Prediction-Based Reachability for Collision Avoidance in Autonomous Driving

Anjian Li\textsuperscript{1}, Liting Sun\textsuperscript{2}, Wei Zhan\textsuperscript{2}, Masayoshi Tomizuka\textsuperscript{2} and Mo Chen\textsuperscript{1}

Abstract—Safety is an important topic in autonomous driving since any collision may cause serious damage to people and the environment. Hamilton-Jacobi (HJ) Reachability is a formal method that verifies safety in multi-agent interaction and provides a safety controller for collision avoidance. However, due to the worst-case assumption on the car’s future actions, reachability might result in too much conservatism such that the normal operation of the vehicle is largely hindered. In this paper, we leverage the power of trajectory prediction, and propose a prediction-based reachability framework for the safety controller. Instead of always assuming for the worst-case, we first cluster the car’s behaviors into multiple driving modes, e.g. left turn or right turn. Under each mode, a reachability-based safety controller is designed based on a less conservative action set. For online purpose, we first utilize the trajectory prediction and our proposed mode classifier to predict the possible modes, and then deploy the corresponding safety controller. Through simulations in a T-intersection and an 8-way roundabout, we demonstrate that our prediction-based reachability method largely avoids collision between two interacting cars and reduces the conservatism that the safety controller brings to the car’s original operations.

I. INTRODUCTION

As the surge of deep learning and advanced sensor technology, there has been great progress for autonomous agent to perceive, analyze and predict the environment [1], [2], and autonomous driving seems not to be a distant dream. However, from time to time, there are reports that autonomous cars fail to maneuver safely and end up in crashes. Therefore, how to ensure safe control of the vehicle is becoming a critical and urgent issue. Especially for safe-critical system like autonomous cars, drones, and ground robots, where collision often causes painful damage, effective and verifiable safety controllers are in need.

Various work has been done on collision avoidance for autonomous agents. In traditional model-based optimal control, one often designs large cost near the obstacles or adds hard/chance constraints when planning the trajectory [3], [4]. Recently learning-based method has been adopted. Autonomous agents can learn collision avoidance controllers from expert demonstrations via either imitation learning [5], [6], [7] or inverse reinforcement learning [8], [9], [10]. Model-free reinforcement learning is also used to learn safety maneuvers in complex and dynamic environment [11], [12]. However, most of the above learning-based methods cannot handle safety constraints. For instance, imitation learning cannot guarantee the safety of the generated actions, particularly for the end-to-end imitation models. Inverse reinforcement learning cannot accurately recover the reward functions in the presence of unknown safety constraints, and reinforcement learning cannot generalize well to real vehicles due to its sim-to-real pipeline. Moreover, for deep imitation learning and reinforcement learning, it’s hard to analyze where the error comes from in the pipeline. Trajectories planned by finite-horizon decision and planning frameworks and methods [13] can satisfy safety constraints within the preview horizon, but safety cannot be guaranteed for infinite horizon if no terminal constraints are included and researchers resort to reachable-set-based methods [14] for such guarantee.

Hamilton-Jacobi (HJ) Reachability is a formal method that can verify the safety of the agents [15]. Given the dynamics of the agent and the collision set of states as the target set, it computes Backward Reachable Tubes (BRT), and the agent will be guaranteed safe when staying outside of the BRT [16], [17]. This is achieved by assuming worst-case control and disturbance inputs to the dynamics, and by computing the global optimal solution in entire state space via dynamic programming [18]. For collision avoidance between two cars, relative dynamics can be used where we give control of the “robot car” and consider the other “human car” as disturbances in the dynamics [19], [20].

Despite the advantages, HJ Reachability faces large challenges in autonomous driving. First, the traffic scenario is often very crowded with intensive interactions between cars. Since HJ Reachability always assumes the worst case of other cars’ actions, the BRT can be too conservative so that the robot car can hardly operate as normal. Second, computing BRT suffers from curse of dimensionality. Though approximating algorithms were developed [21], [22], [23], it’s still hard to solve any dynamics above 4D in real time.

There have been efforts focusing on less conservative and more practical BRTs. In [24], the authors designed an online learning framework to obtain more accurate bounds of wind disturbance. In traffic scenarios, [25] proposed the empirical reachable set, a probabilistic forward reachable set for cars. It rejects unlikely trajectories via non-parametric estimation. [26] modeled the interaction between the robot car and the human car as a pursuit evasion game and computed BRT for the relative systems, which is less conservative than [25]. In this case, BRT can be precomputed and used online. But they also assumed the worst case of the human car and a projected safety controller was needed to avoid too
aggressive behaviors.

Motivated by above methods, we find that trajectory prediction can help obtain more practical BRT since its outputs reduce the action range of the human car. Luckily, with the development of deep learning, state-of-the-art trajectory prediction algorithms achieve great performance. For example, [27] provided a probabilistic prediction of behaviors over candidate anchors, and [28], [29] proposed goal-conditioned trajectory prediction networks. [30] proposed generic features that consider both dynamic and static information in the traffic and succeeds in different scenarios.

The main contribution of this paper is to integrate general trajectory prediction into the design of reachability-based safety controllers to achieve more efficient two-car collision avoidance in real time with probabilistic safety guarantees. First, we model the two-car interaction as pursuit evasion game, with the human car being the pursuer. Our additional key insight is that given the prediction of the human car’s future trajectory, smaller action bounds can be used in the reachability computation, resulting in less conservative BRT. Second, a mode switch strategy is proposed to achieve real time BRT update. Here, the intuition is that if the car is turning left, it’s unlikely that it will suddenly turn right. Thus we cluster the human driver’s behaviors into six common driving modes with associated action bounds. The corresponding BRT for each mode is saved as look-up table and switched online according to the prediction outcomes. Our simulations show that our method not only preserves safety but also minimizes unnecessary impact to the car’s original operations.

Our paper is organized as follows. Section II briefly introduces the reachability formulation and the adopted prediction algorithm. In Sec. III, we cluster cars’ trajectories into common driving modes and introduce our mode classifier. In Sec. IV, we design a reachability-based safety controller for collision avoidance, and present the online strategy of mode switching. Finally Sec. V demonstrates the advantages of the proposed prediction-based safety controller with simulation in intersection and roundabout scenarios.

II. BACKGROUND

A. Hamilton-Jacobi Reachability

In this section, we introduce Hamilton-Jacobi (HJ) Reachability to verify safety in situations when collision may happen between two cars. In this two-car interaction, there’s one car we take control of, named the robot car, and another car for which we can only observe its actions, named the human car. We assume that the dynamics for the robot car and the human car are defined respectively by the ODEs

\[ \dot{z}_r(s) = f_r(z_r, u_r) \quad \text{and} \quad \dot{z}_h(s) = f_h(z_h, u_h), \]

where \( s \) represents time, \( z_r \in \mathbb{R}^{n_r} \) and \( z_h \in \mathbb{R}^{n_h} \) are states, \( u_r \in \mathcal{U}_r \) and \( u_h \in \mathcal{U}_h \) are controls for the robot and human car respectively. We, the robot car, aims to use \( u_r \) to avoid collision considering possible \( u_h \) of the human car.

We model this situation as traditional pursuit-evasion game [19]: the pursuer (human car) wants to make a catch and the evader (robot car) wants to avoid. Since we only care about how close two cars are instead of the exact location the collision happens, we use a relative dynamics \( z_{rel} = f_{rel}(z_{rel}, u_{rel}, d_{rel}) \) to define the joint system where the relative states \( z_{rel} \in \mathbb{R}^{n_{rel}} \) are constructed from \( z_r \) and \( z_h \). The controls of the relative dynamics \( u_{rel} \in \mathcal{U}_{rel} \) and the disturbances \( d_{rel} \in \mathcal{D}_{rel} \) are the robot control \( u_r \) and the human control \( u_h \). We define collision as the human car enters the target set \( \mathcal{T} \) which is usually some area centered around the robot car.

In HJ Reachability, given the above relative dynamics and target set as the collision set, we compute Backward Reachable Tube (BRT), representing states that will inevitably enter the target set within some time horizon \( T \) under worst disturbances \( d_{rel} \) despite best controls \( u_{rel} \) [16]. Numerically, we represent the target set \( \mathcal{T} \) as the sub-level set of a signed distance function \( l(z_{rel}) \), where \( z_{rel} \in \mathcal{T} \iff l(z_{rel}) \leq 0 \). Then with the final value function \( V(z_{rel}, 0) = l(z_{rel}) \), we can solve the following HJ equation to obtain \( V(z_{rel}, s) \) whose sub-level set representing the BRT [17]:

\[
\min \{ D_0 V(z_{rel}, s) + H(z_{rel}, \nabla V(z_{rel}, s)) V(z_{rel}, 0) - V(z_{rel}, s) \} = 0, \quad s \in [-T, 0]
\]

\[
H(z_{rel}, \nabla V(z_{rel}, s)) = \min_{d_{rel} \in \mathcal{D}_{rel}, u_{rel} \in \mathcal{U}_{rel}} \max_{d_{rel} \in \mathcal{D}_{rel}, u_{rel} \in \mathcal{U}_{rel}} \nabla V(z_{rel}, s) ^\top f_{rel}(z_{rel}, u_{rel}, d_{rel})
\]

The corresponding optimal safety controller \( u_{rel} \) is:

\[
u_{rel}^* = \arg \min_{d_{rel} \in \mathcal{D}_{rel}, u_{rel} \in \mathcal{U}_{rel}} \nabla V(z_{rel}, s) ^\top f_{rel}(z_{rel}, u_{rel}, d_{rel}).
\]

By discretizing the state space into grids, (1) can be solved via dynamic programming with level set method [31], [15]. In this paper we use the optimzed_dp toolbox [32] with HeteroCL [33] to solve BRTs.

B. Scenario-transferable probabilistic prediction

To predict the car’s future trajectory, we adopt the probabilistic prediction algorithm in [30]. Trajectory prediction in traffic is hard since it needs to consider the interactions between cars and the road constraints, e.g. curbs. The algorithm in [30] is built on generic representation of both static and dynamic information of the environment. Thus the prediction succeeds in predicting the future trajectory of the selected car in highly interactive traffic and can transfer to different scenarios, e.g. intersections or roundabouts.

It should be noted that our method is adaptable to any trajectory prediction algorithm as long as it can predict series of future positions for cars. However, the confidence of prediction will affect the probability of safety guarantee of our designed controller, discussed in Sec. IV-D.

III. DRIVING MODE ANALYSIS

In this section, we aim to derive common human driving modes with the associated action bounds for each mode, which helps compute less conservative BRT in Sec. IV. We first collect data of predicted car trajectories with algorithm in [30], which is trained on the real world INTERACTION
dataset [34]. Then we cluster the trajectory segments into several driving modes, e.g. left turn or right turn, based on the acceleration and angular speed. Finally we build a classifier to determine the probability over each driving mode when a new predicted trajectory is given. The procedure is summarized in the offline part in Fig. 3.

A. Trajectory collecting and processing

For the selected car in traffic, [30] predicts the n-step future trajectory \( \{(x_t, y_t, v_t)\}_{t=0}^{\Delta t} \) of the selected car, where \( x_t, y_t \) are the global x and y position, \( v_t \) is the speed and the time interval \( \Delta t = 0.1s \). The predicted trajectories are collected from two different scenarios: T-intersection and 8-way roundabout. Please refer to [34] for map details.

To obtain the human car’s action at each time step, we use the extended Dubins Car to model the human car’s dynamics

\[
\begin{align*}
\dot{x}_h &= v_h \cos \psi_h \\
\dot{y}_h &= v_h \sin \psi_h \\
\dot{\psi}_h &= a_h \\
\psi_h &= \omega_h 
\end{align*}
\]

where the states \( \{x_h, y_h, v_h, \psi_h\} \), and \( \psi_h \in [-\pi, \pi) \) is the orientation of the car. The controls of the dynamics, i.e. actions the human car can take, are accelerate \( a_h \in \mathbb{U}_{a_h} \) and angular speed \( \omega_h \in \mathbb{U}_{\omega_h} \). Then the action dataset, \( \{(a_{h,k}, \psi_{h,k})\}_{k=1}^{N} \), can be approximated from \( \{(x_t, y_t, v_t)\}_{t=0}^{\Delta t} \) using differential flatness [35], where \( N = 2365 \) for T-intersection and \( N = 1911 \) for 8-way roundabout scenario.

B. Driving mode clustering

Common patterns and modes can be extracted [36] from large amounts of human driving data. In this paper, we use clustering to extract patterns of driving based on the acceleration \( a_h \) and angular speed \( \omega_h \). Based on our knowledge of driving, before clustering, we pre-define six common driving modes in intersection and roundabout scenarios and set the default action \( a_0(m/s) \) and \( \omega_0(rad/s) \) for each mode:

- Mode 0: Deceleration, \( a_h = -1.5 \), \( \omega_h = 0 \)
- Mode 1: Stable, \( a_h = 0 \), \( \omega_h = 0 \)
- Mode 2: Acceleration, \( a_h = 1.5 \), \( \omega_h = 0 \)
- Mode 3: Left turn, \( a_h = 0 \), \( \omega_h = 0.2 \)
- Mode 4: Right turn, \( a_h = 0 \), \( \omega_h = -0.25 \)
- Mode 5: Roundabout, \( a_h = 0 \), \( \omega_h = 0.4 \)

For each \( (a_{h,k}, \omega_{h,k}) \) pair in the dataset, we first normalize them into \([-1, 1]\). Then we construct a 6D clustering feature \( \{d_{M0}, d_{M1}, d_{M2}, d_{M3}, d_{M4}, d_{M5}\} \), each term being the datapoint’s Euclidean distance to the six mode defaults. With these features, we use kmeans [37] to cluster all the action data into 6 driving modes like Fig. 1.

In Fig. 1, with the x- and y-axis to be acceleration and angular speed, “cross” data and light “dot” data are from roundabout and intersection scenarios, and the whole action dataset is clustered nicely into six driving modes.

C. Probabilistic mode classifier

For any pair \( (a_h, \omega_h) \), it might fall into the action range of zero, one or more modes. Thus given \( (a_h, \omega_h) \) we define its probability over each mode as follows: if it doesn’t fall into a range of any mode, it will be regarded as “Mode -1: Others”; if it only falls into the range of a single mode, it has 100% probability to be in that mode and 0% probability to be in other modes; if it falls into the ranges of a set of modes \( \{\text{Mode } i | i \in \sigma\} \), then the probability to be in Mode \( j, j \in \sigma \), is \((1/d_j)/(\sum_{i \in \sigma} 1/d_i)\), where \( d_j \) is the distance to the closest rectangle boundary of Mode \( i \).

IV. REACHABILITY-BASED SAFE CONTROLLER

In this section we discuss the reachability-based safety controller design for our robot car given the prediction of an observed human car. We will show that after specifying the future driving mode of the human car as in Sec. III, the BRT and safety controllers become less conservative but still maintain safety guarantee in a probabilistic sense.

A. System dynamics

We use pursuit-evasion game in Sec. II-A to model the pairwise interaction between two cars. For human car we use
Eq. (3) to define its dynamics. For our robot car we choose a higher fidelity bicycle dynamics $\dot{z}_r = f_r(z_r, u_r)$ [38]:

$$
\begin{align*}
\dot{x}_r &= v_r \cos(\psi_r + \beta_r) \\
\dot{y}_r &= v_r \sin(\psi_r + \beta_r) \\
\dot{\psi}_r &= a_r \\
\dot{\beta}_r &= \frac{I_f}{I_f + I_r} \tan(\delta_f)
\end{align*}
$$

where the state is $z_r = (x_r, y_r, \psi_r, \beta_r) \in \mathbb{R}$ are global $x$ and $y$ positions, $v_r \in \mathbb{R}$ is the speed, $\psi_r \in [-\pi, \pi]$ is the orientation of the inertia, and $I_f, I_r$ are the distance from the mass center to the front and rear axle. The control inputs are the acceleration $a_r \in \mathcal{U}_a$, and the steering angle $\delta_f \in \mathcal{U}_{\delta_f}$. The control bounds are chosen so that the robot car has the same acceleration and turning ability as the human car.

Similar to [26], we use a relative dynamics to define how close the two cars are evolved with time. The relative dynamics is centered around the robot car and also consistent with its coordinate frame. The relative $x$ position $x_{rel}$ is defined as the position in the robot car’s orientation, and the relative $y$ position $y_{rel}$ is perpendicular to the $x_{rel}$:

$$
\begin{bmatrix}
x_{rel} \\
y_{rel}
\end{bmatrix} = \begin{bmatrix}
\cos \psi_r & \sin \psi_r \\
-\sin \psi_r & \cos \psi_r
\end{bmatrix} \begin{bmatrix}
x_h - x_r \\
y_h - y_r
\end{bmatrix}
$$

(5)

The relative angle $\psi_{rel}$ is defined based on the robot car’s orientation $\psi_{rel} := \psi_h - \psi_r$. However, the speed of the two cars are in their own coordinate frame, so we include both of them individually. Finally we have the following 5D relative dynamics $\dot{z}_{rel} = f_{rel}(z_{rel}, u_{rel}, d_{rel})$:

$$
\begin{align*}
\dot{x}_{rel} &= v_r \cos(\beta_r) * y_{rel} + v_h \cos \psi_{rel} - v_r \cos \psi_r \\
\dot{y}_{rel} &= -v_r \sin(\beta_r) \times x_{rel} + v_h \sin \psi_{rel} - v_r \sin \psi_r \\
\dot{\psi}_{rel} &= \omega_h - v_r \sin(\beta_r) \\
\dot{\omega}_h &= a_h \\
\dot{\psi}_h &= a_r \\
\dot{\beta}_r &= \frac{I_f}{I_f + I_r} \tan(\delta_f)
\end{align*}
$$

(6)

The states $z_{rel} = (x_{rel}, y_{rel}, \psi_{rel}, v_h, \psi_r)$. The control inputs to this dynamics are the robot car’s controls $a_r \in \mathcal{U}_a$, and $\delta_f \in \mathcal{U}_{\delta_f}$, which we take charge of. The human car’s controls $a_h \in \mathcal{U}_a$ and $\omega_h \in \mathcal{U}_{\omega_h}$ are considered as disturbances since they may violate the goal of our robot car. Here $\mathcal{U}_a$ and $\mathcal{U}_{\omega_h}$ are equal to $\mathcal{U}_a$ and $\mathcal{U}_{\omega_h}$ in Sec. III, respectively.

**B. Collision avoidance between cars**

Given the relative dynamics, we set the target set $\mathcal{F}$ to be the collision set which is a rectangle centered around the robot car $\mathcal{F} := \{z_{rel} \mid |x_{rel}| \leq C_1, |y_{rel}| \leq C_2 \}$. The corresponding infinite time horizon BRT represents the states from which collision between the human car and the robot car is inevitable. In this paper, to approximate the infinite time horizon BRT, we compute Eq. (1) for a sufficient time horizon until the value function $V$ is converged.

One of our key contributions is to have less conservative BRTs, and it is achieved by having a smaller range of disturbance set $\mathcal{D}_{a_h}$ and $\mathcal{D}_{\omega_h}$. Recall that in Sec. III, we summarize six common driving modes with different $\mathcal{D}_{a_h}$ and $\mathcal{D}_{\omega_h}$ from clustering, thus we solve Eq. (1) for each mode individually and obtain the safety value $V(z_{rel})$ for Mode $i$, whose zero sub-level set is the BRT. Besides, we compute another $V_{-1}(z_{rel})$ and BRT for Mode -1 using the physical limit of the car.

In Fig. 2, we show the comparison of BRTs for different driving modes. Since the full BRT is 5D, we show the 2D slice at $\psi_{rel} = \pi/4, v_h = 6m/s, v_r = 1m/s$. From Fig. 2 BRT is the largest since it considers all possible controls of the human car as long as within the car’s physical limit, so it’s reasonable that the robot car has to be further away to maintain safety. If we believe that the human car is in some driving mode, for example in Mode 3: Left turn, its acceleration and angular speed will be restricted, and as a result, the BRT will be smaller and extend to the right side.

For Mode $i$, besides the safety value $V(z_{rel})$, we also compute Eq. (2) to obtain the robot car’s safety controller $\alpha_r$ and $\delta_f$ and save them as lookup table $U_{\alpha_r,i}(z_{rel})$ and $U_{\delta_f,i}(z_{rel})$. When the human car is in Mode $i$, our robot car will only track the safety value $V(z_{rel})$ and consider using safe controller $U_{\alpha_r,i}(z_{rel})$ and $U_{\delta_f,i}(z_{rel})$.

**C. Obstacle avoidance for curbs**

In real traffic scenarios, cars not only have to avoid the traffic in the lane, but also need to avoid collision with the curbs. Therefore to design a more practical safety controller, we incorporate another safety controller for curbs only.

Following the traditional reachability setting in [17], we set the target set to be the curb area obtained from the INTERACTION dataset [30], and use the dynamics in Eq. (4) to describe the robot car. By solving HJ equation, we can approximate the infinite time horizon BRT, and save the safety value $V_{curbs}(z_r)$ and safety controller $U_{\alpha_r,curbs}(z_r)$ and $U_{\delta_f,curbs}(z_r)$ as lookup tables.
D. Online mode switch strategy

Usually it is very hard to keep an up-to-date reachability for safety check when the car is operating online, because the reachability for 5D dynamics takes hours to compute. But with the precomputed safety values and safety controllers in Sec. IV-B and IV-C, we can achieve online safe control in real time with adaption to other vehicle’s actions.

We use a mode switch strategy shown as online part in Fig. 3. Here, accurate state estimation is assumed for both cars. The robot car continuously observes the human car’s behaviors to update the prediction of the human car’s future trajectory with algorithm in Sec. II-B. Every time the prediction is updated, we will infer the driving mode as described in Sec. III-C. Suppose the human car is in Mode $i$ and its original controllers are $\hat{a}_r$ and $\hat{\delta}_f$, we design hybrid controllers $\hat{a}_r$ and $\hat{\delta}_f$ for the robot car where the safety controller may take over as follows:

If $\min(V(z_{rel}), V_{curbs}(z_r)) > 0$, use original controllers $\hat{a}_r = \hat{a}_r, \hat{\delta}_f = \hat{\delta}_f$;

Else if $V(z_{rel}) \leq V_{curbs}(z_r)$, use safety controller for cars $\hat{a}_r = U_{a,c}(z_{rel}), \hat{\delta}_f = U_{\delta,f}(z_{rel})$;

Else, use safety controller for curbs $\hat{a}_r = U_{a,c,curbs}(z_r), \hat{\delta}_f = U_{\delta,f,curbs}(z_r)$. (7)

Under this strategy, the safety guarantee is preserved in a probabilistic way. Let $p_{predict}$ be the probability of the predicted trajectory from [30], and let $p_{mode}$ be the probability of this trajectory being in certain driving mode from Sec. III-C, and our designed safety controller in Eq. (7) can guarantee the safety of the human car and robot car with the probability $p_{safety} = p_{predict} \cdot p_{mode}$, with perfect modeling and state estimation assumed.

V. SIMULATION

In this section, we simulate the situation where a controlled robot car and a human car interact in two traffic scenarios. When their planned paths have some overlap, without any safety controller, collision may happen in various ways. We demonstrate that with our proposed prediction-based safety controller, the collision is largely avoided while unnecessary impact to the robot car is limited. In comparison, our baseline uses reachability-based safety controller without any prediction. Although safety is also preserved with the baseline method, the robot car deviates more from its originally planned path due to unnecessary and conservative override of the safety controller. The advantages of our method can also generalize across scenarios of intersection and roundabout.

![Fig. 3. Work flow of our method. Offline, we cluster the trajectory into driving modes and compute BRTs (represented by safety value) and safety controllers as lookup tables for each mode. Online, as the robot car is running, whenever a newly predicted trajectory is given, it will be classified as certain driving mode. Then the robot car will check the safety value and switch to the safety controller for that mode or curbs when necessary.](image)

### TABLE I

<table>
<thead>
<tr>
<th>Intersection</th>
<th>Method</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default</td>
<td>$\leq 0.5$</td>
<td>$\leq 1$</td>
<td>$\leq 0.5$</td>
</tr>
<tr>
<td></td>
<td>Reachability-Pred</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reachability-NoPred</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Roundabout</th>
<th>Method</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Default</td>
<td>$\leq 0.5$</td>
<td>$\leq 1$</td>
<td>$\leq 0.5$</td>
</tr>
<tr>
<td></td>
<td>Reachability-Pred</td>
<td>7</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Reachability-NoPred</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reachability-NoPred</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### TABLE II

<table>
<thead>
<tr>
<th>Intersection</th>
<th>Method</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reacchability-Pred</td>
<td>3.39</td>
<td>16.70</td>
<td>1.78</td>
</tr>
<tr>
<td></td>
<td>Reachability-NoPred</td>
<td>2.01</td>
<td>4.02</td>
<td>2.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Roundabout</th>
<th>Method</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reacchability-Pred</td>
<td>2.32</td>
<td>3.96</td>
<td>2.95</td>
</tr>
<tr>
<td></td>
<td>Reachability-NoPred</td>
<td>2.81</td>
<td>4.60</td>
<td>4.70</td>
</tr>
</tbody>
</table>

A. Simulation Details

1) Method: We compare three different methods of safety controller for the robot car. The first is our proposed prediction-based safety controller using reachability, called Reachability-Pred, which updates the BRT along with the latest prediction of the human car’s driving mode. The second is our baseline method which uses traditional reachability safety controller without prediction, called Reachability-NoPred. It keeps using the same BRT online considering all possible actions of the human car. The third method is the default controller where no safety controller is involved.

2) Path planning: We consider two traffic scenarios: a T-intersection and an 8-way roundabout. The original/default paths for both cars are all from the real world traffic data in the INTERACTION dataset [34]. Specifically, for our robot car, we let it follow a reference path which a real car has taken in the dataset. With the adopted Stanley steering control [39] and PID speed control, the robot car tracks the reference path with a constant target speed of 2 m/s. We also want the human car to imitate a road user’s behaviors. Since we need to predict the human
car’s future trajectories, to simplify, we just let the human 
car operates exactly like the prediction output which is very 
close to a realistic car trajectory. In this case, the trajectory 
prediction of the human car is assumed to be 100% correct.

3) Test case: For the T-intersection or 8-way roundabout, 
we each choose 3 different cases with different reference 
paths for the robot and human car. In each case, we run 10 
or 20 trials with different start positions for the robot car. 
The setting allows us to test how the safety controller reacts 
when the robot car meets the human car at its front, middle 
or the back side.

![Fig. 4](image1.png)
*Fig. 4. The trajectory comparison between Reachability-Pred and Reachability-NoPred when meeting a dangerous human car in roundabout scenario. The Reachability-NoPred deviates more to maintain safety than Reachability-Pred.*

![Fig. 5](image2.png)
*Fig. 5. The speed profiles comparison for Reachability-Pred and Reachability-NoPred in roundabout scenario. For Reachability-NoPred, the safety controller takes over earlier than Reachability-Pred and pushes the robot car to a more extreme speed.*

### TABLE III

<table>
<thead>
<tr>
<th>Intersection</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>car car</td>
<td>car curb</td>
<td>curbs urb</td>
</tr>
<tr>
<td>Reachability-Pred</td>
<td>7.81 9.26</td>
<td>7.81 0.00</td>
<td>13.76 2.95</td>
</tr>
<tr>
<td>Reachability-NoPred</td>
<td>7.95 1.76</td>
<td>7.81 0.00</td>
<td>17.10 5.52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Roundabout</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>car car</td>
<td>car curb</td>
<td>curbs urb</td>
</tr>
<tr>
<td>Reachability-Pred</td>
<td>11.45 0.00</td>
<td>11.19 1.33</td>
<td>10.10 0.00</td>
</tr>
<tr>
<td>Reachability-NoPred</td>
<td>12.95 0.00</td>
<td>17.90 1.19</td>
<td>11.14 0.00</td>
</tr>
</tbody>
</table>

### B. Simulation result

1) Safety controller vs No safety controller: First, we 
demonstrate how the safety controller helps collision avoid-
ance. We summarized the statistics of minimum distance 
between two cars over all trials. In Table I we count the 
number that the two cars are closer than 0.5m/1m in each 
case. In all case without safety controller, there are several 
trials that two cars are too close to each other and may 
cause collision. With Reachability-NoPred and Reachability-
Pred, collision is significantly reduced. Note that, although 
Reachability-Pred takes a much less conservative way for 
safety controller since it only considers a subset of human 
car’s actions, it’s almost as good as Reachability-NoPred for 
collision avoidance when a perfect prediction is given.

2) Prediction vs No prediction: Besides preserving safety 
of the car, our proposed Reachability-Pred enables smoother 
operation of the robot car with less impact from the safety 
controller compared to Reachability-NoPred. We verify this 
by computing the average deviation and maximum deviation 
from the planned path, and the time that the safety controller 
takes effect in each trial. We can see from Table II and 
Table III that, in every case besides case 1 in intersection, 
Reachability-NoPred makes the car deviate more from its 
path to maintain safety. The safety controller is activated 
more often and longer, which lowers the efficiency of the 
robot car for achieving its own goal.

3) Generalizability across scenarios: Our method demon-
strates its advantages in both intersection and roundabout 
scenarios. In Table I, II and III, Reachability-Pred has good 
collision rate, smaller deviation and less safety controller 
time.

### C. Error case analysis

We check the failure case of our designed safety controller, 
i.e., case 1 in intersection. The robot car first uses the safety 
controller to avoid the oncoming human car, which leads 
it to the boundary of the upper curb. Afterward, the safety 
controller for the curbs takes effect and forces the car to keep
going up, which further increases the deviation. Finally the car loses the ability to track its own reference path. To solve this, we need to have a planner with a better higher-level decision-making system, which is out of this paper’s scope.

VI. CONCLUSIONS

In this paper we present a prediction-based safety controller for two-car collision avoidance using HJ Reachability. For each clustered driving mode, less conservative BRT and safety controller are precomputed and then switched online when the prediction of the human car is updated. Simulation shows our work is superior in bringing less impact to the car’s original operation while maintaining safety.

We believe this is a step forward to make HJ Reachability-based safety controller more practical in crowded scenarios for autonomous cars and ground robots. For future work, we hope to extend our method to multi-agent collision avoidance, and want to incorporate it into vision-based perception and planning in partially observed environment.

REFERENCES

[34] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, “INTERACTION Dataset: An INTERNational, Ad-


