RobustTP: End-to-End Trajectory Prediction for Heterogeneous Road-Agents in Dense Traffic with Noisy Sensor Inputs

Rohan Chandra
University of Maryland, College Park

Uttaran Bhattacharya
University of Maryland, College Park

Christian Roncal
University of Maryland, College Park

Aniket Bera
University of North Carolina, Chapel Hill

Dinesh Manocha
University of Maryland, College Park



Figure 1: Trajectory Prediction Results: We highlight the performance of various end-to-end trajectory prediction methods on the TRAF dataset with different types of road-agents. We showcase six scenarios with different densities, heterogeneity, camera positions (fixed or moving), times 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 prediction results for our approach, RobustTP are shown as a solid red line. The prediction results of other methods (RNN-Encoder Decoder, Social-GAN, CS-LSTM) are drawn with different dashed lines. The green circles denote the road-agent whose trajectory is being predicted. RobustTP predictions are closest to GT in all the scenarios. We observe up to 18% improvement in average displacement error and up to 35.5% final displacement error over prior methods for dense, heterogeneous traffic.

ABSTRACT

We present RobustTP, 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. RobustTP 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CSCS '19, October 8, 2019, Kaiserslautern, Germany
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-7004-2/19/10…$15.00
https://doi.org/10.1145/3359999.3360495

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.

CCS CONCEPTS

• **Computing methodologies → Tracking**; **Machine learning**; *Artificial intelligence*.

KEYWORDS

Trajectory Prediction, Tracking, Deep Learning, Machine Learning

ACM Reference Format:
Rohan Chandra, Uttaran Bhattacharya, Christian Roncal, Aniket Bera, and Dinesh Manocha. 2019. RobustTP: End-to-End Trajectory Prediction for Heterogeneous Road-Agents in Dense Traffic with Noisy Sensor Inputs. In *COMPUTER SCIENCE IN CARS SYMPOSIUM (CSCS '19), October 8, 2019, Kaiserslautern, Germany*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3359999.3360495

1 INTRODUCTION

Increasingly powerful GPUs and advanced computer vision tools have made it possible to perform end-to-end, realtime tracking of heterogeneous road-agents such as cars, pedestrians, two-wheelers, etc. These tools can further be used for many applications such as autonomous driving, surveillance, action recognition, and collision-free navigation. In addition to tracking, these tools are also essential to predicting the future trajectory of each road agent. The predicted trajectories are useful for performing safe autonomous navigation, traffic forecasting, vehicle routing, and congestion management [Deo et al. 2018; Schreier et al. 2014].

In this paper, we focus on dense traffic composed of heterogeneous road agents. The traffic density corresponds to the number of distinct road agents captured in a single frame of the video or the number of agents per unit length (*e.g.*, a kilometer) of the roadway. High-density traffic is described as traffic with more than 100 road agents per kilometer. The heterogeneity corresponds to different types of road agents with varying dynamics such as cars, buses, pedestrians, two-wheelers (scooters and motorcycles), three-wheelers (rickshaws), animals, etc. These agents have different shapes, dynamic constraints, and levels of maneuverability. The difficulty of performing trajectory prediction increases in such traffic because the trajectory of any single road-agent is affected by other road-agents in close proximity. To accurately predict the trajectory, a model for interaction with nearby road-agents needs to be considered.

Many methods have been proposed for end-to-end trajectory prediction. The agent types can be both pedestrians [Alahi et al. 2016; Gupta et al. 2018] and traffic road-agents [Chandra et al. 2019; Deo and Trivedi 2018; Ma et al. 2018b]. However, a major disadvantage common to all of the above methods is that they rely on manually annotated trajectories that are often not readily available for training their models. Additionally, training manually annotated trajectories introduces dataset bias and results in over-fitting, thereby generating results that do not emulate real traffic scenarios. Our goal in this paper is to perform robust end-to-end trajectory prediction using sensor trajectories. The sensors, in this case, are static or moving RGB cameras. The trajectories are obtained using a tracking algorithm and thus contains noise. In this case, we consider noise as the deviation or perturbation from the ground truth trajectory. The amount of noise depends on the accuracy of the tracking algorithm.

The chance of collisions with other road-agents in close proximity increases in such traffic. Advanced Driver Assistance Systems (ADAS) help prevent or reduce the likelihood of traffic accidents by mitigating the adverse effects caused by human errors. ADAS collects information from a road-agent's surroundings and utilizes that information to implement critical actions to assist drivers. Predicting road-agents' trajectories is a crucial task for any ADAS to avoid collisions. Some ADAS applications for predicting a road-agent's trajectory have been proposed [Barth and Franke 2009; Houenou et al. 2013; Weiskircher and Ayalew 2015]. However, the methods are designed for simple road conditions with sparse traffic. Moreover, some approaches are computationally expensive as they rely on lidar-based 3D point clouds [Barth and Franke 2009]. Some algorithms are controls-based systems that are susceptible to highly dynamic environments such as dense and heterogeneous traffic [Weiskircher and Ayalew 2015].

Main Contributions.

(1) We present an end-to-end trajectory prediction approach, RobustTP, for road-agents in dense and heterogeneous traffic. The input to our algorithm is a video captured using a static or moving RGB camera and the output is the predicted trajectory over a span of 3-5 seconds. We outperform state-of-the-art methods for end-to-end trajectory prediction that use sensor inputs for training their models. We achieve an increase of up to 18% average displacement accuracy and an increase of 35.5% final displacement accuracy over the next best method. All experiments were set up on an Nvidia TiTan Xp GPU. Finally, RobustTP is a proof of concept for an Advanced Driver Assistance Systems (ADAS) application and can be integrated into current ADAS applications.
(2) Additionally, we build a software framework, TrackNPred, that contains implementations of many different tracking and trajectory prediction methods, including our novel algorithm, and tools for quickly applying them to other dense traffic datasets. The purpose of TrackNPred is twofold: to reduce collisions by computing safer paths in dense traffic and to create a common interface for many trajectory prediction approaches. TrackNPred can also benchmark different algorithms and generate performance comparisons using standard error measurement metrics on real-world dense and heterogeneous traffic datasets.

2 RELATED WORK

In this section, we give a brief overview previous work in trajectory prediction and Advanced Driver Assistance Systems (ADAS).

2.1 Trajectory Prediction

Trajectory prediction has been well studied through multiple approaches such as the Bayesian formulation [Lefèvre et al. 2011], the Monte Carlo simulation [Danielsson et al. 2007], Hidden Markov Models (HMMs) [Firl et al. 2012], control-based systems [Barth and Franke 2009] and kalman filters [Kalman 1960]. However, these

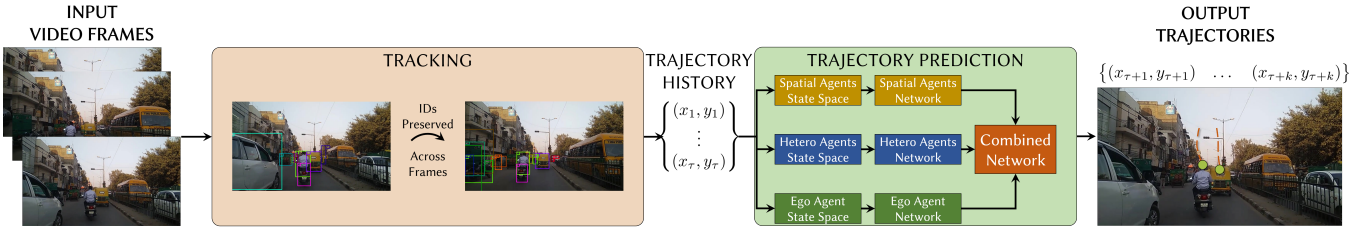


Figure 2: Overview of RobustTP: RobustTP is an end-to-end trajectory prediction algorithm that uses sensor input trajectories as training data instead of manually annotated trajectories. The sensor input is an RGB video from a moving or static camera. The first step is to compute trajectories using a tracking algorithm (light orange block). The details of the tracking algorithm are provided in Section 3.2. The trajectories generated are the training data for the trajectory prediction algorithm (green block), the details of which are provided in Section 3.3. 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.

methods are highly susceptible to dense and dynamics environments. [Barth and Franke 2009] is additionally computationally expensive as it relies on lidar-based 3D pointclouds.

Another line of research investigates trajectory prediction by modeling interactions between the road-agents, either explicitly or implicitly. These methods work by reducing the task of trajectory prediction to one of predicting sequences using neural networks. The sequence prediction model uses the past trajectories as training data and predicts spatial coordinates that form the future trajectory sequence. Examples of methods that explicitly model road-agent interaction include techniques based on social forces [Helbing and Molnar 1995; Yamaguchi et al. 2011], velocity obstacles [Van Den Berg et al. 2011], LTA [Pellegrini et al. 2009], 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 [Bera et al. 2016]. Techniques based on velocity obstacles have been extended using kinematic constraints to model the interactions between heterogeneous road agents [Ma et al. 2018a]. On the other hand, there are some methods [Chandra et al. 2019] that model the heterogeneous interactions between road agents implicitly.

Approaches based on deep learning use Recurrent Neural Networks (RNNs) and its variants for sequence modeling. The benefits of RNNs for sequence modeling makes them a reasonable choice for traffic prediction. Since RNNs cannot utilize information from too far back in its memory, many traffic trajectory prediction methods use long short-term memory networks (LSTMs) to predict trajectory sequences. These include algorithms to predict trajectories in traffic scenarios with few heterogeneous interactions [Deo and Trivedi 2018; Ma et al. 2018b]. Moreover, the advantage of our approach over these methods lies in our heterogeneous interaction module. In addition, we are lane-agnostic, that is, we do not assume vehicles conform to lane-driving, which is often the case in dense urban areas in developing countries. Trajectory prediction has also been used for pedestrians in a crowd [Alahi et al. 2016; Vemula et al. 2018]. In particular, Bhattacharyya et al. [Bhattacharyya et al. 2018] use a bayesian RNN network to capture the uncertainty in pedestrian motion to predict trajectories in pedestrian crowds.

RNN-based methods can be combined with other deep learning architectures to form hybrid networks for trajectory prediction. Some examples of deep learning architectures include CNNs, GANs, VAEs, and LSTMs. Each architecture has its own set of advantages and, for many tasks, the accuracy of the performance can be increased by combining the advantages of individual architectures. For example, generative models have been successfully used for tasks such as super resolution [Ledig et al. 2017], image-to-image translation [Isola et al. 2017], and image synthesis [Gregor et al. 2015]. 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 [Gupta et al. 2018] and in sparse traffic [Lee et al. 2017]. Alternatively, Convolutional Neural Networks (CNNs or ConvNets) have also been successfully used in many computer vision applications like object recognition [Zhao et al. 2018]. Recently, they have also been used for traffic trajectory prediction [Chou et al. 2019; Djuric et al. 2018].

2.2 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.

ADAS for trajectory prediction have been proposed [Barth and Franke 2009; Houenou et al. 2013; Weiskircher and Ayalew 2015]. However, [Houenou et al. 2013] uses a constant acceleration motion model to predict trajectories, which is unrealistic in dense and heterogeneous traffic. [Weiskircher and Ayalew 2015] uses a Model

Predictive Control based system, which is susceptible to highly dynamic environments. Lastly, [Barth and Franke 2009] uses a kalman filter approach and relies on lidars to collect 3D pointcloud information which is computationally expensive.

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.

3 ROBUSTTP: OVERVIEW AND ALGORITHM

In this section, 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.

3.1 Problem Setup and Notation

Given a set of N road agents $\mathcal{R} = \{r_i\}_{i=1\dots N}$, the trajectory history of each road agent r_i over τ frames, denoted $\mathcal{T}_i = \{(x_1, y_1), (x_2, y_2), \dots, (x_\tau, y_\tau)\}$, 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

$$\Omega_i := \begin{bmatrix} \mathcal{T}_i & \Delta\mathcal{T}_i & c & l \end{bmatrix}^\top, \tag{1}$$

where Δ is a derivative operator that is used to compute the velocity of the road agent, and $c := [c(x_1, y_1), \dots, c(x_\tau, y_\tau)]^\top$. 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 [Bouguet 2015; Bradski and Kaehler 2008] 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 [Alahi et al. 2016; Gupta et al. 2018]. 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).

3.2 Tracking by Detection

Manually labeled training data are not representative of real-world trajectories and thus do not produce realistic results. Moreover, they are not easily available for dense and heterogeneous traffic scenarios.

Instead, we implement an end-to-end system where a tracking algorithm computes the input trajectories that form the training set for the trajectory prediction algorithm. We first use instance segmentation based on the Mask R-CNN algorithm to generate background-subtracted representations of the identified road-agent at some time step t. Then, our tracking methodology uses the RVO collision avoidance formulation to predict the next state, Ω, for a road-agent at time $t + 1$. Finally, the algorithm computes the ID of the road-agent at time $t + 1$ using a feature matching process described in [Wojke et al. 2017]. We now describe the tracking and detection processes separately in detail:

Detection. The main difficulty in detecting road-agents in dense traffic from a front camera can be attributed to an increased likelihood of occlusions. In such instances, bounding boxes are an inefficient visual representation. Therefore, we use Mask R-CNN to perform instance segmentation to reduce background clutter and generate efficient and occlusion-free representations (Figures 4).

Mask R-CNN begins by generating a set of bounding boxes for each road-agent in each frame. $\mathcal{B} = \{\mathbb{B} \mid \mathbb{B} = [\langle x, y\rangle_{\text{top left}}, w, h, s, r], r_i \in \mathcal{H}\}$ denotes the set of bounding boxes for each r_i at current time t, where $\langle x, y\rangle_{\text{top left}}$, w, h, s, and r denote the top left corner, width, height, scale, and aspect ratio of $\mathbb{B}$, respectively.

Mask R-CNN also outputs a set of masks, $\mathcal{M}$, for each corresponding bounding box. That is, $\mathcal{M} = \{\mathbb{M} \mid r_i \in \mathcal{H}\}$ denotes the set of masks for each r_i at current time t, where each $\mathbb{M}$ is a $[w \times h]$ matrix of boolean variables. Now, let $\mathcal{W} = \{\mathbb{W}(\cdot) \mid r_i \in \mathcal{H}\}$ be the set of white canvases where each canvas, $\mathbb{W} = [1]_{w\times h}$, w and h are the width and height of each $\mathbb{B}$ at current time t. Then,

$$\mathcal{S} = \{\mathbb{W}(\mathbb{M}) \mid \mathbb{W} \in \mathcal{W}, \mathbb{M} \in \mathcal{M}, r_i \in \mathcal{H}\},$$

is the set of background-subtracted segmented representations for each road-agent in the current frame. Note that during each iteration of the detection process, the IDs of the road-agents are known, and the task is to identify the road-agents at the next timestep.

Tracking. Our tracking by detection algorithm uses a non-linear motion model to predict the next state of the agent. Prior motion models with constant velocity or constant acceleration assumptions have been shown to not accurately model dense scenarios as they do not take into account collision avoidance behavior. Reciprocal Velocity Obstacles (RVO) [Van Den Berg et al. 2011] models collision avoidance behavior for dense scenes. RVO can be applied to pedestrians in a crowd as well as dense traffic environments. We use the RVO formulation and modify the formulation for segmented road-agents.

The RVO formulation requires an additional parameter v_{pref}. v_{pref} is the velocity the pedestrian would have taken in the absence of

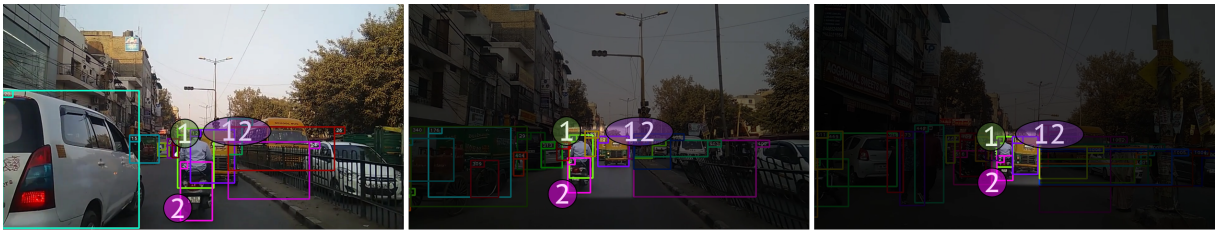


Figure 3: Qualitative analysis of our tracking algorithm on the TRAF dataset consisting of approximately 30 road-agents in the traffic scene. Frames are chosen with a gap equal to the length of the prediction window, *i.e.* 5 seconds(~ 150 frames). Each bounding box color is associated with a unique ID (displayed on the top left corner of each bounding box). Observation: Note the consistencies in the (ID)(color); for example, observe rider+two-wheeler (1-2)(green-pink) and three-wheeler (12)(purple).

obstacles or colliding pedestrians, computed using the standard RVO formulation.

The computation of the new state, Ω_{t+1}, is formulated as an optimization problem. For each road-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 [Van Den Berg et al. 2011]). If the ORCA constraints forbid a road-agent's preferred velocity, that road-agent chooses the velocity closest to its preferred velocity that lies in the feasible region, as given by the following optimization: $v_{\text{new}} = \text{argmin}_{v \notin ORCA} ||v - v_{\text{pref}}||$. The velocity, v_{new}, is then used to calculate the new position of a traffic agent.

To combine instance segmentation with RVO, we modify the state vector, Ω_t, to include bounding box information by setting the position to the centers of the bounding boxes. The centers of the bounding boxes are by extension, centers of each segmented road-agent.

3.3 Trajectory Prediction

The goal is to predict trajectories, *i.e.* temporal sequences of spatial coordinates of a road agent using neural networks. In standard neural networks, the objective function can be stated as $\min_w \|y - \phi(w^T x)\|$, where x is the input trajectory of a road-agent, ϕ is the non-linearity,



Figure 4: Efficient representations of road-agents in dense traffic. Instance segmentation helps reduce occlusions by reducing background clutter otherwise present in typical bounding box representations. These representations are used in our tracking by detection algorithm. The different colors simply denote individual road-agents.

and $\phi\left(w^T x\right)$ is the predicted trajectory of the road-agent. The objective is to learn a set of weights, w, that minimizes the oringal optimization equation. However, the difficulty of trajectory prediction in dense traffic lies in the fact that a road-agent's trajectory is affected by the trajectories of other road-agents in close proximity, not just it's own. This is especially prominent when agents are heterogeneous. For example, the maneuverability of a bus differs significantly from that of pedestrians; a pedestrian can change directions quickly while a bus cannot. Additionally, behavioral cues should also be emphasized in heterogeneous traffic. Aggressive drivers have non-uniform trajectories due to maneuvers like overspeeding, tailgating, and overtaking, while conservative agents tend not to stray from their current trajectories. However, the ability of non-linearity functions in neural networks to model behavioral aspects of human drivers is still an unsolved problem. This leads to a need for a model that accounts for interactions of a road-agent with other nearby road-agents.

In [Chandra et al. 2019], the authors propose such a model that considers two forms of interactions: spatial interactions based on the location of a neighboring road-agent with respect to the ego road-agent and heterogeneous interactions that consider the different static parameters (size) and dynamic parameters (steering angle) of road-agents inside a pre-computed neighborhood. The spatial interactions are motivated by the observation that, in dense traffic, a road-agent focuses primarily on road-agents that are in its horizon view (this is defined in the original paper). In heterogeneous interactions, however, the road-agent learns to assign adaptive weights to different heterogeneous road-agents. Combining the two interactions, the objective function is replaced by,

$$\min_w \left\| y - \left\{ \phi\left(w_x^T x\right) + \phi\left(w_s^T s\right) + \phi\left(w_h^T h\right) \right\} \right\|, \tag{2}$$

where $w = [w_x \; w_s \; w_h]^T$ is a vector comprising the weights for the input for the ego road-agent, the road-agents modeled by the spatial interaction formulation, and the road-agents modeled by the heterogeneous interaction formulation, respectively. Temporal sequence prediction requires neural networks that can capture temporal dependencies in data, such as LSTMs [Graves 2013]. 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

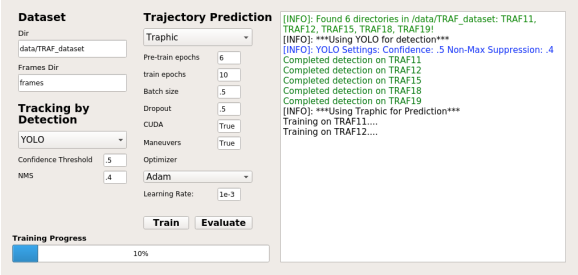


Figure 5: TrackNPred is a deep learning-based framework that integrates trajectory prediction methods with tracking by detection algorithms to motivate further research in end-to-end trajectory prediction. In this figure, we show the graphical user interface of TrackNPred where one can select the tracking 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.

have been used in computer vision applications with greater success because they can learn locally dependent features from images. Thus, to leverage the benefits of both, the authors of [Chandra et al. 2019] combine ConvNets with LSTMs to learn locally useful relationships (both in space and in time) between heterogeneous road agents.

4 TRACKNPRED: A SOFTWARE FRAMEWORK FOR END-TO-END TRAJECTORY PREDICTION

TrackNPred is a python-based software library [1] 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 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 [Gupta et al. 2018]), Recurrent Neural Networks (LSTMs [Alahi et al. 2016; Chandra et al. 2019], and Convolutional Neural Networks (CNNs [Chandra et al. 2019; Deo and Trivedi 2018]). 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 5).

Table 1: The list of algorithms currently implemented in TrackNPred.

	Methods
Tracking by Detection	Mask R-CNN + DeepSORT YOLO + DeepSORT
Trajectory Prediction	RNN- Encoder Decoder [Britz et al. 2017] Social-GAN [Gupta et al. 2018] Covolutional Social-LSTM [Deo and Trivedi 2018] TraPHic [Chandra et al. 2019]

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 [Alahi et al. 2016; Chandra et al. 2019; Deo and Trivedi 2018; Gupta et al. 2018; Ma et al. 2018b] 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[2] [Chandra et al. 2019]. 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 level (sparse/moderate/heavy/challenging). All the videos have a resolution of 1280×720.

4.1 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 1 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,

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

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

near-optimal initialization of each road-agent's state information in the first video frame. Further, methods that do not utilize 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 [Wojke et al. 2017] 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 [Van den Berg et al. 2008], 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.

Trajectory Prediction Module. TrackNPred currently supports the following end-to-end trajectory prediction algorithms: Social-Gan [Gupta et al. 2018], Convolutional Social-LSTM [Deo and Trivedi 2018], RNN Encoder-Decoder [Britz et al. 2017], and TraPHic [Chandra et al. 2019]. All trajectory prediction methods that are implemented in TrackNPred work in essentially the same manner. However, there are some differences which we highlight. Social-GAN [Gupta et al. 2018] was originally trained to predict the trajectories of pedestrians in a crowd. Additionally, CS-LSTM [Deo and Trivedi 2018] was designed to predict trajectories for road-agents in sparse and homogeneous traffic. Our goal is to perform trajectory prediction in dense and heterogeneous traffic environments. Therefore, we trained all three implementations on a real world dense and heterogeneous traffic dataset.

5 EXPERIMENTAL EVALUATION

We first compare the performance of our trajectory prediction algorithm, RobustTP, with trajectory prediction methods trained on manually annotated trajectories, on the TRAF dataset [Chandra et al. 2019] in Section 5.1. We also compare with the same methods when all are trained on sensor inputs. We use our software framework, TrackNPred for the second set of experiments in Section 5.2.

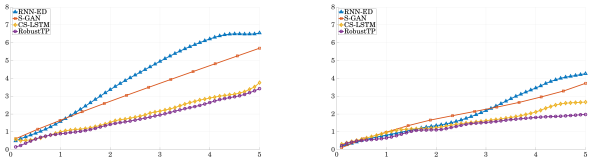


Figure 6: RMSE Curve Plot: We compare the RMSE-s of RobustTP with state-of-the-art end-to-end trajectory prediction methods on the TRAF dataset. All methods were trained using sensor inputs instead of manually annotated trajectories. The graphs were generated using TrackNPred. Overall, we observe that using YOLO for detection provides lower RMSE-s for all the methods. More importantly, the RMSE of RobustTP is consistently lower than all the other methods after the first second of prediction, implying it has better longer-term performance.

Table 2: We evaluate RobustTP with methods that use manually annotated trajectory histories, on the TRAF Dataset. The results are reported in the following format: ADE/FDE, where ADE is the average displacement RMSE over the 5 seconds of prediction and FDE is the final displacement RMSE at the end of 5 seconds. We observe that RobustTP is at par with the state-of-the-art.

RNN-ED	S-GAN	CS-LSTM	TraPHic	RobustTP
3.24/5.16	2.76/4.79	1.15/3.35	**0.78/2.44**	1.75/3.42

5.1 Comparison with Methods Using Manually Annotated Inputs

We use the following well-known metrics for evaluation:

- Average Displacement Error (ADE): The average of the root mean squared error (RMSE) between the ground truth and the predicted trajectory position at every time frame for the entire duration of 5 seconds. A lower ADE for a method implies that the method has a lower drift from the ground truth on the average, which is desirable.
- Final Displacement Error (FDE): The RMSE between the ground truth and the predicted trajectory position at the last time frame. A lower FDE for a method indicates it has a better prediction in the longer term.

In Table 2, we compare our algorithm that is trained on sensor inputs (using Mask R-CNN [He et al. 2017] for detection), with existing methods that, by contrast, use manually annotated ground truth trajectory history as training data. The RNN Encoder-Decoder model [Britz et al. 2017] is used for sequence modeling in many applications and has been adapted for trajectory prediction [Alahi et al. 2016; Deo and Trivedi 2018]. However, these approaches mainly target either sparse traffic or sparse pedestrian crowds. Therefore, we trained it from scratch on the TRAF dataset. Social-GAN [Gupta et al. 2018] is state-of-the-art for predicting trajectories of pedestrians in sparse scenarios, we, therefore, trained this method on the TRAF dataset from scratch as well. The CS-LSTM [Deo and Trivedi 2018], is designed for trajectory prediction for road-agents

Table 3: 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 [He et al. 2017] and YOLO [Redmon et al. 2015]. 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**

for sparse traffic, therefore we fine-tuned on top of it with the TRAF dataset.

After preparing the methods for fair evaluation, we observe that even with noisy trajectories as input, the ADE of our algorithm is 1.49 meters lower than the RNN Encoder-Decoder model [Britz et al. 2017] and 1.98 meters lower than Social-GAN [Gupta et al. 2018]. This is primarily because Social-GAN [Gupta et al. 2018] is trained for a single road-agent (pedestrians) in sparse scenarios, which does not transfer to modeling heterogeneous agents in dense traffic in the TRAF dataset. Additionally, our FDE is 2.72 meters and 2.35 meters lower than RNN-ED and S-GAN, respectively.

We also observe that the ADE of our algorithm is 0.6 meters higher than CS-LSTM [Deo and Trivedi 2018]. CS-LSTM weighs all agents in the neighborhood of the ego-agent equally. Thus it cannot adapt to dense, heterogeneous traffic. However, this problem is offset by the fact that we used manually annotated trajectory history as input to the CS-LSTM, resulting in overall better performance. Finally, compared to TraPHic [Chandra et al. 2019], our ADE is only slightly (0.97 meters) higher, since TraPHic also models dense and heterogeneous road-agents and uses ground truth trajectory history for prediction. Overall upon comparison, we note that RobustTP is at par with the state-of-the-art performance of TraPHic and CS-LSTM, while being able to achieve high accuracy by training on trajectories generated from sensor inputs.

5.2 Comparison with Methods Using Sensor Inputs

We use our software framework, TrackNPred, to run a set of experiments to evaluate the performance of end-to-end trajectory prediction methods using sensor inputs instead of manually annotated trajectories on the TRAF dataset [Chandra et al. 2019]. We use the output of two tracking methods in the library — one uses Mask R-CNN [He et al. 2017] for detection and the other uses YOLO [Redmon et al. 2015] for detection — to generate the input trajectory histories. We then use each of these trajectory histories as input to each of the trajectory prediction methods to obtain the trajectory predictions for the next 5 seconds. Finally, we compare and contrast their performances in Table 3. We also compare the RMSE curves produced by all these methods in Figure 6. For each detection model, we can observe that, when all the methods use trajectory history from noisy tracking data, our method has a clear advantage. For example, in the case of Mask R-CNN [He et al. 2017] as the detection model, in addition to outperforming both RNN-ED [Britz et al. 2017] and Social-GAN [Gupta et al. 2018] by 2.24 meters and 1.48 meters respectively, RobustTP also outperforms CS-LSTM by 0.16 meters on the ADE metric. It is also state-of-the-art on the FDE metric. The RMSE curves in Figure 6 further show that beyond the first second of prediction, the RMSE of RobustTP is consistently lower than all the other methods. Moreover, at the end of 5 seconds, the final RMSE of RobustTP is well below 4 meters (less than the length of an average car). Thus, RobustTP is more reliable than the other methods in longer-term prediction, which is a crucial benefit to consider when deciding trajectory prediction methods for real-world applications.

Finally, these experiments also serve to highlight how TrackNPred makes it convenient to benchmark the tracking and trajectory prediction methods that best suits specific end-to-end trajectory prediction tasks, thereby encouraging further research on novel end-to-end trajectory prediction algorithms.

6 CONCLUSION, LIMITATIONS, AND FUTURE WORK

We presented a novel end-to-end algorithm, RobustTP, for predicting the trajectories of road agents in dense and heterogeneous traffic. Our approach does not require manually annotated trajectories for training our model. We use 3 seconds of trajectory history as input and predict the next 5 seconds of the road-agent's trajectory.

RobustTP has some limitations. The size of the TRAF dataset prohibits training larger deep learning networks. Therefore, we cannot guarantee generalization to all forms of dense and heterogeneous traffic scenarios.

Regarding future work, RobustTP is a proof of concept and can be designed as an effective ADAS. The resulting ADAS would improve upon existing state-of-the-art trajectory prediction ADAS in many ways. First it is applicable to dense and heterogeneous traffic. Prior trajectory prediction-based ADAS [Barth and Franke 2009; Houenou et al. 2013; Weiskircher and Ayalew 2015] are either computationally expensive or susceptible to dynamic environments. RobustTP is computationally cost-effective since it uses a single RGB camera and the learning algorithm generalizes across scenes with varying dynamics. Second, it improves generalization to real-world scenarios. In autonomous driving, the degree to which a pre-trained trajectory prediction model generalizes the real world dynamics is a major concern. Online Hard Example Mining (OHEM) is an online machine learning training procedure wherein negatively classified training examples identified by the model would be added to the training set, and re-trained in an online manner. This has shown to increase generalization to new unseen test data, an advantage that is desired in autonomous driving. RobustTP, with its end-to-end capability integrating real-world trajectories in real-time, can support OHEM and offer better generalization in dense and heterogeneous traffic. We also plan to expand TrackNPred by adding more algorithms and providing more hyperparameter tuning capabilities.

REFERENCES

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

Alexander Barth and Uwe Franke. 2009. Estimating the driving state of oncoming vehicles from a moving platform using stereo vision. *IEEE Transactions on Intelligent Transportation Systems* 10, 4 (2009), 560–571.

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

Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. 2018. Long-term on-board prediction of people in traffic scenes under uncertainty. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4194–4202.

Jean-Yves Bouguet. 2015. Complete Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/

Gary Bradski and Adrian Kaehler. 2008. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

D. Britz, A. Goldie, T. Luong, and Q. Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints* (March 2017). arXiv:cs.CL/1703.03906

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

Fang-Chieh Chou, Tsung-Han Lin, Henggang Cui, Vladan Radosavljevic, Thi Nguyen, Tzu-Kuo Huang, Matthew Niedoba, Jeff Schneider, and Nemanja Djuric. 2019. Predicting Motion of Vulnerable Road Users using High-Definition Maps and Efficient ConvNets. *arXiv preprint arXiv:1906.08469* (2019).

Simon Danielsson, Lars Petersson, and Andreas Eidehall. 2007. Monte carlo based threat assessment: Analysis and improvements. In *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE, 233–238.

Nachiket Deo, Akshay Rangesh, and Mohan M. Trivedi. 2018. How would surround vehicles move? A Unified Framework for Maneuver Classification and Motion Prediction. *CoRR* abs/1801.06523 (2018). arXiv:1801.06523 http://arxiv.org/abs/1801.06523

Nachiket Deo and Mohan M Trivedi. 2018. Convolutional Social Pooling for Vehicle Trajectory Prediction. *arXiv preprint arXiv:1805.06771* (2018).

N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, and J. Schneider. 2018. Short-term Motion Prediction of Traffic Actors for Autonomous Driving using Deep Convolutional Networks. *ArXiv e-prints* (Aug. 2018). arXiv:1808.05819

Jonas Firl, Hagen Stübing, Sorin A Huss, and Christoph Stiller. 2012. Predictive maneuver evaluation for enhancement of Car-to-X mobility data. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 558–564.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623* (2015).

A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. 2018. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. *ArXiv e-prints* (March 2018). arXiv:cs.CV/1803.10892

K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask R-CNN. *ArXiv e-prints* (March 2017). arXiv:cs.CV/1703.06870

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

Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. 2013. Vehicle trajectory prediction based on motion model and maneuver recognition. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4363–4369.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Rudolph Emil Kalman. 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering* 82, Series D (1960), 35–45.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.

Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. 2017. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmán. 2011. Exploiting map information for driver intention estimation at road intersections. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 583–588.

Y. Ma, D. Manocha, and W. Wang. 2018a. AutoRVO: Local Navigation with Dynamic Constraints in Dense Heterogeneous Traffic. In *Computer Science in Cars Symposium (CSCS)*. ACM.

Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha. 2018b. TrafficPredict: Trajectory Prediction for Heterogeneous Traffic-Agents. *ArXiv e-prints* (Nov. 2018). arXiv:cs.CV/1811.02146

Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. 2009. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 261–268.

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. *CoRR* abs/1506.02640 (2015). arXiv:1506.02640 http://arxiv.org/abs/1506.02640

Matthias Schreier, Volker Willert, and Jürgen Adamy. 2014. Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 334–341.

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

Jur Van den Berg, Ming Lin, and Dinesh Manocha. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, 1928–1935.

Anirudh Vemula, Katharina Muelling, and Jean Oh. 2018. Social attention: Modeling attention in human crowds. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1–7.

Thomas Weiskircher and Beshah Ayalew. 2015. Predictive adas: A predictive trajectory guidance scheme for advanced driver assistance in public traffic. In *2015 European Control Conference (ECC)*. IEEE, 3402–3407.

N. Wojke, A. Bewley, and D. Paulus. 2017. Simple Online and Realtime Tracking with a Deep Association Metric. *ArXiv e-prints* (March 2017). arXiv:cs.CV/1703.07402

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

Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. 2018. Object Detection with Deep Learning: A Review. *arXiv e-prints*, Article arXiv:1807.05511 (Jul 2018), arXiv:1807.05511 pages. arXiv:cs.CV/1807.05511