Online Monitoring for Safe Pedestrian-Vehicle Interactions

Peter Du, Zhe Huang[†], Tianqi Liu[†], Tianchen Ji[†], Ke Xu[†], Qichao Gao[†],
Hussein Sibai, Katherine Driggs-Campbell, and Sayan Mitra

Abstract— **As autonomous systems begin to operate amongst humans, methods for safe interaction must be investigated. We consider an example of a small autonomous vehicle in a pedestrian zone that must safely maneuver around people in a free-form fashion. We investigate two key questions: How can we effectively integrate pedestrian intent estimation into our autonomous stack? Can we develop an online monitoring framework to give rigorous assurances on the safety of such human-robot interactions? We present a pedestrian intent estimation framework that can accurately predict future pedestrian trajectories given multiple possible goal locations. We integrate this into a reachability-based online monitoring and decision making scheme that formally assesses the safety of these interactions with nearly real-time performance (approximately $0.1s$). These techniques are both tested in simulation and integrated on a test vehicle with a complete in-house autonomous stack, demonstrating safe interaction in real-world experiments.**

I. Introduction

Autonomous systems are quickly entering human dominated fields in the form of drones and self-driving cars, and becoming tangible technologies that will impact the human experience. As these systems begin to share space and operate among humans, safety becomes a primary concern [1], [2]. Despite many successful demonstrations of autonomous driving, safe interaction between autonomous vehicles and other road users remains an open problem [3]–[5].

We consider two key factors currently impeding safe interactive autonomy. First, designing interactive controllers that consider predictions about human movement remains a challenge due to the high variability in human behavior [6]. Second, providing formal guarantees for autonomous systems is challenging due to complexities introduced by many intertwined modules in the autonomous stack as well as multi-agent interactions [7].

One test case where this is particularly obvious is an autonomous shuttle scenario, where an autonomous vehicle operates on small roads and sidewalks and may encounter pedestrians through free-form interaction. Such automated systems have been developed for campus settings [8], [9] and pedestrian zones [10], [11]. These systems typically require and have heuristic decision-making for determining safe interaction (e.g., if an obstacle is within some preset distance, stop [12]). One common "lesson learned" is that interacting with pedestrians is difficult due to their unpredictability [9], [13]. As pedestrians are vulnerable agents and have very few constraints on their motion, determining methods for safe interaction is both critical for reducing risk of accidents and improving efficiency by limiting over-conservative performance. Further, many studies state that an on-board safety driver or watchdog is necessary to monitor the system operation [9], [11].

In this work, we present an initial foray into integrating human motion prediction with an autonomous system's control framework as well as an online monitoring system. Leveraging data-driven verification approaches to provide formal guarantees, we demonstrate how DryVR, an offline reachability tool for hybrid dynamical systems [14], can be used in an online setting to give a formal assessment of safety as the autonomous vehicle drives in close proximity to a human. In short, we present the following contributions:

1) *Intent estimation.* We present a particle filter-based approach to predict human motion and provide future trajectories as well as estimate their likelihoods over multiple goals for each agent.
2) *Safety monitoring.* We develop a predictive reachability analysis technique. This near-realtime, forward reachability analysis allows the autonomous system to make risk-aware decisions, and can help designers assess safety of the overall system.
3) *System integration.* We demonstrate the feasibility of our efforts on a complete autonomous system which includes pedestrian tracking and intent prediction, localization, waypoint tracking control, and an online reachability analysis module.

The paper is organized as follows. In Section II, we present a literature review of works related to modeling pedestrian intent and online monitoring, as applied to autonomous systems. Then, we present an overview of our experimental platform, in-house autonomous stack, and methodology for pedestrian intent estimation in Section III. In Section IV, we discuss our approach for online monitoring via reachability. We conclude with an analysis of our integrated system and discuss findings and future directions in Sections V and VI.

II. Background

We present a brief literature review of techniques for modeling pedestrian interactions for use in decision-making and control frameworks as well as verification methods to provide guarantees for autonomous systems.

A. Pedestrian Modeling

There are three main techniques used to predict pedestrian trajectories: data-driven approaches like neural networks,

*This research is supported by research grants from the National Science Foundation (FMITF:1918531) and the United States Air Force Office of Scientific Research (FA9550-17-1-0236). We are grateful to the Illinois Center for Autonomy, Coordinated Science Lab, and AutonomouStuff for supporting our experimental infrastructure.

[†] Z. Huang, T. Liu, T. Ji, K. Xu, and Q. Gao contributed equally.

P. Du, Z. Huang, T. Liu, T. Ji, Q. Gao, H. Sibai, K. Driggs-Campbell, and S. Mitra are with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. email: {peterdu2,zheh4,tliu51,tj12,qgao10,sibai2,krdc,mitras}@illinois.edu

K. Xu is with the Department of Computer Science at the University of Illinois at Urbana-Champaign. email: kexu6@illinois.edu

decision-making approaches like Markov models, and filtering approaches like Kalman filters.

Neural networks, especially Long Short Term Memory Networks (LSTM) [15]–[17], have been successfully implemented to model social interactions between humans in crowded scenarios by integrating neighbor information into pedestrian trajectory prediction. Yi et al. encoded pedestrian behavior into sparse displacement volumes, which are used as input and output of the Behavior-CNN they proposed [18]. These methods are built on the assumption that the observation data provides perfect pedestrian positions, so they may not guarantee robust performance under noisy measurements.

There has also been plenty of research where pedestrian behavior modeling is treated as a sequential decision making problem [19]–[22]. Jump-Markov process is proposed in [19] to model pedestrian behavior, which assumes that the goal is slowly time-varying and that the pedestrians are rational. Partially Observable Markov Decision Process (POMDP) [20] and Mixed Observability Markov Decision Process (MOMDP) [21] are used to incorporate intents as hidden variables. In different problem settings, intents may be defined as goals [21], sub-goals [20], [23] or a policy in a Markov Decision Process [19]. In our study, the intent always refers to the desired goal of the pedestrian.

Filtering is one of the most common methods for state estimation in real world applications. Filtering can be applied to not only pedestrian tracking [24]–[26] but also pedestrian movement prediction [27]–[30]. Moreover, particle filtering integrated with intent hypotheses model pedestrian behavior as a multi-hypotheses dynamical system, and thus capture the switching feature of complicated pedestrian dynamics [29], [30]. Thus, our work is built on [29] to provide effective pedestrian intent estimation given noisy measurements.

B. Reachability Analysis and Safety Verification

The *reachability* problem asks, given a system model $\mathcal{A}$, an initial set of states Θ, and a particular target state $\mathbf{x}^*$, whether any behavior of $\mathcal{A}$ starting from Θ can reach $\mathbf{x}^*$? Reachability analysis has played a fundamental role in both control theory and formal methods for cyber-physical systems [31]–[34]. A key application of reachability analysis that is relevant for this paper is in *safety verification*: If the target $\mathbf{x}^*$ is defined as an *unsafe state*, then answers to the reachability question can determine whether the system $\mathcal{A}$ can become unsafe starting from Θ.

It is well-known that the reachability problem is *undecidable* for general dynamical and hybrid system models [35]. Researchers have developed algorithms for relaxations of the problem by considering (a) finite time horizon behaviors of $\mathcal{A}$, (b) allowing false-positives in the answers, and (c) allowing answers with probabilistic guarantees. A series of breakthroughs have led to the creation of software tools and libraries that can perform effective reachability analysis. For example, SpaceEX can perform reachability analysis of linear hybrid systems with dozens of continuous dimensions [36]; HyLAA has successfully verified sparse linear models with thousands of dimensions [37], [38]. Nonlinear models with hundreds of dimensions can now be verified with tools like C2E2 [39], [40], Flow* [41], and CORA [42]. All of these approaches are geared towards *off-line safety verification* and they have been applied to autonomous vehicles [40], medical devices [43], and space operations [44].

Online reachability analysis can be used for predictive safety monitoring and decision making for an autonomous vehicle. In this setup, $\mathcal{A}$ is the model of the vehicle, and the initial set Θ is instantiated to be the current state estimate for this system (e.g., state of vehicle and pedestrians). Then reachability analysis up to a *look-ahead* time horizon $T_{Look} > 0$ can predict whether the vehicle is going to be unsafe within T_{Look} time. For this analysis to be useful the reachable set computation has to finish before the next control decision, which is typically 10-100 milliseconds. We will revisit these issues in Section IV.

The feasibility of online verification for automatic lane change maneuvers on a Cadillac SRX was demonstrated in [45], [46]. This analysis used linearized approximations of the vehicle dynamics relative to the behaviors of all possible surrounding cars. In contrast, our reachability analysis and decision module uses pedestrian intents. Further, our data-driven reachability analysis approach can directly handle nonlinear vehicle models. In [47], a research platform that is similar to ours was presented, where uncertainty of the predicted behavior of the pedestrians is taken into account but the vehicle dynamics is not used for reachability. In a similar vein, [48] uses reachability analysis to check safety of human-driven vehicles. This work also uses linearized dynamics and does not implement the solution on a full autonomous vehicle nor does it address pedestrian intent detection. Several works use reachability for safe controller synthesis, typically assuming perfect sensor information within static environments [49], [50]. Finally, in [51], an approach similar to ours is followed in the context of safety relative to neighboring human driven vehicles.

III. Autonomy Modules

We describe our autonomous testbed in terms of the experimental hardware and the software components used in our autonomous stack.

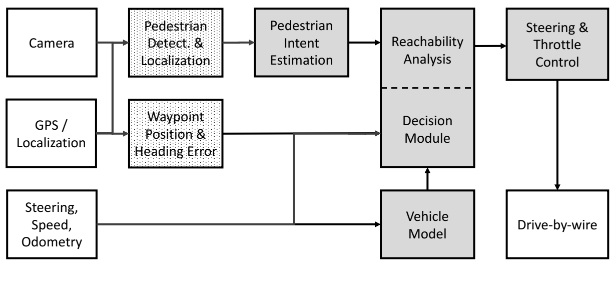
Camera
Pedestrian Detect. & Localization
Pedestrian Intent Estimation
Reachability Analysis
Steering & Throttle Control
GPS / Localization
Waypoint Position & Heading Error
Decision Module
Steering, Speed, Odometry
Vehicle Model
Drive-by-wire

Fig. 1: System architecture. The dark gray modules constitute the key contributions of this work.

A. Vehicle Hardware and Experimental Setup

Our vehicle is a Polaris GEM electric car outfitted with sensing and computing hardware by AutonomouStuff (Figure 2). The sensors include a Velodyne LIDAR, a Radar, GPS & Inertial Measurement Units, and a single Mako G-319C forward facing camera. Our experimental arena is a 3,000 sq. ft. indoor open research space.

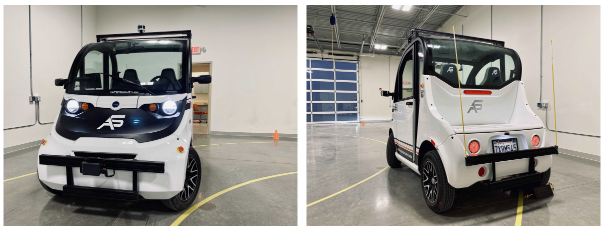
Fig. 2: Polaris GEM E2 in our testing arena.

B. Localization and Pedestrian Detection

We implemented a localization protocol that gives 2D global coordinates of the vehicle using Decawave ultra-wideband beacons and time-of-flight calculations. To improve the precision and reduce the errors caused by noise, Kalman filtering is applied on the position measurements received from Decawave. In our testing, the localization protocol can provide a 2D localization accuracy within 20 cm and message drop rates less than one per minute with position data update rates of 3Hz. Our indoor testing facility was unobstructed by walls and allowed for a clear line of sight among the Decawave chips.

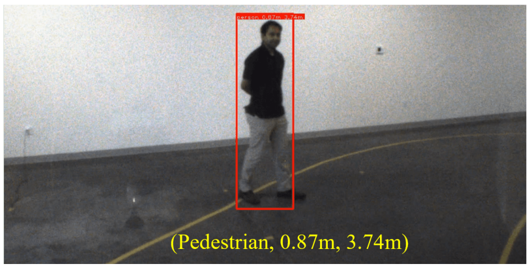

Fig. 3: Detected pedestrian and position estimates.

For pedestrian detection, we use YOLOv3: a state-of-the-art real-time object detection system. Once a pedestrian is detected on an input image, YOLOv3 places a bounding box around it. To estimate the longitudinal distance y of the detected pedestrian with respect to the vehicle, we use the size of bounding box, focal length of the camera, and assumed width of the pedestrian (0.47 m). The same is done to estimate the lateral deviation x. The estimated x and y-coordinates are converted from vehicle coordinates to global coordinates using position data from the vehicle's localization. Using this approach, we get an observed error on the pedestrian's location that does not exceed $0.5m$.

C. Pedestrian Intent Estimation (PIE) Module

The pedestrian intent module takes a pedestrian position measurement in the global coordinate frame and produces a future trajectory that ends at a predefined goal state. The predicted trajectory is then fed into the decision module (*DM*) to assess the safety of the driving scenario and advise the controller accordingly.

The estimation algorithm assumes that the map information and all possible pedestrian intents are given as a set of possible goal locations. For example, in a crosswalk scenario, the pedestrian may have two possible intents. One is to cross the road, which is represented by an intent on the other side of the road. The other is to go alongside the road, for which the intent can be set as a location on the same side of the road as the pedestrian. Based on this assumption, the sequential measurements of the pedestrian are used to predict its intent.

An accurate pedestrian model is necessary to predict the intent. The Generalized Potential Field Approach (GPFA) is used to model the pedestrian behavior [52]. A set of sources are created to represent the obstacles in the map and to generate the repulsive force on the pedestrian, while a sink is set at goal locations to attract the pedestrian. The sum of forces is the acceleration, which is the control input u_t to the state space model of the pedestrian, giving the following dynamical model:

$$\begin{aligned} x_t &= F x_{t-1} + G u_{t-1} + w_t \\ y_t &= H x_{t-1} + v_t \end{aligned} \tag{1}$$

where F is the state transition matrix, G is the control matrix, x_t is the state vector (position and velocity) of the pedestrian at time t, H is the measurement matrix, and y_t is the measurement of the pedestrian's location. It is assumed that the process noise w_t and the measurement noise v_t are Gaussian white noise.

As the transition model involves a highly nonlinear GPFA, we employ a Multi-hypothesis Particle Filter to estimate the likely intent [29]. Initially, each particle sampled will be assigned a hypothesis on pedestrian intent I_i. The number of intents N considered in this paper is three, corresponding to either crossing the vehicle's path or heading parallel along the path in either direction. The probability of assigning different intents to a particle is the same, as $1/N_I$ for each intent. In the prediction step, each particle predicts the pedestrian position at the next time stamp based on the hypothesis of the intent to which it is assigned. In the update step, the weights of particles are updated based on the deviation of the position prediction from the measurement.

In contrast to the method presented in [29], the weights of particles with the same intent hypothesis are summed to generate the probability distribution of different intents at that timestamp. Sequential importance re-sampling is implemented after the weights are updated. The intent with the highest probability is chosen to create the trajectory of the pedestrian in the future using GPFA based on the current measured position. In [29], the re-sampling step was not needed due to the robustness requirement for changes of the true intent of the pedestrian. However, no re-sampling would make the intent estimation vulnerable to sample degeneracy (i.e., only few particles with significant weights are kept throughout the intent estimation process), which leads to the loss of population nature of the particle filter. In fact, sequential importance re-sampling makes use of the quantity of particles to inherit the probability distribution inferred from the last round and causes no harm to the algorithm's robustness. Another difference is that though many of the model parameters are the same as in [29], [52], no additional noise is manually added due to the sensor noise inherent in the measurement. For online performance, we initially generate 200 particles for each intent (600 total).

Ultimately, this module produces the estimated pedestrian trajectory as a sequence of location-time pairs $\tau = \{\langle p_1, t_1\rangle, \ldots, \langle p_k, t_k\rangle\}$, where the final point is the estimated goal location. The algorithm runs at 5 Hz, which is sufficient

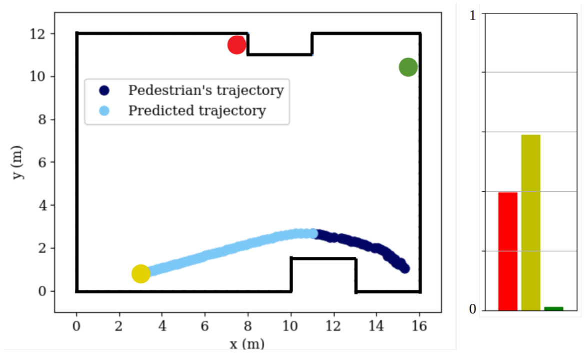

Fig. 4: Pedestrian intent estimation for scenario with three goals (red, yellow, green). *Left:* Predicted trajectory. *Right:* Probability distribution of pedestrian intents.

for the online monitoring application. Figure 4 shows an example of the algorithm output on a public dataset [53]. There are three possible intents in this case (shown in red, yellow, and green). The pedestrian trajectory is in dark blue and the predicted trajectory is in light blue. Our methodology is well tuned and validated on this public dataset [53], accurately predicting the intent and effectively producing the future trajectory.

D. Velocity and Waypoint Following Control

We developed a control module to generate throttle and steering commands to navigate the vehicle autonomously. The control module is integrated into our ROS framework to interface with the other modules of the GEM vehicle platform as well as the PACMod drive-by-wire system (see Figure 1). We use two standard PID controllers, one for maintaining the vehicle at a desired speed v_r (set by the user), and a second one for steering the vehicle through a sequence of predefined waypoints. Spline curve fitting is used to get a planned path through the waypoints. The lateral deviation of the vehicle from the path is obtained at each timestep, and used to generate a steering velocity command with the PID steering velocity controller.

The speed controller reads the current vehicle speed and outputs acceleration commands to track the reference speed v_r. It takes an additional input from the reachability-based decision module (*DM*), which chooses the mode of operation between $brake$ and $trackspeed$, based on the the safety of the predicted behavior of the car. If the *DM* chose the $trackspeed$ mode, the PID speed controller sets the acceleration a of the vehicle to track v_r. If the *DM* chose the $brake$ mode instead, the speed controller releases the throttle and applies the brakes to bring the vehicle to a stop. In both modes, the steering velocity output u is taken from the PID steering controller to track the planned path.

In summary, the inputs to the control module include the state of the vehicle shown in Section IV, reference constant speed v_r, lateral deviation from the planned path, as well as a chosen mode of operation from the reachability-based decision module *DM*. The outputs include the throttle (acceleration/braking) a and steering velocity control u.

IV. Online Predictive Reachability Analysis Module (*OPRA*)

In this section, we discuss how our predictive online reachability analysis module works to inform the controller, and helps avoid collisions with pedestrians.

A. Online Reachability

Recall from Section II, that for online safety verification or monitoring, the estimated current states Θ of the vehicle and the pedestrian(s) are propagated forward in time for a look-ahead period T_{Look}, according to the model $\mathcal{A}$, to check whether there can be a loss of safe separation within that time. For this to be effective, the online reachability analysis must be: (1) *accurate*—to reduce false positives, and (2) *quick*—to allow for timely recovery and mitigating actions. Specifically, for $T_{Look} = 3$ to 5 seconds look-ahead, the reachability analysis must finish within 10-100 milliseconds. Current offline reachability algorithms for hybrid systems need minutes of computation time for 6 to 12-dimensional nonlinear models. Making the problem more challenging, the uncertainties in the initial position, heading, and speed, defining the set Θ may be considerable because of estimation errors (see Sec. III-A). The larger these errors are, the larger Θ needs to be to achieve the same level of confidence in that it contains the actual initial state of the car, and consequently achieve the same level of confidence that the corresponding reach set contains the future state of the car. Further, there may be multiple pedestrian and vehicle trajectories to propagate forward and multiple modes to evaluate. In this paper, we identify a restricted class of scenarios, and for those, we show how this is achieved in our *OPRA*.

Given a dynamical system $\mathcal{A}$ over a state space $\mathcal{X}$ and having an input space $\mathcal{U}$, a mode space $\mathcal{P}$, a dynamics function $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, a closed loop controller function $g : \mathcal{X} \times \mathcal{P} \to \mathcal{U}$, a set of possible initial states $\Theta \subseteq \mathcal{X}$, a mode $p \in P$, and a look-ahead time $T > 0$, a state $\mathbf{x} \in \mathcal{X}$ is reachable if there exists an execution of $\mathcal{A}$ starting from Θ, having a fixed mode p, and of duration at most T, that reaches $\mathbf{x}$. The set of reachable states is written as $\mathsf{Reach}_{\mathcal{A}}(\Theta, p, T) \subseteq \mathcal{X}$. Our design of *OPRA* uses the data-driven reachability approach as implemented in the DryVR [40] and C2E2 [39] tools. Roughly speaking, simulation data generated from a model or an executable for $\mathcal{A}$, is used together with sensitivity analysis of $\mathcal{A}$ to compute over-approximations of $\mathsf{Reach}_{\mathcal{A}}(\Theta, p, T)$. We are able to achieve the online-level computation times (as shown in Table I) by using a few tricks that may be broadly applicable: (a) we fixed the function β quantifying sensitivity off-line, (b) we capped the number of simulation traces used—this sacrifices precision for lowering computation, (c) we use low-dimensional dynamical vehicle models that are adequate for simple safety requirements like separation. The loss of accuracy from these approximations is partly mitigated by computing multiple reach sets from different initial sets corresponding to different levels of uncertainty or confidence in state estimation.

In more detail, our reachability analysis procedure is adopted from the algorithm developed in [14]. First, in an off-line procedure, the sensitivity function β of the system is learned from sampling system trajectories. This function bounds the distance between trajectories (or solutions) of $\mathcal{A}$ starting from different initial states. For this step, we use the DryVR tool of [14].

For the online computation, consider an initial state estimate Θ with radius r. This set is partitioned into m regions. For each region i, a numerical simulation $\xi(\mathbf{x_i}, \mathbf{t})$ of $\mathcal{A}$ starting from a representative state $\mathbf{x_i}$ is computed up to the look-ahead time T_{Look}. The over-approximation of the reach set $\mathsf{Reach}_{\mathcal{A}}(\Theta, p, T_{Look})$ is computed by *bloating* each simulation $\xi(\mathbf{x_i}, \mathbf{t})$ with β, and then taking the union of all these sets. It has been shown in earlier works that [39], [54], provided the sensitivity function is accurate, the computed reachable set can be made arbitrarily precise by increasing m. The output $\mathsf{Reach}_{\mathcal{A}}(\Theta, p, T_{Look})$ is used to detect separation with the pedestrian's predicted paths by the decision module. Different r can be chosen based on the desired level of confidence. If we want a higher level of confidence, a larger r will be preferable, which will increase the covered region of the initial set and consequently, the reach set $\mathsf{Reach}_{\mathcal{A}}(\Theta, p, T_{Look})$. Thus, the risk of actual trajectory of the vehicle not being covered by the reach set will be mitigated.

Vehicle model: We use the standard 5-dimensional bicycle model for the vehicle, which has been shown to accurately forecast an autonomous vehicle state [55], where f is given by:

$$\dot{x} = v\cos\theta,\ \dot{y} = v\sin\theta,\ \dot{\phi} = u,\ \dot{v} = a,\ \dot{\theta} = \frac{v}{L}\tan(\phi)$$

where u is the input steering angular velocity and a is the input acceleration that are determined by the velocity and waypoint following controller g of Section III-D, (x, y) is the position, v is the speed, θ is the steering angle, ϕ is its heading angle and L is the length of the car. Hence, the state space $\mathcal{X} = \mathbb{R}^5$, the control space $\mathcal{U} = \mathbb{R}^2$, and the set of modes is $\{\mathit{trackspeed}, \mathit{brake}\}$. The mode affects the dynamics indirectly by determining the control inputs computed in Section III-D. In the *brake* mode, we set the reference speed to zero and apply the resulting PID control, as there is no brakes to apply in the bicycle model as opposed to the real car.

State estimation uncertainties: There are several sources of errors and uncertainties in measuring the initial state of the above system. For example, the position of the car and the pedestrian have uncertainties arising from the localization system. We define a nested sequence of initial sets $\Theta_1 \subset \Theta_2 \subset \ldots \Theta_k$—increasingly conservative and with increasing levels of confidence. For each Θ_i we compute corresponding over-approximations of $\mathsf{Reach}_{\mathcal{A}}(\Theta_i, p, T)$ using the above procedure starting from Θ_i. From the monotonicity of $\mathsf{Reach}_{\mathcal{A}}$, it follows that $\mathsf{Reach}_{\mathcal{A}}(\Theta_1, p, T) \subseteq \mathsf{Reach}_{\mathcal{A}}(\Theta_2, p, T) \subseteq \ldots \mathsf{Reach}_{\mathcal{A}}(\Theta_k, p, T)$, and therefore, if $\mathsf{Reach}_{\mathcal{A}}(\Theta_i, p, T)$ is verified safe, then so are the smaller sets. Thus, if $\mathsf{Reach}_{\mathcal{A}}(\Theta_i, p, T)$ is verified safe in the *DM*, then computation stops.

B. Evaluation of online reachability

Implementing these strategies with a look-ahead time of $T_{Look} = 3$ seconds gave us a maximum reachability computation time of only 0.1 seconds on the vehicle's computers as shown in Table I. We can see as well, from the same table, that the computation time of reach sets increases linearly with the look-ahead time. This shows the promise of our approach particularly because our current implementation of *OPRA* does not yet exploit parallelism of the simulation step of the algorithm.

TABLE I: Reach Set Computation Time

T_{Look} (s)	3.0	3.5	4.0	4.5	5.0
Compute Time (s)	0.096	0.103	0.129	0.136	0.163

TABLE II: Reach Set Accuracy

T_{Look} (s)	3.0	3.5	4.0	4.5	5.0
High Conf. (%)	98.905	98.338	98.617	97.825	96.851
Med Conf. (%)	98.161	96.629	97.138	95.917	94.874
Low Conf. (%)	95.825	94.078	94.672	93.416	92.078

The results in Table II show that the bicycle vehicle dynamics and the reach set obtained accurately predict the behavior of the real car. Even with the lowest confidence reach sets obtained from the smallest initial sets considered and longest look-ahead time, we acheived 92% accuracy, while the maximum is about 99%. To calculate the accuracy of the reach sets generated by the *OPRA*, we replay several simulations offline and check at each timestep t of the simulation whether the reach set contains the real trajectory of the vehicle. We consider the computed reach set to be "accurate" for a given timestep t if this is true.

C. Decision Module

For any initial set of states Θ, mode p, and look-ahead time $T_{Look} > 0$, if the intersection $R \cap U$ of an over-approximation $R \supseteq \mathsf{Reach}_{\mathcal{A}}(\Theta, p, T_{Look})$ and an unsafe set U, is empty, then we can safely decide that the system is safe up to time T_{Look}. On the other hand, if $R \cap U \neq \emptyset$ then we cannot infer that there necessarily exists an unsafe execution.

The decision module (*DM*) runs the reachability module *OPRA* on the mode *trackspeed* and checks if the resulting over-approximation of the reach set R is safe when the unsafe set U is the union of the reach sets of all surrounding pedestrians. Their reach sets are computed by bloating the trajectories obtained from the *PIE* module using a constant sensitivity function equal to r, where r is the radius of the initial set representing the uncertain position of the pedestrian. If $R \cap U \neq \emptyset$, the decision module *DM* sets the mode for the controller to *brake* to avoid possible collision. Otherwise, it sets it to *trackspeed*.

The decision module *DM* can run the reachability module *OPRA* on mode *brake* as well, to check if there is a possibility of unavoidable collision, and alert the pedestrian. It can also compute the reach sets using initial sets with different radii to get different levels of confidence in safety.

Lastly, the results from Tables I and II show that *DM* can operate accurately in real time to ensure safety while preserving a smooth ride with little unnecessary braking.

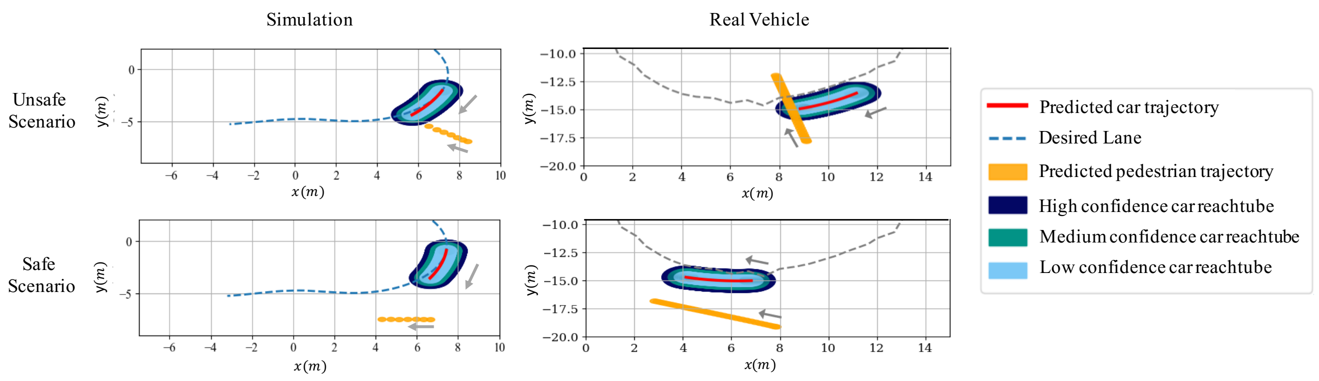

Fig. 5: Experimental results showing predicted pedestrian trajectories and vehicle reachtubes. Left two graphs show experiments done in simulation. Right two graphs show experiments performed on real vehicle.

V. Experimental Results

In this section, we present the results of our monitoring system, both in simulation and on our test vehicle.

A. Simulation

We created a simulation environment in Gazebo (shown in Fig. 6) to test our vehicle controllers and online reachability analysis module before deployment on a physical system.[1] The scenario used consists of a vehicle following a curved path with a pedestrian walking in close proximity.

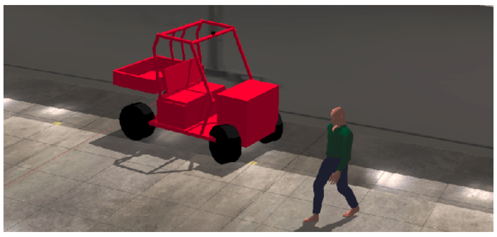

Fig. 6: Gazebo simulation environment

Fig. 5 shows the results of the reachability analysis computed by the *OPRA*. It generates three sets of reach sets corresponding to high, medium, and low confidence levels. The different levels of confidence can be used to encode the uncertainty associated with the vehicle's starting state. The graphs in the upper and lower left hand side of Fig. 5 show simulation results. In the first case, the pedestrian moves parallel to the vehicle's path and the *OPRA* predicts that the future trajectory of the vehicle will be safe and allows it to continue. In the second case, the pedestrian crosses the vehicles path and the *OPRA* sends a signal to the controller telling the vehicle to brake.

B. Real-World Testing

To demonstrate our complete pipeline (with off the shelf components and in-house autonomous modules), we run our autonomous system in two test cases. Fig. 5 shows two examples of the our integrated system on a real vehicle. In both cases, as the vehicle comes around the curved lane in the test arena, a pedestrian is detected and tracked.

[1]Gazebo vehicle model adapted from [56].

The *PIE* module estimates the goal location and passes the predicted trajectory to the *OPRA*. In the lower right portion of Fig. 5, the predicted trajectory shows the pedestrian walking near the lane without intersecting the vehicle's path. As a result, the vehicle proceeds to safely move past the pedestrian without braking at any point during the experiment. In the upper right portion of Fig. 5, the predicted trajectory suggests that the pedestrian will cross the lane. The *OPRA* decides this situation is unsafe as a collision is likely and signals the controller to brake.

The *OPRA* gets estimates of the pedestrian's intent and vehicle state and performs reachability analysis to infer whether the system is safe up to a look-ahead time T_{Look}. As discussed earlier, the state estimates can have large and fluctuating errors and thus we use nested uncertainty bounds with different levels of confidence. These confidence levels are illustrated in the high, medium, and low confidence reach sets in Figure 5. Depending on the level of risk one can tolerate, the decision to brake can be made using a higher or lower confidence reach set.

Similar to what we observed in simulation, our experiments on a real vehicle, for a look-ahead time of $T_{Look} = 3$ seconds, saw reachability computation times of less than 0.1 seconds on average. Increasing the number of reach sets by using initial sets with different confidence levels or allowing reach set refinements can give more precise monitoring at the cost of increased computation.

VI. Discussion

We present an integrated autonomous system that uses a novel pedestrian intent estimator to safely maneuver amongst humans. We added another layer of safety and risk assessment by developing a reachability-based online monitoring scheme that formally assesses the safety of these interactions with nearly real-time performance ($\sim 0.1s$). These techniques are tested in simulation and integrated on a test vehicle with a complete in-house autonomous stack, demonstrating effective and safe interaction in real-world experiments.

In our current experiments, we make many assumptions and control many aspects of the system to ensure the baseline performance is functional. However, in reality, any of the submodules may fail. For example, we only tested scenarios where the *PIE* correctly predicted the pedestrian. Given that no model can perfectly predict a human, we hope to explore scenarios where the prediction is incorrect, making the need for an online monitor more pressing. We note that similar statements hold for the vision and localization modules. We also found that timing and synchronization of the components were difficult and had a noticeable impact on the safety assessment. We hope to include such inaccuracies and uncertainties in future iterations of our online monitor.

References

[1] A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, J. F. Fisac, S. Deglurkar, A. D. Dragan, and C. J. Tomlin, "A scalable framework for real-time multi-robot, multi-human collision avoidance," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[2] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions," in *Robotics Science and Systems (RSS)*, 2018.

[3] S. M. Thornton, F. E. Lewis, V. Zhang, M. J. Kochenderfer, and J. C. Gerdes, "Value sensitive design for autonomous vehicle motion planning," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[4] B. Chen, D. Zhao, and H. Peng, "Evaluation of automated vehicles encountering pedestrians at unsignalized crossings," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[5] K. Driggs-Campbell, V. Govindarajan, and R. Bajcsy, "Integrating intuitive driver models in autonomous planning for interactive maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3461–3472, 2017.

[6] K. Driggs-Campbell, R. Dong, and R. Bajcsy, "Robust, informative human-in-the-loop predictions via empirical reachable sets," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 300–309, 2018.

[7] V. Govindarajan, K. Driggs-Campbell, and R. Bajcsy, "Data-driven reachability analysis for human-in-the-loop systems," in *IEEE Conference on Decision and Control (CDC)*, 2017.

[8] P. Rodríguez, "Safety of pedestrians and cyclists when interacting with automated vehicles—a case study of the wepods," Ph.D. dissertation, Master thesis, TU Eindhoven, 2017.

[9] J. W. Van der Wiel, "Automated shuttles on public roads: Lessons learned," in *ITS European Congress*, 2017.

[10] G. Eden, B. Nanchen, R. Ramseyer, and F. Evéquoz, "On the road with an autonomous passenger shuttle: Integration in public spaces," in *CHI Extended Abstracts on Human Factors in Computing Systems*, 2017.

[11] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "Autonomous robot navigation in highly populated pedestrian zones," *Journal of Field Robotics*, vol. 32, no. 4, pp. 565–589, 2015.

[12] S. Nordhoff, J. de Winter, R. Madigan, N. Merat, B. van Arem, and R. Happee, "User acceptance of automated shuttles in Berlin-Schöneberg: A questionnaire study," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 58, pp. 843–854, 2018.

[13] E. Trulls, A. Corominas Murtra, J. Pérez-Ibarz, G. Ferrer, D. Vasquez, J. M. Mirats-Tur, and A. Sanfeliu, "Autonomous navigation for mobile service robots in urban pedestrian environments," *Journal of Field Robotics*, vol. 28, no. 3, pp. 329–354, 2011.

[14] C. Fan, B. Qi, S. Mitra, and M. Viswanathan, "Dryvr:data-driven verification and compositional reasoning for automotive systems," in *Computer Aided Verification*, 2017.

[15] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[16] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[17] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, "Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[18] S. Yi, H. Li, and X. Wang, "Pedestrian behavior understanding and prediction with deep neural networks," in *European Conference on Computer Vision*, 2016.

[19] V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto, "Intent-aware long-term prediction of pedestrian motion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[20] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[21] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 475–491.

[22] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 201–214.

[23] T. Ikeda, Y. Chigodo, D. Rea, F. Zanlungo, M. Shiomi, and T. Kanda, "Modeling and prediction of pedestrian behavior based on the sub-goal concept," *Robotics*, vol. 10, 2013.

[24] H. Wang, H. Lenz, A. Szabo, J. Bamberger, and U. D. Hanebeck, "Wlan-based pedestrian tracking using particle filters and low-cost mems sensors," in *IEEE 4th workshop on positioning, navigation and communication*, 2007.

[25] M. Meuter, U. Iurgel, S.-B. Park, and A. Kummert, "The unscented kalman filter for pedestrian tracking from a moving host," in *IEEE Intelligent Vehicles Symposium (IV)*, 2008.

[26] X. Wang and Z. Tang, "Modified particle filter-based infrared pedestrian tracking," *Infrared Physics & Technology*, vol. 53, no. 4, pp. 280–287, 2010.

[27] N. Schneider and D. M. Gavrila, "Pedestrian path prediction with recursive bayesian filters: A comparative study," in *German Conference on Pattern Recognition*. Springer, 2013, pp. 174–183.

[28] E. Rehder and H. Kloeden, "Goal-directed pedestrian prediction," in *IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2015.

[29] F. Particke, M. Hiller, C. Feist, and J. Thielecke, "Improvements in pedestrian movement prediction by considering multiple intentions in a multi-hypotheses filter," in *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018.

[30] F. Particke, C. Hofmann, M. Hiller, H. Bey, C. Feist, and J. Thielecke, "Entropy-based intention change detection with a multi-hypotheses filter," in *IEEE International Conference on Information Fusion (FUSION)*, 2018, pp. 610–616.

[31] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.

[32] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*, 1st ed. Springer, 2009.

[33] J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. S. Sastry, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 2–17, 2003.

[34] A. M. Bayen, E. Cruck, and C. Tomlin, "Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: solving the hamilton-jacobi equation using viability techniques," in *HSCC*, ser. LNCS, C. Tomlin and M. R. Greenstreet, Eds., vol. 2289. Springer, 2002, pp. 90–104.

[35] T. A. Henzinger and P. -H. Ho, "Algorithmic analysis of nonlinear hybrid systems," in *Proceedings of the 7th International Conference On Computer Aided Verification*, 1995, pp. 225–238.

[36] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 379–395.

[37] S. Bak and P. S. Duggirala, "Hylaa: A tool for computing simulation-equivalent reachability for linear systems," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, 2017, pp. 173–178.

[38] S. Bak, H. Tran, and T. T. Johnson, "Numerical verification of affine systems with up to a billion dimensions," in *ACM International Conference on Hybrid Systems: Computation and Control, HSCC*, 2019.

[39] P. S. Duggirala, S. Mitra, and M. Viswanathan, "Verification of annotated models from executions," in *EMSOFT*, 2013.

[40] C. Fan, B. Qi, and S. Mitra, "Data-driven formal reasoning and their applications in safety analysis of vehicle autonomy features," *IEEE Design & Test*, vol. 35, no. 3, pp. 31–38, 2018.

[41] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds. Springer Berlin Heidelberg, 2013, vol. 8044, pp. 258–263.

[42] M. Althoff and D. Grebenyuk, "Implementation of interval arithmetic in CORA 2016," in *Proc. of the 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2016, pp. 91–105.

[43] X. Chen, S. Dutta, and S. Sankaranarayanan, "Formal verification of a multi-basal insulin infusion control model," in *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2017, pp. 75–91.

[44] N. Chan and S. Mitra, "Verified hybrid LQ control for autonomous spacecraft rendezvous," in *IEEE Conference on Decision and Control, CDC*, 2017.

[45] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[46] M. Althoff and J. M. Dolan, "Set-based computation of vehicle behaviors for the online verification of autonomous vehicles," in *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011.

[47] L. Ferranti, B. Brito, E. Pool, Y. Zheng, R. M. Ensing, R. Happee, B. Shyrokau, J. F. P. Kooij, J. Alonso-Mora, and D. M. Gavrila, "Safevru: A research platform for the interaction of self-driving vehicles with vulnerable road users," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[48] P. Falcone, M. Ali, and J. Sjoberg, "Predictive threat assessment via reachability analysis and set invariance theory," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1352–1361, Dec 2011.

[49] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments," *CoRR*, vol. abs/1905.00532, 2019.

[50] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, July 2019.

[51] K. Leung, E. Schmerling, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone, "On infusing reachability-based safety assurance within probabilistic planning frameworks for human-robot vehicle interactions," in *International Symposium on Experimental Robotics*, 2018.

[52] F. Particke, L. Patino-Studencki, J. Thielecke, and C. Feist, "Pedestrian tracking using a generalized potential field approach." in *VISIGRAPP (6: VISAPP)*, 2017, pp. 509–514.

[53] B. Majecka, "Edinburgh informatics forum pedestrian database," *URl: http://homepages. inf. ed. ac. uk/rbf/FORUMTRACKING*, 2010.

[54] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification (CAV 2010)*, ser. Lecture Notes in Computer Science. Springer, 2010, vol. 6174, pp. 167–170.

[55] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015.

[56] "Polaris urdf," https://wiki.aalto.fi/download/attachments/151495770/Final_report_project_12.pdf?api=v2, accessed: 2020-02-28.