16	SGD	1e−3	0.9	5e−4	35

Table 5.4: **Object detection on Waymo and KITTI:** We report the standard mAP for many widely used methods on autonomous driving datasets.

	DETR [236]	CenterNet	YOLO v3	Def. DETR	Swin-T
KITTI [74]	23.00	80.40	81.60	42.20	–
Waymo [223]	65.31	64.83	56.93	65.31	37.20
METEOR	**8.30**	**12.10**	**14.30**	**15.80**	**32.60**

DETR [236], Deformable DETR [237] (with iterative bounding box refinement), YOLOv3 [7] (with scale 608), CenterNet [238] (with normal convolutions), and Swin-T [240]. The models are pre-trained on the COCO dataset [241] and fine-tuned on METEOR. We provide the training details in Table 5.3 and report results using the standard mAP, mAP_{50}, mAP_{75}, mAP_{S}, mAP_{M}, and mAP_{L}. We refer the reader to [242] for a primer on these metrics.

In Table 5.4, we report the mAP for the 2D object detectors listed above. We observe that the most widely used 2D object detectors, that perform well on the state-of-the-art autonomous driving datasets, like the Waymo Open Motion Dataset [223] and the KITTI dataset [74], do not perform well on METEOR. More specifically, the detectors achieve $37\% - 65\%$ and $23\% - 81\%$ mAP on the Waymo and KITTI datasets, respectively, while the same methods achieve $8\% - 31\%$ mAP on the METEOR dataset. In other words, the best possible result on METEOR is $\frac{1}{2}\times$ and $\frac{1}{3}\times$ the best result on the Waymo and KITTI datasets, respectively. In Table 5.5, we compare METEOR in depth with the Waymo dataset using the Swin-T method [240], which

Table 5.5: **Swin-T on Waymo and METEOR:** We present a more detailed analysis of Swin-T, one of the state-of-the-art object detection approaches, on Waymo and METEOR.

	mAP	mAP_{50}	mAP_{75}	mAP_S	mAP_M	mAP_L
Waymo [223]	37.20	70.60	52.00	17.20	41.80	67.20
METEOR	**32.60**	**46.90**	**36.20**	**20.50**	**35.40**	**54.70**

is currently one of the top performing methods on the standard COCO 2D object detection benchmark leaderboard [241]. The Swin-T method performs 14% better on the Waymo Dataset.

There are two possible reasons for performance degradation on METEOR. First, 2D detectors are typically pre-trained on MS COCO [241] and ImageNet [243], which contain only up to 9 categories of the commonly occurring traffic agents. This was not an issue for detectors on existing datasets like Waymo and KITTI since those datasets contain a subset of those 9 classes. METEOR, on the other hand, contains 16 agent categories that are approximately equally distributed. The approximately $7-8$ traffic agent categories that are contained in METEOR but do not appear in MS COCO are novel to these 2D object detectors and are not classified correctly.

The other reason why object detection deteriorates on METEOR is due to the challenging traffic environments in METEOR. More specifically, METEOR contains many challenging scenarios such as bad weather, nighttime traffic, rural area, high density traffic, etc. (see Figure 5.2). We analyze the effect of meta-features such as traffic conditions (density and heterogeneity), road conditions, weather, and time-of the day on 2D object detection and present this analysis in Table 5.2. For this analysis, we form separate test sets corresponding to each label in a meta-feature (for example, we have two test sets for day and night). Most datasets contain videos of medium density traffic. In Table 5.2, we see that the performance of the DETR, Deformable DETR, YOLOv3, and CenterNet suffers as the traffic density increases from medium

to high. Similar reasoning can be made for other factors–object detection is less effective for homogeneous traffic, in rural areas, at nighttime, and in rainy weather. In most datasets, the number of annotated data samples with these adverse and challenging factors are a fraction of the entire dataset, which partly explains why 2D detectors are more successful on those datasets. The analysis in this section empirically validates the difficulty that the heavy-tail problem poses to perception tasks in autonomous driving.

5.4.2 Multi-Agent Behavior Recognition

Multi-agent behavior recognition (MABR) is the task of first localizing agents in a video followed by classifying their behaviors. This task has drawn attention in recent years and plays an important role in autonomous driving. Unlike object detection, which can be accomplished solely by observing visual appearances, MABR reasons about the actors' interactions with the surrounding context, including environments, other people and objects.

Dataset Preparation: The METEOR dataset is ideal for spatio-temporal MABR due to the availability of bounding box annotations and their corresponding behavior labels for more than 1231 video clips, each lasting one minute in duration, and over 2 million annotated frames. We use 1000 video clips for training and 231 video clips for testing. As the guidelines of the benchmarks, we evaluate 16 behavior classes with mean Average Precision (mAP) as the metric, using a frame-level IoU threshold of 0.5.

Framework: We use the ActorContext-Actor Relation Network (ACAR-Net) [245] which builds upon a novel high-order relation reasoning operator and an actor-context feature bank for indirect relation reasoning for spatio-temporal action localization. This framework is composed of an

Table 5.6: **ACAR-Net on AVA and METEOR:** We applied currently the state-of-the-art multi-agent action recognition approach on AVA to our METEOR dataset. (PT: pre-train, BS: batch size, Opt.: Optimization, LR: learning rate, WD: weight decay, FR(RX-101): Faster R-CNN (ResNeXt-101), Kin.-700: Kinetics-700, CR(Swin-T): Cascade R-CNN (Swin-T))

Dataset	Detector	PT	BS	Opt.	LR	WD	mAP
AVA [244]	FR(RX-101)	Kin.-700	32	SGD	0.008	1e−7	30.0
METEOR	CR(Swin-T)	Kin.-700	32	SGD	0.008	1e−7	**6.10**

object detector, backbone network, and ACAR components.

Object Detector: For the object detection step, we use the Swin-T detector, generated by combining a Cascade R-CNN [246] with a Swin-T [240] backbone. The model is pre-trained on ImageNet and MS COCO, and fine-tuned on METEOR using the same settings as Swin-T [240]: multi-scale training [247] (resizing the input with the shorter side between 480 and 800 and the longer side at most 1333), AdamW [248] optimizer (initial learning rate of $1\text{e}{-4}$, weight decay of 0.05, and batch size of 16), and $1\times$ schedule (12 epochs).

Backbone Network: Following ACAR-Net [245], we use SlowFast networks [249] as the backbone in the localization framework and double the spatial resolution of res5. We conduct experiments using a SlowFast R-101 8×8, pre-trained on the Kinetics-700 dataset [250], without non-local blocks. The inputs are 64-frame clips, where we sample $T = 8$ frames with a temporal stride $\tau = 8$ for the slow pathway, and $\alpha T (\alpha = 4)$ frames for the fast pathway.

Training Settings: We train ACAR-Net using synchronous SGD with a batch size of 16. For the first 3 epochs, we use a base learning rate of 0.008, which is then decreased by a factor of 10 at iterations 4 epochs and 5 epochs. We use a weight decay of $1\text{e}{-7}$ and Nesterov momentum of 0.9. We use both ground-truth boxes and predicted object boxes for training. For inference, we scale the shorter side of input frames to 384 pixels and use detected object boxes with scores greater than 0.85 for final behavior classification.

Results: We compare METEOR with the AVA dataset [244] as the latter is the state-of-the-art in multi-agent action recognition. In Table 5.6, we show that the current state-of-the-art approach, ACAR, achieves 30.0% mAP on AVA but yields 6.1% mAP on METEOR. There are several reasons why ACAR performs better on AVA. AVA focuses exclusively on only one target, humans, a category which most state-of-the-art object detectors can detect with ease. Furthermore, the videos in the AVA dataset consist of high-definition movies, in which agents (actors) are clearly visible, the background is simple, and the movements performed are also exaggerated and easier to identify. METEOR, on the other hand consists of 16 different categories of agents from vehicles to animals, most of which are novel for most detectors and therefore hard to detect. Moreover, the movements of the agents on the road are very fast, making them hard to capture. Finally, different agents have different motion patterns; for example, pedestrians move differently than vehicles and buses move differently than motorbikes. All of these factors collectively contribute to the complexity of MABR in dense, heterogeneous, and unstructured traffic scenarios. Our experiments and analysis show that there is much room for improvement and our hope with METEOR is that it provides the research community the resources it needs to tackle this important problem.

5.5 Conclusion

We present a new dataset, METEOR, for autonomous driving applications in dense, heterogeneous, and unstructured traffic scenarios. rain consists of more than 1000 one-minute video clips, over 2 million annotated frames with 2D and GPS trajectories for 16 unique agent categories, and more than 13 million bounding boxes for traffic agents. We found that current models for object

detection and multi-agent behavior prediction fail on the METEOR dataset. METEOR marks the first step towards the development of more sophisticated and robust perception models for dense, heterogeneous, and unstructured scenarios.

Our dataset has some limitations. While METEOR contains bounding box information for the surrounding agents, we currently do not provide trajectory information from a fixed reference frame. One would have to use depth estimation techniques to extract such trajectories. Furthermore, our dataset does not contain HD maps ad pointcloud data, which are used in many applications. For future work, we hope that our dataset can benefit in terms of design and evaluation of new motion forecasting and behavior prediction algorithms in dense and heterogeneous traffic. Finally, we hope to include semantic segmentation capability as part of METEOR by providing pixel labels for each object.

Chapter 6: Conclusion

This dissertation addressed many key problems in autonomous driving towards handling dense, heterogeneous, and unstructured traffic environments. We developed new techniques to perceive, predict, and plan among human drivers in traffic that is significantly denser in terms of number of traffic-agents, more heterogeneous in terms of size and dynamic constraints of traffic agents, and where many drivers do not follow the traffic rules. In this thesis, we present work along three themes—perception, driver behavior modeling, and planning. Our novel contributions include:

1. Improved tracking and trajectory prediction algorithms for dense and heterogeneous traffic using a combination of computer vision and deep learning techniques.

2. A novel behavior modeling approach using graph theory for characterizing human drivers as aggressive or conservative from their trajectories.

3. Behavior-driven planning and navigation algorithms in mixed (human driver and AV) and unstructured traffic environments using game theory and risk-aware control.

Additionally, we have released a new traffic dataset, METEOR, which captures rare and interesting, multi-agent driving behaviors in India. These behaviors are grouped into traffic violations, atypical interactions, and diverse scenarios. We evaluate our perception work on

tracking and trajectory prediction using standard autonomous driving datasets such as the Waymo Open Motion, Argoverse, NuScenes datasets, as well as public leaderboards where our tracking approach resulted in achieving rank 1 among over a 100 methods. We apply human driver behavior modeling in planning and navigation at unsignaled intersections and highways scenarios using state-of-the-art traffic simulators and show that our approach yields fewer collisions and deadlocks compared to methods based on deep reinforcement learning. We conclude the presentation with a discussion on future work.

Bibliography

[1] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple Online and Realtime Tracking with a Deep Association Metric. *arXiv preprint arXiv:1703.07402*, March 2017.

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

[3] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[4] Chen Long, Ai Haizhou, Zhuang Zijie, and Shang Chong. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[5] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE international conference on computer vision*, pages 4705–4713, 2015.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *ArXiv e-prints*, March 2017.

[7] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[8] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. Grip: Graph-based interaction-aware trajectory prediction. *arXiv preprint arXiv:1907.07792*, 2019.

[9] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. *ArXiv e-prints*, 2018.

[10] Fridulv Sagberg, Selpi, Giulio Francesco Bianchi Piccinini, and Johan Engström. A review of research on driving styles and road safety. *Human factors*, 57(7):1248–1275, 2015.

[11] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. Densepeds: Pedestrian tracking in dense crowds using front-rvo and sparse features. *arXiv preprint arXiv:1906.10313*, 2019.

[13] Mingyu Wang, Negar Mehr, Adrien Gaidon, and Mac Schwager. Game-theoretic planning for risk-aware interactive agents. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6998–7005. IEEE, 2020.

[14] Rohan Chandra, Uttaran Bhattacharya, Trisha Mittal, Aniket Bera, and Dinesh Manocha. Cmetric: A driving behavior measure using centrality functions. *arXiv preprint arXiv:2003.04424*, 2020.

[15] Rohan Chandra, Uttaran Bhattacharya, Trisha Mittal, Xiaoyu Li, Aniket Bera, and Dinesh Manocha. Graphrqi: Classifying driver behaviors using graph spectrums. *arXiv preprint arXiv:1910.00049*, 2019.

[16] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[17] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *ArXiv e-prints*, November 2013.

[19] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[20] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *ArXiv e-prints*, June 2015.

[21] Ruilong Li, Xin Dong, Zixi Cai, Dingcheng Yang, Haozhi Huang, Song-Hai Zhang, Paul Rosin, and Shi-Min Hu. Pose2seg: Human instance segmentation without detection. *arXiv preprint arXiv:1803.10683*, 2018.

[22] Jinshi Cui, Hongbin Zha, Huijing Zhao, and Ryosuke Shibasaki. Tracking multiple people using laser and vision. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2116–2121. IEEE, 2005.

[23] Louis Kratz and Ko Nishino. Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (99):1–1, 2011.

[24] Allison Bruce and Geoffrey Gordon. Better motion prediction for people-tracking. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2004.

[25] Haifeng Gong, Jack Sim, Maxim Likhachev, and Jianbo Shi. Multi-hypothesis motion planning for visual object tracking. In *2011 International Conference on Computer Vision*, pages 619–626. IEEE, 2011.

[26] Lin Liao, Dieter Fox, Jeffrey Hightower, Henry Kautz, and Dirk Schulz. Voronoi tracking: Location estimation using sparse and noisy sensor data. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[27] Ramin Mehran, Alexis Oyama, and Mubarak Shah. Abnormal crowd behavior detection using social force model. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition,CVPR*, pages 935–942. IEEE, 2009.

[28] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *IEEE International Conference on Computer Vision (ICCV)*, pages 261–268. IEEE, 2009.

[29] Aniket Bera and Dinesh Manocha. REACH: Realtime crowd tracking using a hybrid motion model. *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2015.

[30] Aniket Bera, Sujeong Kim, Tanmay Randhavane, Srihari Pratapa, and Dinesh Manocha. Glmp-realtime pedestrian path prediction using global and local movement patterns. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5528–5535. IEEE, 2016.

[31] Frank Dellaert and Chuck Thorpe. Robust car tracking using kalman filtering and bayesian templates. In *Conference on intelligent transportation systems*, volume 1, 1997.

[32] Daniel Streller, K Furstenberg, and Klaus Dietmayer. Vehicle and object models for robust tracking in traffic scenes using laser range images. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 118–123. IEEE, 2002.

[33] Anna Petrovskaya and Sebastian Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009.

[34] Andreas Ess, Konrad Schindler, Bastian Leibe, and Luc Van Gool. Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research*, 29(14):1707–1725, 2010.

[35] Akshay Rangesh and Mohan M Trivedi. No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras & lidars. *arXiv preprint arXiv:1802.08755*, 2018.

[36] Michael Darms, Paul Rybski, and Chris Urmson. Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1197–1202. IEEE, 2008.

[37] Julien Moras, Véronique Cherfaoui, and Philippe Bonnifait. Credibilist occupancy grids for vehicle perception in dynamic environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 84–89. IEEE, 2011.

[38] Nicolai Wojke and Marcel Häselich. Moving vehicle detection and tracking in unstructured environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3082–3087. IEEE, 2012.

[39] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271–288, 1998.

[40] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, March 2016.

[41] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):58–72, 2013.

[42] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696–4704, 2015.

[43] Hao Sheng, Li Hao, Jiahui Chen, Yang Zhang, and Wei Ke. Robust local effective matching model for multi-target tracking. In *Pacific Rim Conference on Multimedia*, pages 233–243. Springer, 2017.

[44] Aniket Bera and Dinesh Manocha. Realtime multilevel crowd tracking using reciprocal velocity obstacles. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 4164–4169. IEEE, 2014.

[45] Rohan Chandra, Uttaran Bhattacharya, Tanmay Randhavane, Aniket Bera, and Dinesh Manocha. Roadtrack: Realtime tracking of road agents in dense and heterogeneous environments. *arXiv*, pages arXiv–1906, 2019.

[46] Aniket Bera, Nico Galoppo, Dillon Sharlet, Adam Lake, and Dinesh Manocha. Adapt: real-time adaptive pedestrian tracking for crowded scenes. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1801–1808. IEEE, 2014.

[47] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, Sept 2009.

[48] Kota Yamaguchi, Alexander C Berg, Luis E Ortiz, and Tamara L Berg. Who are you with and where are you going? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1345–1352. IEEE, 2011.

[49] Ioannis Karamouzas, Peter Heil, Pascal Van Beek, and Mark H Overmars. A predictive collision avoidance model for pedestrian simulation. In *International Workshop on Motion in Games*, pages 41–52. Springer, 2009.

[50] Gianluca Antonini, Santiago Venegas Martinez, Michel Bierlaire, and Jean Philippe Thiran. Behavioral priors for detection and tracking of pedestrians in video sequences. *International Journal of Computer Vision*, 69(2):159–180, 2006.

[51] Shu-Yun Chung and Han-Pang Huang. A mobile robot that understands pedestrian spatial behaviors. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5861–5866. IEEE, 2010.

[52] Harold W Kuhn. The hungarian method for the assignment problem. In *50 Years of Integer Programming 1958-2008*, pages 29–47. Springer, 2010.

[53] Michael Levandowsky and David Winter. Distance between sets. *Nature*, 234(5323):34, 1971.

[54] Edward Twitchell Hall. *The hidden dimension*, volume 609. Garden City, NY: Doubleday, 1966.

[55] Satoru Satake, Takayuki Kanda, Dylan F Glas, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. How to approach humans?: strategies for social robots to initiate interaction. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 109–116. ACM, 2009.

[56] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 300–311, 2017.

[57] Kuan Fang, Yu Xiang, Xiaocheng Li, and Silvio Savarese. Recurrent autoregressive networks for online multi-object tracking. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 466–475. IEEE, 2018.

[58] Qing Zhang, Mengru Zhang, Mengdi Wang, Wanchen Sui, Chen Meng, Jun Yang, Weidan Kong, Xiaoyuan Cui, and Wei Lin. Efficient deep learning inference based on model compression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[59] Meiqi Wang, Zhisheng Wang, Jinming Lu, Jun Lin, and Zhongfeng Wang. E-lstm: An efficient hardware architecture for long short-term memory. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2019.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[61] Long Chen, Haizhou Ai, Chong Shang, Zijie Zhuang, and Bo Bai. Online multi-object tracking with convolutional neural networks. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 645–649. IEEE, 2017.

[62] Min Yang, Yuwei Wu, and Yunde Jia. A hybrid data association framework for robust online multi-object tracking. *arXiv preprint arXiv:1703.10764*, 2017.

[63] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4846–4855, 2017.

[64] Ricardo Sanchez-Matilla, Fabio Poiesi, and Andrea Cavallaro. Online multi-target tracking with strong and weak detections. In *European Conference on Computer Vision*, pages 84–99. Springer, 2016.

[65] Rohan Chandra, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8483–8492, June 2019.

[66] 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.

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

[68] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[69] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 336–345, 2017.

[70] Fang-Chieh Chou, Tsung-Han Lin, Henggang Cui, Vladan Radosavljevic, Thi Nguyen, Tzu-Kuo Huang, Matthew Niedoba, Jeff Schneider, and Nemanja Djuric. Predicting motion of vulnerable road users using high-definition maps and efficient convnets. *arXiv preprint arXiv:1906.08469*, 2019.

[71] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, and J. Schneider. Short-term Motion Prediction of Traffic Actors for Autonomous Driving using Deep Convolutional Networks. *ArXiv e-prints*, August 2018.

[72] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 6120–6127, 2019.

[73] U.S. Federal Highway Administration. U.s. highway 101 and i-80 dataset. 2005.

[74] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 32, pages 1231–1237. Sage Publications Sage UK: London, England, 2012.

[75] Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50):24972–24978, 2019.

[76] Zhang Wei-hua. Selected model and sensitivity analysis of aggressive driving behavior. volume 25, pages 106–112. Xi'an Highway University, 2012.

[77] Murray R Barrick and Michael K Mount. The big five personality dimensions and job performance: a meta-analysis. *Personnel psychology*, 44(1):1–26, 1991.

[78] Barbara Krahé and Ilka Fenske. Predicting aggressive driving behavior: The role of macho personality, age, and power of car. *Aggressive Behavior: Official Journal of the International Society for Research on Aggression*, 28(1):21–29, 2002.

[79] Jian Rong, Kejun Mao, and Jianming Ma. Effects of individual differences on driving behavior and traffic flow characteristics. *Transportation research record*, 2248(1):1–9, 2011.

[80] Eric R Dahlen, Bryan D Edwards, Travis Tubré, Michael J Zyphur, and Christopher R Warren. Taking a look behind the wheel: An investigation into the personality predictors of aggressive driving. *Accident Analysis & Prevention*, 45:1–9, 2012.

[81] Kenneth H. Beck, Bina Ali, and Stacey B Daughters. Distress tolerance as a predictor of risky and aggressive driving. *Traffic injury prevention*, 15 4:349–54, 2014.

[82] A Hamish Jamson, Natasha Merat, Oliver MJ Carsten, and Frank CH Lai. Behavioural changes in drivers experiencing highly-automated vehicle control in varying traffic conditions. *Transportation research part C: emerging technologies*, 30:116–125, 2013.

[83] Pirkko Rämä. Effects of weather-controlled variable speed limits and warning signs on driver behavior. *Transportation Research Record*, 1689(1):53–59, 1999.

[84] Jiangpeng Dai, Jin Teng, Xiaole Bai, Zhaohui Shen, and Dong Xuan. Mobile phone based drunk driving detection. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, 2010.

[85] Mohammad Saifuzzaman, Md Mazharul Haque, Zuduo Zheng, and Simon Washington. Impact of mobile phone use on car-following behaviour of young drivers. *Accident Analysis & Prevention*, 82:10–19, 2015.

[86] Ahmad Aljaafreh, Nabeel Alshabatat, and Munaf S. Najim Al-Din. Driving style recognition using fuzzy logic. *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, pages 460–463, 2012.

[87] Yi Lu Murphey, Richard Milton, and Leonidas Kiliaris. Driver's style classification using jerk analysis. *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, pages 23–28, 2009.

[88] Ishak Mohamad, Mohd. Alauddin Mohd. Ali, and Mahamod Ismail. Abnormal driving detection using real time global positioning system data. *Proceeding of the 2011 IEEE International Conference on Space Science and Communication (IconSpace)*, pages 1–6, 2011.

[89] Ernest Cheung, Aniket Bera, Emily Kubin, Kurt Gray, and Dinesh Manocha. Identifying driver behaviors using trajectory features for vehicle navigation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3445–3452. IEEE, 2018.

[90] Alex Zyner, Stewart Worrall, and Eduardo Nebot. A recurrent neural network solution for predicting driver intention at unsignalized intersections. *IEEE Robotics and Automation Letters*, 3(3):1759–1764, 2018.

[91] Alex Zyner, Stewart Worrall, James Ward, and Eduardo Nebot. Long short term memory for driver intent prediction. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1484–1489. IEEE, 2017.

[92] Geqi Qi, Yiman Du, Jianping Wu, and Ming Xu. Leveraging longitudinal driving behaviour data with data mining techniques for driving style analysis. *IET intelligent transport systems*, 9(8):792–801, 2015.

[93] Bin Shi, Li Xu, Jie Hu, Yun Tang, Hong Jiang, Wuqiang Meng, and Hui Liu. Evaluating driving styles by normalizing driving behavior based on personalized driver modeling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45:1502–1508, 2015.

[94] Wenshuo Wang, Junqiang Xi, Alexandre Chong, and Lin Li. Driving style classification using a semisupervised support vector machine. *IEEE Transactions on Human-Machine Systems*, 47:650–660, 2017.

[95] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee D. Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, 36(3, Part 2):6164–6173, 2009.

[96] Hongbin Yin, S.C. Wong, Jianmin Xu, and C.K. Wong. Urban traffic flow prediction using a fuzzy-neural approach. *Transportation Research Part C: Emerging Technologies*, 10(2):85–98, 2002.

[97] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, April 2015.

[98] Alireza Ermagun and David Levinson. Spatiotemporal traffic forecasting: review and proposed directions. *Transport Reviews*, 38(6):786–814, 2018.

[99] I. Lana, J. Del Ser, M. Velez, and E. I. Vlahogianni. Road traffic forecasting: Recent advances and new challenges. *IEEE Intelligent Transportation Systems Magazine*, 10(2):93–109, Summer 2018.

[100] X. Cheng, R. Zhang, J. Zhou, and W. Xu. Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2018.

[101] Chaoyun Zhang and Paul Patras. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Mobihoc '18, pages 231–240, New York, NY, USA, 2018. ACM.

[102] Rohan Chandra, Uttaran Bhattacharya, Christian Roncal, Aniket Bera, and Dinesh 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*, pages 1–9, 2019.

[103] Sujeong Kim, Stephen J Guy, Wenxi Liu, David Wilkie, Rynson WH Lau, Ming C Lin, and Dinesh Manocha. Brvo: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research*, 34(2):201–217, 2015.

[104] Rohan Chandra, Tianrui Guan, Srujan Panuganti, Trisha Mittal, Uttaran Bhattacharya, Aniket Bera, and Dinesh Manocha. Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms. *IEEE Robotics and Automation Letters*, 2020.

[105] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[106] James W Demmel. *Applied numerical linear algebra*, volume 56. Siam, 1997.

[107] Rose Yu, Stephan Zheng, Anima Anandkumar, and Yisong Yue. Long-term forecasting using tensor-train rnns. *arXiv preprint arXiv:1711.00073*, 2017.

[108] Braxton Osting, Chris D White, and Édouard Oudet. Minimal dirichlet energy partitions for graphs. *SIAM Journal on Scientific Computing*, 2014.

[109] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 2007.

[110] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[111] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[112] Tianrui Guan, Jun Wang, Shiyi Lan, Rohan Chandra, Zuxuan Wu, Larry Davis, and Dinesh Manocha. M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers. *arXiv preprint arXiv:2104.11896*, 2021.

[113] Rohan Chandra and Dinesh Manocha. Gameplan: Game-theoretic multi-agent planning with human drivers at intersections, roundabouts, and merging. *arXiv preprint arXiv:2109.01896*, 2021.

[114] Angelos Mavrogiannis, Rohan Chandra, and Dinesh Manocha. B-gap: Behavior-guided action prediction for autonomous navigation. *arXiv preprint arXiv:2011.03748*, 2020.

[115] Rohan Chandra, Mridul Mahajan, Rahul Kala, Rishitha Palugulla, Chandrababu Naidu, Alok Jain, and Dinesh Manocha. Meteor: A massive dense & heterogeneous behavior dataset for autonomous driving. *arXiv preprint arXiv:2109.07648*, 2021.

[116] Dev Seth and Mary L Cummings. Traffic efficiency and safety impacts of autonomous vehicle aggressiveness. *simulation*, 19:20, 2019.

[117] Xiaowei Shi, Zhen Wang, Xiaopeng Li, and Mingyang Pei. The effect of ride experience on changing opinions toward autonomous vehicle safety. *Communications in Transportation Research*, 1:100003, 2021.

[118] Dirty Tesla. 25 miles of full self driving — tesla challenge 2 — autopilot —. `https://www.youtube.com/watch?v=Rm8aPR0aMDE`, 2019.

[119] Neil C Rabinowitz, Frank Perbet, H Francis Song, Chiyuan Zhang, SM Eslami, and Matthew Botvinick. Machine theory of mind. *arXiv preprint arXiv:1802.07740*, 2018.

[120] Rohan Chandra, Aniket Bera, and Dinesh Manocha. Stylepredict: Machine theory of mind for human driver behavior from trajectories. *arXiv preprint arXiv:2011.04816*, 2020.

[121] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[122] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[123] Alex Davies. Google's self-driving car caused its first crash, 2016.

[124] Francisco Aparecido Rodrigues. Network centrality: An introduction. *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*, page 177, 2019.

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

[126] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[127] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):11, 2019.

[128] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[129] Minju Park, Kitae Jang, Jinwoo Lee, and Hwasoo Yeo. Logistic regression model for discretionary lane changing under congested traffic. *Transportmetrica A: transport science*, 11(4):333–344, 2015.

[130] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[131] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, 2016.

[132] Dave Ferguson, Thomas M Howard, and Maxim Likhachev. Motion planning in urban environments: Part ii. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1070–1076. IEEE, 2008.

[133] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[134] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[135] Tingxiang Fan, Xinjing Cheng, Jia Pan, Dinesh Monacha, and Ruigang Yang. Crowdmove: Autonomous mapless navigation in crowded scenarios. *arXiv preprint:1807.07870*, 2018.

[136] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia 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.

[137] Michael Everett, Yu Fan Chen, and Jonathan 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)*, pages 3052–3059. IEEE, 2018.

[138] Chengxi Li, Yue Meng, Stanley H Chan, and Yi-Ting Chen. Learning 3d-aware egocentric spatial-temporal interaction via graph convolutional networks. *arXiv preprint arXiv:1909.09272*, 2019.

[139] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[140] NVIDIA 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.

[141] Sylvia A Morelli, Desmond C Ong, Rucha Makati, Matthew O Jackson, and Jamil Zaki. Empathy and well-being correlate with centrality in different social networks. *Proceedings of the National Academy of Sciences*, 114(37):9843–9847, 2017.

[142] Glenn Lawyer. Understanding the influence of all nodes in a network. *Scientific reports*, 5(1):1–9, 2015.

[143] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[144] Mile Šikić, Alen Lančić, Nino Antulov-Fantulin, and Hrvoje Štefančić. Epidemic centrality—is there an underestimated epidemic impact of network peripheral nodes? *The European Physical Journal B*, 86(10):440, 2013.

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

[146] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

[147] Kazi Iftekhar Ahmed. *Modeling drivers' acceleration and lane changing behavior*. PhD thesis, MIT, 1999.

[148] Peng Wang, Xinyu Huang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[149] Edouard Leurent and Jean Mercat. Social attention for autonomous decision-making in dense traffic. *arXiv preprint arXiv:1911.12250*, 2019.

[150] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818. IEEE, 2017.

[151] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[152] Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model mobil for car-following models. *Transportation Research Record*, 1999(1):86–94, 2007.

[153] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[154] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[155] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 387–396. ACM, 2017.

[156] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[157] James W Demmel. *Applied numerical linear algebra*, volume 56. Siam, 1997.

[158] William W Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989.

[159] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications*, 415(1):20–30, 2006.

[160] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. Timers: Error-bounded svd restart on dynamic networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[161] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[162] Virginia Vassilevska Williams. Multiplying matrices in o (n2. 373) time. 2014.

[163] Meha Kaushik, K Madhava Krishna, et al. Parameter sharing reinforcement learning architecture for multi agent driving behaviors. *arXiv preprint arXiv:1811.07214*, 2018.

[164] Meha Kaushik, Vignesh Prasad, K Madhava Krishna, and Balaraman Ravindran. Overtaking maneuvers in simulated highway driving using deep reinforcement learning. In *2018 ieee intelligent vehicles symposium (iv)*, pages 1885–1890. IEEE, 2018.

[165] Haoran Song, Wenchao Ding, Yuxuan Chen, Shaojie Shen, Michael Yu Wang, and Qifeng Chen. PiP: Planning-Informed Trajectory Prediction for Autonomous Driving. *Lecture Notes in Computer Science*, page 598–614, 2020.

[166] Sam Anthony. Self-driving cars still can't mimic the most natural human behavior. `https://qz.com/1064004/self-driving-cars-still-cant-mimic-the-most-natural-human-behavior/`, 2017.

[167] Erik Vinkhuyzen and Melissa Cefkin. Developing socially acceptable autonomous vehicles. In *Ethnographic Praxis in Industry Conference Proceedings*, volume 2016, pages 522–534. Wiley Online Library, 2016.

[168] Anup Doshi and Mohan M Trivedi. Examining the impact of driving style on the predictability and responsiveness of the driver: Real-world and simulator analysis. In *2010 IEEE Intelligent Vehicles Symposium*, pages 232–237. IEEE, 2010.

[169] Andrea Corti, Carlo Ongini, Mara Tanelli, and Sergio M Savaresi. Quantitative driving style estimation for energy-oriented applications in road vehicles. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3710–3715. IEEE, 2013.

[170] Alessandro Paolo Capasso, Paolo Maramotti, Anthony Dell'Eva, and Alberto Broggi. End-to-end intersection handling using multi-agent deep reinforcement learning. *arXiv preprint arXiv:2104.13617*, 2021.

[171] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039. IEEE, 2018.

[172] Shixiong Kai, Bin Wang, D. Chen, Jianye Hao, Hongbo Zhang, and Wulong Liu. A multi-task reinforcement learning approach for navigating unsignalized intersections. *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1583–1588, 2020.

[173] Teng Liu, Xingyu Mu, Bing Huang, Xiaolin Tang, Fuqing Zhao, Xiao Wang, and Dongpu Cao. Decision-making at unsignalized intersection for autonomous vehicles: Left-turn maneuver with deep reinforcement learning. *arXiv preprint arXiv:2008.06595*, 2020.

[174] Nan Li, Yu Yao, Ilya Kolmanovsky, Ella Atkins, and Anouck R Girard. Game-theoretic modeling of multi-vehicle interactions at uncontrolled intersections. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[175] Ran Tian, Nan Li, Ilya Kolmanovsky, Yildiray Yildiz, and Anouck R Girard. Game-theoretic modeling of traffic in unsignalized intersection network for autonomous vehicle control verification and validation. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[176] Junha Roh, Christoforos Mavrogiannis, Rishabh Madan, Dieter Fox, and Siddhartha S Srinivasa. Multimodal trajectory prediction via topological invariance for navigation at uncontrolled intersections. *arXiv preprint arXiv:2011.03894*, 2020.

[177] Noam Buckman, Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela L Rus. Sharing is caring: Socially-compliant autonomous intersection negotiation. 2020.

[178] Matteo Vasirani and Sascha Ossowski. A market-inspired approach for intersection management in urban road traffic networks. *Journal of Artificial Intelligence Research*, 43:621–659, 2012.

[179] Andrea Censi, Saverio Bolognani, Julian G Zilly, Shima Sadat Mousavi, and Emilio Frazzoli. Today me, tomorrow thee: Efficient resource allocation in competitive settings using karma games. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 686–693. IEEE, 2019.

[180] DianChao Lin and Saif Eddin Jabari. Pay for intersection priority: A free market mechanism for connected vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[181] Dustin Carlino, Stephen D Boyles, and Peter Stone. Auction-based autonomous intersection management. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 529–534. IEEE, 2013.

[182] David Rey, Michael W Levin, and Vinayak V Dixit. Online incentive-compatible mechanisms for traffic intersection auctions. *European Journal of Operational Research*, 2021.

[183] Muhammed O Sayin, Chung-Wei Lin, Shinichi Shiraishi, Jiajun Shen, and Tamer Başar. Information-driven autonomous intersection control via incentive compatible mechanisms. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):912–924, 2018.

[184] Tim Roughgarden. *Twenty lectures on algorithmic game theory*. Cambridge University Press, 2016.

[185] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.

[186] Heiko Schepperle and Klemens Böhm. Agent-based traffic control using auctions. In *International Workshop on Cooperative Information Agents*, pages 119–133. Springer, 2007.

[187] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. Lucidgames: Online unscented inverse dynamic games for adaptive trajectory prediction and planning. *arXiv preprint arXiv:2011.08152*, 2020.

[188] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. Algames: A fast solver for constrained dynamic games. *arXiv preprint arXiv:1910.09713*, 2019.

[189] Jaime F Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S Shankar Sastry, and Anca D Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9590–9596. IEEE, 2019.

[190] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. 2009.

[191] Kuanqi Cai, Chaoqun Wang, Shuang Song, Haoyao Chen, and Max Q-H Meng. Risk-aware path planning under uncertainty in dynamic environments. *Journal of Intelligent & Robotic Systems*, 101(3):1–15, 2021.

[192] Anirudha Majumdar, Sumeet Singh, Ajay Mandlekar, and Marco Pavone. Risk-sensitive inverse reinforcement learning via coherent risk models. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[193] Lillian J Ratliff and Eric Mazumdar. Inverse risk-sensitive reinforcement learning. *IEEE Transactions on Automatic Control*, 65(3):1256–1263, 2019.

[194] Minae Kwon, Erdem Biyik, Aditi Talati, Karan Bhasin, Dylan P. Losey, and Dorsa Sadigh. When humans aren't optimal: Robots that collaborate with risk-aware humans. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, page 43–52. Association for Computing Machinery, 2020.

[195] Hans Föllmer and Thomas Knispel. Entropic risk measures: Coherence vs. convexity, model ambiguity and robust large deviations. *Stochastics and Dynamics*, 11(02n03):333–351, 2011.

[196] R Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

[197] Peter Whittle and Peter R Whittle. *Risk-sensitive optimal control*, volume 20. Wiley New York, 1990.

[198] Wendell H Fleming and William M McEneaney. Risk sensitive optimal control and differential games. In *Stochastic theory and adaptive control*, pages 185–197. Springer, 1992.

[199] Takayuki Osogami. Robustness and risk-sensitivity in markov decision processes. In *Advances in Neural Information Processing Systems 25*, pages 233–241. Curran Associates, Inc., 2012.

[200] Yin-Lam Chow and Marco Pavone. A framework for time-consistent, risk-averse model predictive control: Theory and algorithms. In *2014 American Control Conference*, pages 4204–4211. IEEE, 2014.

[201] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Advances in Neural Information Processing Systems 28*, pages 1522–1530. Curran Associates, Inc., 2015.

[202] Samantha Samuelson and Insoon Yang. Safety-aware optimal control of stochastic systems using conditional value-at-risk. In *2018 American Control Conference*, pages 6285–6290. IEEE, 2018.

[203] Margaret P Chapman, Jonathan Lacotte, Aviv Tamar, Donggun Lee, Kevin M Smith, Victoria Cheng, Jaime F Fisac, Susmit Jha, Marco Pavone, and Claire J Tomlin. A risk-sensitive finite-time reachability approach for safety of stochastic dynamic systems. In *2019 American Control Conference*, pages 2958–2963. IEEE, 2019.

[204] Peter Whittle. Risk-sensitive linear/quadratic/gaussian control. *Advances in Applied Probability*, 13(4):764–777, 1981.

[205] Arvind A Pereira, Jonathan Binney, Geoffrey A Hollinger, and Gaurav S Sukhatme. Risk-aware path planning for autonomous underwater vehicles using predictive ocean models. *Journal of Field Robotics*, 30(5):741–762, 2013.

[206] Vishnu D Sharma, Maymoonah Toubeh, Lifeng Zhou, and Pratap Tokekar. Risk-aware planning and assignment for ground vehicles using uncertain perception from aerial vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11763–11769. IEEE, 2020.

[207] Vishnu Dutt Sharma and Pratap Tokekar. Risk-aware path planning for ground vehicles using occluded aerial images. *arXiv preprint arXiv:2104.11709*, 2021.

[208] Arnav Choudhry, Brady Moon, Jay Patrikar, Constantine Samaras, and Sebastian Scherer. Cvar-based flight energy risk assessment for multirotor uavs using a deep energy model. *arXiv preprint arXiv:2105.15189*, 2021.

[209] Vishnu D Sharma, Maymoonah Toubeh, Lifeng Zhou, and Pratap Tokekar. Risk-aware planning and assignment for ground vehicles using uncertain perception from aerial vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11763–11769. IEEE, 2020.

[210] Vishnu D Sharma, Maymoonah Toubeh, Lifeng Zhou, and Pratap Tokekar. Risk-aware planning and assignment for ground vehicles using uncertain perception from aerial vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11763–11769. IEEE, 2020.

[211] Vishnu D Sharma, Maymoonah Toubeh, Lifeng Zhou, and Pratap Tokekar. Risk-aware planning and assignment for ground vehicles using uncertain perception from aerial vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11763–11769. IEEE, 2020.

[212] Orit Taubman-Ben-Ari, Mario Mikulincer, and Omri Gillath. The multidimensional driving style inventory—scale construct and validation. *Accident Analysis & Prevention*, 36(3):323–332, 2004.

[213] E Gulian, G Matthews, Aleck Ian Glendon, DR Davies, and LM Debney. Dimensions of driver stress. *Ergonomics*, 1989.

[214] Davina J French, Robert J West, James Elander, and John Martin WILDING. Decision-making style, driving style, and self-reported involvement in road traffic accidents. *Ergonomics*, 36(6):627–644, 1993.

[215] Jerry L Deffenbacher, Eugene R Oetting, and Rebekah S Lynch. Development of a driving anger scale. *Psychological reports*, 1994.

[216] Motonori Ishibashi, Masayuki Okuwa, Shun'ichi Doi, and Motoyuki Akamatsu. Indices for characterizing driving style and their relevance to car following behavior. In *SICE Annual Conference 2007*, pages 1132–1137. IEEE, 2007.

[217] Stephen J. Guy, Sujeong Kim, M. Chiao Lin, and Dinesh Manocha. Simulating heterogeneous crowd behaviorsusing personality trait theory. In *Symposium on Computer Animation*, 2011.

[218] Aniket Bera, Tanmay Randhavane, and Dinesh Manocha. Aggressive, tense or shy? identifying personality traits from crowd videos. In *IJCAI*, 2017.

[219] J Christopher Brill, Mustapha Mouloua, Edwin Shirkey, and Pascal Alberti. Predictive validity of the aggressive driver behavior questionnaire (adbq) in a simulated environment. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 53, pages 1334–1337. SAGE Publications Sage CA: Los Angeles, CA, 2009.

[220] Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

[221] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision (ECCV)*, pages 414–430. Springer, 2020.

[222] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[223] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. *arXiv preprint arXiv:2104.10133*, 2021.

[224] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[225] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Level 5 perception dataset 2020. `https://level-5.global/level5/data/`, 2019.

[226] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020.

[227] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019.

[228] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

[229] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[230] Girish Varma, Anbumani Subramanian, Anoop Namboodiri, Manmohan Chandraker, and CV Jawahar. Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019.

[231] Vasili Ramanishka, Yi-Ting Chen, Teruhisa Misu, and Kate Saenko. Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7699–7707, 2018.

[232] Ashesh Jain, Hema S Koppula, Shane Soh, Bharad Raghavan, Avi Singh, and Ashutosh Saxena. Brain4cars: Car that knows before you do via sensory-fusion deep learning architecture. *arXiv preprint arXiv:1601.00740*, 2016.

[233] Zhengping Che, Guangyu Li, Tracy Li, Bo Jiang, Xuefeng Shi, Xinsheng Zhang, Ying Lu, Guobin Wu, Yan Liu, and Jieping Ye. D2-city: A large-scale dashcam video dataset of diverse traffic scenarios. *arXiv preprint arXiv:1904.01975*, 2019.

[234] Xin Wang Wenqi Xian Yingying Chen, Fangchen Liu Vashisht Madhavan Trevor Darrell, Fisher Yu, and Haofeng Chen. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. *arXiv preprint arXiv: 1805.04687*, 2018.

[235] Aymery Constant and Emmanuel Lagarde. Protecting vulnerable road users from injury. *PLoS medicine*, 7(3):e1000228, 2010.

[236] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*. Springer, 2020.

[237] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[238] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[239] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[240] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021.

[241] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*. Springer, 2014.

[242] R. Padilla, S. L. Netto, and E. A. B. da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020.

[243] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009.

[244] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6047–6056, 2018.

[245] Junting Pan, Siyu Chen, Mike Zheng Shou, Yu Liu, Jing Shao, and Hongsheng Li. Actor-context-actor relation network for spatio-temporal action localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 464–474, 2021.

[246] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6154–6162, 2018.

[247] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pages 213–229. Springer, 2020.

[248] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[249] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 6202–6211, 2019.

[250] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.